ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΙΣ ΕΦΑΡΜΟΣΜΕΝΕΣ ΜΑΘΗΜΑΤΙΚΕΣ ΕΠΙΣΤΗΜΕΣ

Διπλωματική εργασία

Δέντρα ταξινόμησης, Boosting και Bagging Μέθοδοι για την κατασκευή μοντέλων

Μ.Φ.: Ελενη Γκρουγια
Επιβλεπων Καθηγητης: Χρηστος Κουκουβινος

Αθηνα, Φεβρουαριος 2013

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή στα δέντρα απόφασης

1.1 Εισαγωγή

Τα CART (Classification & Regression Trees) αποτελούν μια κλάση απαραμετρικών, μη-γραμμικών μοντέλων πρόβλεψης τα οποία διαχωρίζονται σε δύο κατηγορίες: σε δέντρα ταξινόμησης (Classification trees) και σε δέντρα παλινδρόμησης (Regression trees). Η βασική ιδέα της μεθοδολογίας είναι πολύ απλή. Αρχικά τα CART διαμερίζουν τα δεδομένα σε δύο υποσύνολα, έτσι ώστε οι καταγραφές εντός των υποσυνόλων να είναι περισσότερο ομοιογενείς μεταξύ τους συγκρινόμενες από όταν βρίσκονταν στο ίδιο σύνολο. Πρόκειται για μια επαναληπτική (recursive) διαδικασία, καθώς καθένα από αυτά τα δυό υποσύνολα διασπάται ξανά, με τη διαδικασία να επαναλαμβάνεται μέχρι να επιτευθεί κάποιο κριτήριο ομοιογένειας ή μέχρι να ικανοποιηθεί κάποιο άλλο κριτήριο τερματισμού [8].

Υπενθυμίζουμε ότι το ζητούμενο είναι η πρόβλεψη της τάξης ή της απόκρισης Y μέσα από τα δεδομένα $X_1, X_2, \ldots, X_p$. Όταν εφαρμόζεται κάποιο πολύπλοκο μοντέλο μη-γραμμικής παλινδρόμησης, υποθέτουμε ότι αυτό ισχύει για ολόκληρο το χώρο των δεδομένων. Στην περιπτωση όμως που τα δεδομένα έχουν χαρακτηριστικά τα οποία αλληλεπιδρούν μεταξύ τους με πολύπλοκους, μη-γραμμικούς τρόπους, τότε η χρήση του μοντέλου παλινδρόμησης δεν βοηθά [1]. Διότι ακόμη και αν μπορέσουμε να εφαρμόσουμε το μοντέλο, η ερμηνεία του θα είναι εξαιρετικά περίπλοκη.

Συνεπώς μια εναλλακτική προσέγγιση της μη-γραμμικής παλινδρόμησης είναι αρχικά η διαμέριση του χώρου σε μικρότερα κομμάτια, όπου μπορούμε να χειριστούμε ευκολότερα τις αλληλεπιδράσεις. Και κατόπιν η εφαρμογή απλών μοντέλων σε όλα τα κελιά της διαμέρισης που προέκυψαν.

Σχηματικά αναπτύσσεται ένα δυαδικό δέντρο. Εντός αυτού, σε κάθε κόμβο του (node) εφαρμόζεται ένα τεστ που αφορά σε μια μεταβλητή ξεχωριστά. Ανάλογα με το αποτέλεσμα του τεστ, μετακινούμαστε είτε προς την αριστερή είτε προς την δεξιά διακλάδωση του δέντρου. Κάποια στιγμή φτάνουμε σε τελικό κόμβο ή αλλιώς σε φύλλο (terminal node/leaf) όπου γίνεται μια πρόβλεψη.

Η διαδικασία CART αποτελείται από τρία βήματα. Στο πρώτο βήμα χτίζεται το πλήρες δέντρο διαδοχικά με τη χρήση δυαδικών ερωτήσεων σχετικά με τις τιμές κάθε μεταβλητής ξεχωριστά. Βασικό μειονέκτημα του πλήρους δέντρου είναι το overfitting. Στο δεύτερο βήμα αντιμετωπίζεται το ζήτημα του overfitting με τη διαδικασία του «κλαδέματος» (pruning) του δέντρου σε ένα ικανοποιητικό μέγεθος. Αυτή η διαδικασία παράγει μια σειρά λιγότερο πολύπλοκων δέντρων. Τέλος στο τρίτο βήμα

επιλέγεται το βέλτιστο δέντρο χρησιμοποιώντας τη διαδικασία του cross-validation.

1.2 Δημιουργώντας το δέντρο

1.2.1 Συμβολισμοί

Η μέθοδος της διαμέρισης μπορεί να εφαρμοστεί σε διαφορετικά είδη δεδομένων. Εξετάζουμε αρχικά το πρόβλημα της Ταξινόμησης το οποίο έχει πιό πολύπλοκες εξισώσεις. Έστω ότι έχουμε ένα δείγμα n παρατηρήσεων οι οποίες προέρχονται από C τάξεις. Το μοντέλο διαμερίζει τις παρατηρήσεις σε k ομάδες σε κάθε μια εκ των οποίων ανατίθεται μια προβλεπόμενη τάξη η οποία είναι η εξαρτημένη μεταβλητή του προβλήματος μας [15].

π_i $i = 1, 2, \ldots, C$ Εκ των προτέρων πιθανότητες κάθε τάξης

$L(i,j)$ $i = 1, 2, \ldots, C$ Πίνακας με τις απώλειες λόγω λανθασμένης ταξινόμησης κάποιου i ως j Προφανώς $L(i, i) = 0$

A Κάποιος κόμβος του δέντρου ο οποίος αντιπροσωπεύει, μέσω του δέντρου που τον παρήγαγε, έναν κανόνα ταξινόμησης για μελλοντικά δεδομένα.

$\tau(x)$ Η πραγματική τάξη μιας παρατήρησης x,
όπου x είναι το διάνυσμα των ανεξάρτητων μεταβλητών.

$\tau(A)$ Η τάξη που ανατίθεται στον A, εάν θεωρηθεί ως τελικός κόμβος ή αλλιώς φύλλο

n_i, n_A Το πλήθος των παρατηρήσεων του δείγματος που ανήκουν στην τάξη i και το πλήθος των παρατηρήσεων του κόμβου A

n_{iA} Το πλήθος των παρατηρήσεων του δείγματος που ανήκουν στην τάξη i και στον κόμβο A

$P(A)$ Πιθανότητα του κόμβου A (για μελλοντικές παρατηρήσεις)
$= \sum_{i=1}^{C} \pi_i P\{x \in A | \tau(x) = i\} \approx \sum_{i=1}^{C} \pi_i \{n_{iA}/n_i\}$

$p(i|A)$ $P\{\tau(x) = i | x \in A\}$ (για μελλοντικές παρατηρήσεις)
$= \pi_i P\{x \in A | \tau(x) = i\} / P\{x \in A\}$
$\approx \pi_i (n_{iA}/n_i) / \sum_{i=1}^{C} \pi_i (n_{iA}/n_i)$

$R(A)$ Το ρίσκο του κόμβου A
$= \sum_{i=1}^{C} p(i|A) L(i, \tau(A))$ όπου το $\tau(A)$ επιλέχθηκε ώστε να ελαχιστοποιεί το ρίσκο

$R(T)$ Το ρίσκο ενός μοντέλου/δέντρου T
$= \sum_{j=1}^{k} P(A_j) R(A_j)$ όπου A_j είναι οι τελικοί κόμβοι του δέντρου

Εάν $L(i,j) = 1$ για όλα τα $i \neq j$ και θέσουμε τις εκ των προτέρων πιθανότητες π να είναι ίσες με τις παρατηρούμενες συχνότητες κάθε τάξης μέσα στο δείγμα, τότε $p(i|A) = n_{iA}/n_A$ και $R(T)$ είναι η αναλογία λανθασμένης ταξινόμησης.

1.2.2 Κριτήρια Διαχωρισμού

Εάν ένας κόμβος A διαχωριστεί στα αριστερά στον κόμβο A_L και δεξιά στον A_R, τότε έχουμε[1]

$$P(A_L)R(A_L) + P(A_R)R(A_R) \leq P(A)R(A).$$

Ένας προφανής τρόπος ανάπτυξης ενός δέντρου είναι η επιλογή του διαχωρισμού εκείνου ο οποίος μεγιστοποιεί την ποσότητα ΔR, δηλαδή τη μείωση στο ρίσκο. Όμως η συγκεκριμένη προσέγγιση έχει κάποια μειονεκτήματα όπως φαίνεται απ'το ακόλουθο παράδειγμα [15].

Έστω ότι οι απώλειες είναι ίσες, ότι το 80% των δεδομένων είναι τάξης 1 και ότι κάποιος δοκιμαστικός διαχωρισμός έχει ως αποτέλεσμα στον A_L κόμβο το 54% να είναι τάξης 1 ενώ στον A_R το 100% να είναι τάξης 1. Η εξαρτημένη μεταβλητή είναι δίτιμη και μετρά την τάξη (τάξεις 0 και 1). Επειδή η ελάχιστη πρόβλεψη ρίσκου τόσο για τον κόμβο A_L όσο και για τον A_R είναι ίσες με $\tau(A_L) = \tau(A_R) = 1$, γι'αυτό το διαχωρισμό θα έχουμε $\Delta R = 0$ παρόλο που αυτό αποτελεί μια πολύ πληροφοριακή διαμέριση του δείγματος. Σε πραγματικά δεδομένα με μια τέτοια πλειοψηφία οι πρώτοι διαχωρισμοί συχνά έχουν την ίδια απόδοση.

Ένα σοβαρότερο μειονέκτημα της μεγιστοποίησης της ποσότητας ΔR είναι ότι η μείωση του ρίσκου είναι ουσιαστικά γραμμική. Εάν υπήρχαν δύο ανταγωνιστικοί μεταξύ τους διαχωρισμοί, με τον έναν να διαμερίζει τα δεδομένα σε δύο ομάδες καθαρότητας 85% και 50%, και με τον άλλον σε δύο ομάδες καθαρότητας 70% και επίσης 70% τότε θα προτιμούσαμε τον πρώτο. Και αυτό στην περίπτωση που μοναδικό κριτήριο επιλογής μας ήταν ποιός εκ των δύο προσφέρεται για καλύτερους μελλοντικούς διαχωρισμούς.

Το πακέτο *rpart* χρησιμοποιεί ένα μέτρο μη-καθαρότητας (measure of impurity) ενός κόμβου. Έστω f κάποια συνάρτηση μη-καθαρότητας, τότε η μη-καθαρότητα του κόμβου A ορίζεται ως εξής

$$I(A) = \sum_{i=1}^{C} f(p_{iA})$$

όπου p_{iA} είναι η αναλογία των μονάδων του κόμβου A που ανήκουν στην τάξη i για μελλοντικά δείγματα. Αφού ιδανικά θα θέλαμε $I(A) = 0$ όταν ο A είναι καθαρός, η f πρέπει να είναι κοίλη συνάρτηση με $f(0) = f(1) = 0$.

Δυό υποψήφιες συναρτήσεις f είναι ο δείκτης πληροφορίας $f(p) = -plog(p)$ και ο δείκτης Gini $f(p) = p(1-p)$. Τότε χρησιμοποιείται ο διαχωρισμός με τη μέγιστη μείωση της μη-καθαρότητας

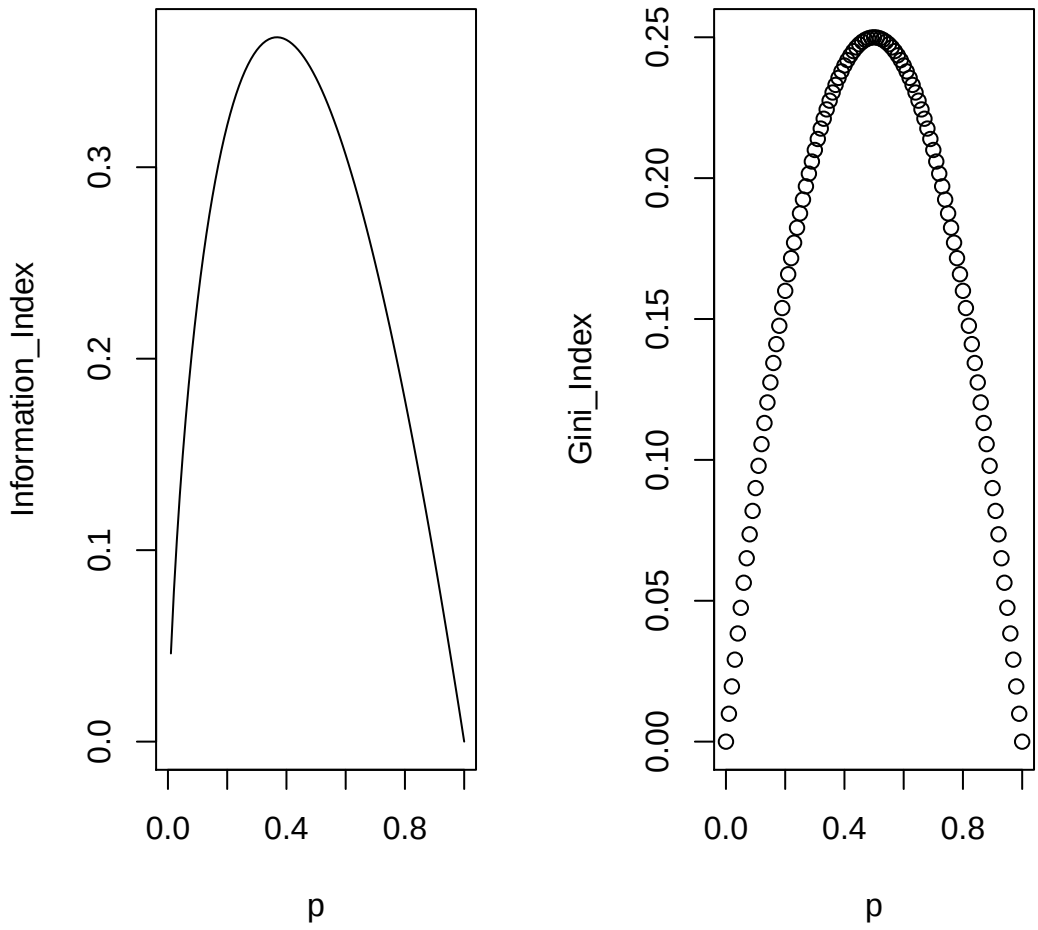
$$\Delta I = p(A)I(A) - p(A_L)I(A_L) - p(A_R)I(A_R)$$

.

1.2.3 Ενσωματώνοντας τις απώλειες

Στα CART υπάρχουν δύο διαφορετικοί τρόποι επέκτασης των κριτηρίων μη-καθαρότητας οι οποίοι περιλαμβάνουν τις απώλειες. Είναι ο γενικευμένος δείκτης Gini και οι τροποποιημένες εκ των προτέρων πιθανότητες. Το πακέτο *rpart* χρησιμοποιεί μόνο τη μέθοδο των τροποποιημένων εκ των προτέρων πιθανοτήτων.

[1] Αποδεικνύεται στο βιβλίο *Classification and Regression Trees* των L. Breiman, J.H. Friedman, R.A. Olshen και C.J Stone, Εκδόσεις Wadsworth, Belmont CA, 1983.



Σχήμα 1.1: Οι συναρτήσεις μη-καθαρότητας.

Ο γενικευμένος δείκτης Gini Μια ενδιαφέρουσα ερμηνεία του απλού δείκτη Gini είναι η ακόλουθη. Υποθέτουμε ότι ένα αντικείμενο επιλέγεται τυχαία από μια εκ των C τάξεων σύμφωνα με το διάνυσμα πιθανοτήτων $(p_1, p_2, \ldots, p_C)$ και τυχαία του ανατίθεται μια τάξη με τη χρήση της ίδιας κατανομής. Η πιθανότητα λανθασμένης ταξινόμησης είναι η ακόλουθη,

$$\sum_i \sum_{j \neq i} p_i p_j = \sum_i \sum_j p_i p_j - \sum_i {p_i}^2 = \sum_i 1 - {p_i}^2$$

Έστω $L(i,j)$ η απώλεια για τη λανθασμένη ταξινόμηση ενός αντικειμένου στην τάξη j ενώ αυτό στην πραγματικότητα ανήκει στην τάξη i. Το αναμενόμενο κόστος λανθασμένης ταξινόμισης ισούται με $\sum \sum L(i,j) p_i p_j$. Κατα συνέπεια ο γενικευμένος δείκτης Gini, στον οποίο λαμβάνονται υπόψιν οι απώλειες λόγω λάθους ταξινόμησης, δίνεται από τη σχέση

$$G(p) = \sum_i \sum_j L(i,j) p_i p_j$$

Το αντίστοιχο κριτήριο διαχωρισμού φαίνεται να είναι αποτελεσματικό σε εφαρμογές στις οποίες εμπλέκονται κόστη λανθασμένης ταξινόμησης μεταβλητών. Ωστόσο ένα μειονέκτημα του παραπάνω ορισμού είναι ότι δεν είναι απαραίτητα μια κοίλη συνάρτηση της αναλογίας p κάτι που είναι χρήσιμο στα μέτρα μη-καθαρότητας.

Τροποποιημένες εκ των προτέρων πιθανότητες Υπενθυμίζεται ότι το ρίσκο του κόμβου A ισούται με

$$\begin{aligned} R(A) &= \sum_{i=1}^{C} p_{iA} L(i, \tau(A)) \\ &= \sum_{i=1}^{C} \pi_i L(i, \tau(A)) (n_{iA}/n_i)(n/n_A) \end{aligned}$$

Έστω ότι υπάρχουν $\overline{\pi}$ και $\overline{L}$ ούτως ώστε

$$\overline{\pi}_i \overline{L}(i,j) = \pi_i L(i,j) \quad \forall i,j \in C$$

Τότε η ποσότητα $R(A)$ δε μεταβάλλεται από τις καινούριες απώλειες και τις καινούριες εκ των προτέρων. Εαν ο $\overline{L}$ είναι ανάλογος ως προς τον $0-1$ πίνακα απώλειας, τότε οι εκ των προτέρων $\overline{\pi}$ πρέπει να χρησιμοποιηθούν στα κριτήρια διαχωρισμού. Αυτό είναι δυνατό μόνο όταν ο L είναι της μορφής

$$L(i,j) = \begin{cases} L_i, & i \neq j \\ 0, & i = j \end{cases}$$

και

$$\overline{\pi}_i = \frac{\pi_i L_i}{\sum_j \pi_j L_j}$$

Αυτό είναι πάντα δυνατό όταν $C = 2$, και άρα οι τροποποιημένες εκ των προτέρων είναι ακριβείς για το πρόβλημα των δύο τάξεων. Για $C > 2$ το rpart χρησιμοποιεί τον παραπάνω τύπο όπου $L_i = \sum_j L(i,j)$.
Ένας ακόμη λόγος χρήσης των λεγόμενων altered priors είναι ο ακόλουθος. Ένας δείκτης μη-καθαρότητας $I(A) = \sum f(p_i)$ μεγιστοποιείται όταν $p_1 = p_2 = \ldots = p_C = 1/C$. Εάν για παράδειγμα σε ένα πρόβλημα η απώλεια από τη λανθασμένη ταξινόμηση στην τάξη 1 ήταν διπλάσια από τις λανθασμένες ταξινομήσεις στις τάξεις 2 ή 3, τότε θα θέλαμε η μη-καθαρότητα $I(A)$ να μεγιστοποιείται όταν $p_1 = 1/5, p_2 = p_3 = 2/5$, αφού αυτό είναι το χειρότερο δυνατό σετ αναλογιών με βάση το οποίο πρέπει ν'αποφασίσουμε ποιά είναι η τάξη του κόμβου.
Η τεχνική των τροποποιημένων εκ των προτέρων πιθανοτήτων κάνει ακριβώς αυτή τη δουλειά, μετατοπίζει την αναλογία p_i.

Οι altered priors επηρεάζουν μόνο την επιλογή του διαχωρισμού. Οι συνήθεις απώλειες και οι συνήθεις εκ των προτέρων πιθανότητες, χρησιμοποιούνται για τον υπολογισμό του ρίσκου του κόμβου. Οι τροποποιημένες εκ των προτέρων απλώς

βοηθούν τον κανόνα μη-καθαρότητας να επιλέξει διαχωρισμούς που πιθανώς είναι «καλοί» ως προς το ρίσκο.
Το επιχείρημα υπερ των τροποποιημένων εκ των προτέρων ευσταθεί τόσο για τον κανόνα διαχωρισμού του Gini όσο και για τον αντίστοιχο κανόνα του δείκτη πληροφορίας.

Κεφάλαιο 2

Δέντρα Ταξινόμησης

2.1 Εφαρμογή με δεδομένα του UCI

Το λογισμικό που χρησιμοποιείται είναι το πακέτο tree. Το σύνολο δεδομένων processed.cleveland.data[1] αποτελείται από 303 παρατηρήσεις, με την απόκριση να δίνει τη διάγνωση του ασθενούς. Οι τιμές 1, 2, 3 και 4 δηλώνουν την παρουσία καρδιακού νοσήματος ενώ η τιμή 0 δηλώνει την απουσία του. Δεν είναι διαθέσιμες συνολικά 6 τιμές (missing values). Οι ελλιπείς τιμές αντικαθίστανται με τιμές που προκύπτουν από τη γραμμική παρεμβολή κάθε στήλης ξεχωριστά με τη συνάρτηση na.approx() του πακέτου zoo της R.

- V_1 Ηλικία σε έτη (age)
- V_2 Φύλο (sex)
- V_3 Τύπος πόνου στο στήθος (1: τυπική στηθάγχη, 2: ατυπική στηθάγχη, 3: άλλος πόνος, 4: ασυμπτωματικός) (cp)
- V_4 Πίεση του ασθενούς κατά την εισαγωγή του στο νοσοκομείο (trestbps)
- V_5 Τιμή της χοληστερίνης (chol)
- V_6 Τιμή του σακχάρου στο αίμα (1: fasting blood sugar$>$120 mg/dl, 0: fasting blood sugar$\leq$120 mg/dl) (fbs)
- V_7 Ηλεκτροκαρδιογραφικά αποτελέσματα (0: κανονικά, 1: εμφάνιση ανωμαλιών, 2: πιθανή υπερτροφία αριστερής κοιλίας) (restecg)
- V_8 Μέγιστος καρδιακός ρυθμός (thalach)
- V_9 Στηθάγχη που προκλήθηκε λόγω άσκησης (1: ναι, 0: όχι) (exang)
- V_{10} Κατάσπαση του τμήματος ST στο ηλεκτροκαρδιογράφημα που προκλήθηκε κατά την άσκηση (oldpeak)
- V_{11} Κλίση του τμήματος ST κατά την άσκηση (1: κλίση προς τα επάνω, 2: επίπεδη, 3: κλίση προς τα κάτω) (slope)
- V_{12} Πλήθος βασικών αγγείων που χρωματίστηκαν κατά την ακτινοσκόπηση, λαμβάνει τιμές από 0 έως 3 (ca)
- V_{13} Αποτελέσματα σαρωτή θαλλίου (3: κανονικά, 6: μόνιμο πρόβλημα, 7: αναστρέψιμο πρόβλημα) (thal)

[1] Είναι διαθέσιμο στην ιστοσελίδα http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data του τμήματος Μηχανικής Μάθησης του πανεπιστημίου Irvine. Δημιουργήθηκε από τον Robert Detrano, M.D., Ph.D του ιδρύματος Cleveland Clinic Foundation και η δωρεά του στο τμήμα έγινε από τον David W. Aha.

Η κατασκευή του πλήρους δέντρου

```
heartmodel1<-tree(yfactor~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13, heartdata)
heartmodel1
plot(heartmodel1, y = NULL, type = "uniform");text(heartmodel1)

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 303 418.000 0 ( 0.54125 0.45875 )
   2) V13 < 4.5 167 176.600 0 ( 0.77844 0.22156 )
     4) V12 < 0.75 118  81.860 0 ( 0.88983 0.11017 )
       8) V1 < 57.5 83  25.810 0 ( 0.96386 0.03614 )
        16) V8 < 152.5 18  16.220 0 ( 0.83333 0.16667 )
          32) V5 < 226.5 9   0.000 0 ( 1.00000 0.00000 ) *
          33) V5 > 226.5 9  11.460 0 ( 0.66667 0.33333 ) *
        17) V8 > 152.5 65   0.000 0 ( 1.00000 0.00000 ) *
       9) V1 > 57.5 35  41.880 0 ( 0.71429 0.28571 )
        18) V6 < 0.5 29  37.360 0 ( 0.65517 0.34483 ) *
        19) V6 > 0.5 6   0.000 0 ( 1.00000 0.00000 ) *
     5) V12 > 0.75 49  67.910 0 ( 0.51020 0.48980 )
      10) V3 < 3.5 29  32.050 0 ( 0.75862 0.24138 )
        20) V1 < 65.5 22  27.520 0 ( 0.68182 0.31818 )
          40) V1 < 55.5 13   7.051 0 ( 0.92308 0.07692 ) *
          41) V1 > 55.5 9  11.460 1 ( 0.33333 0.66667 ) *
        21) V1 > 65.5 7   0.000 0 ( 1.00000 0.00000 ) *
      11) V3 > 3.5 20  16.910 1 ( 0.15000 0.85000 )
        22) V2 < 0.5 6   8.318 0 ( 0.50000 0.50000 ) *
        23) V2 > 0.5 14   0.000 1 ( 0.00000 1.00000 ) *
   3) V13 > 4.5 136 153.000 1 ( 0.25000 0.75000 )
     6) V3 < 3.5 45  62.180 0 ( 0.53333 0.46667 )
      12) V8 < 143 12  10.810 1 ( 0.16667 0.83333 ) *
      13) V8 > 143 33  42.010 0 ( 0.66667 0.33333 )
        26) V5 < 205.5 6   0.000 0 ( 1.00000 0.00000 ) *
        27) V5 > 205.5 27  36.500 0 ( 0.59259 0.40741 ) *
     7) V3 > 3.5 91  63.020 1 ( 0.10989 0.89011 )
      14) V10 < 0.55 22  28.840 1 ( 0.36364 0.63636 )
        28) V5 < 237.5 12  16.300 0 ( 0.58333 0.41667 ) *
        29) V5 > 237.5 10   6.502 1 ( 0.10000 0.90000 ) *
      15) V10 > 0.55 69  18.110 1 ( 0.02899 0.97101 )
        30) V13 < 6.5 12  10.810 1 ( 0.16667 0.83333 ) *
        31) V13 > 6.5 57   0.000 1 ( 0.00000 1.00000 ) *
```

Παρατηρούμε ότι προέκυψε ένα δέντρο που αποτελείται από **33 κόμβους**, εκ των οποίων οι 17 είναι **τελικοί** (φέρουν αστερίσκο) και από το *Σχήμα (2.1)* είναι εμφανείς οι 16 διαχωρισμοί/**splits**. Οι κόμβοι που προκύπτουν από κάποιον κόμβο x συμβολίζονται με τους αριθμούς $2x$ και $2x+1$ και στο προηγούμενο αποτέλεσμα εμφανίζονται στην αρχή κάθε γραμμής. Ακολουθεί ο ορισμός του διαχωρισμού βάση του οποίου προέκυψε ο κόμβος, το πλήθος των ατόμων του κόμβου, η απόκλισή του **(deviance)**, η προβλεπόμενη τάξη του κόμβου, και στο τέλος το διάνυσμα των πιθανοτήτων για κάθε τάξη (τάξη 0, τάξη 1). Έτσι στον υπ'αριθμόν 2 κόμβο για παράδειγμα, τα άτομα για τα οποία ισχύει $V_{13} < 4.5$ έχουν πιθανότητα 0.77844 να είναι υγιή σύμφωνα με το συγκεκριμένο δέντρο/μοντέλο.

Η **απόκλιση** είναι το μέτρο της ετερογένειας του κόμβου που εφαρμόζεται από τον

αλγόριθμο που κατασκευάζει το δέντρο. Ένα δέντρο επεκτείνεται με τον αλγόριθμο διαμέρισης να μοιράζει αναδρομικά τα δεδομένα σε κάθε κόμβο μέχρι να επιτευχθεί ομοιογένεια σε κάθε έναν, ή μέχρι να περιέχει πολύ λίγες παρατηρήσεις, ≤ 5. Ένας κόμβος με τέλεια ομοιογένεια πρέπει να έχει μηδενική απόκλιση.

Ακολουθεί μια ερμηνεία ορισμένων αποτελεσμάτων του πίνακα,

- Ο διαχωρισμός στη μεταβλητή που δίνει τα αποτελέσματα του σαρωτή θαλλίου (V_{13}) διαμερίζει τα 303 άτομα σε ομάδες των 167 και 136 (κόμβοι υπ'αριθμόν 2 και 3) με πιθανότητα ύπαρξης καρδιακού νοσήματος να ισούται με 0.22156 και 0.75 αντίστοιχα.

- Κατόπιν η ομάδα ατόμων του κόμβου 2 διαμερίζεται σε ομάδες των 118 και 49 (κόμβοι 4 και 5) ανάλογα με το πλήθος βασικών αγγείων που χρωματίστηκαν κατά την ακτινοσκόπηση (V_{12}) ήταν κανένα ή ένα και πάνω.

- Η ομάδα του κόμβου 4 διαρείται σε ομάδες των 83 και 35 ανάλογα με το εαν η ηλικία (V_1) είναι μικρότερη ή μεγαλύτερη από τα 57.5 έτη.

- Η διαδικασία συνεχίζεται με αποτέλεσμα να έχουμε 17 ξεχωριστές ομάδες με την πιθανότητα για εκδήλωση της νόσου να κυμαίνεται από 0 έως 1 ανάμεσά τους.

```
summary(heartmodel1)

Classification tree:
tree(formula = yfactor ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +
    V9 + V10 + V11 + V12 + V13, data = heartdata)
Variables actually used in tree construction:
[1] "V13" "V12" "V1"  "V8"  "V5"  "V6"  "V3"  "V2"  "V10"
Number of terminal nodes:  17
Residual mean deviance:  0.5475 = 156.6 / 286
Misclassification error rate: 0.1353 = 41 / 303
```

Βλέπουμε ότι χρησιμοποιήθηκαν μόνο οι 9 εκ των 13 μεταβλητών για την κατασκευή του δέντρου. Έμειναν δηλαδή εκτός οι V_4, V_7, V_9 και V_{11} ενώ ο **ρυθμός λανθασμένης ταξινόμησης** για το πλήρες μοντέλο ισούται με 0.1353.

Σχετικά με την ποσότητα **Residual mean deviance** για το πρόβλημα της ταξινόμησης ισχύουν τα εξής,

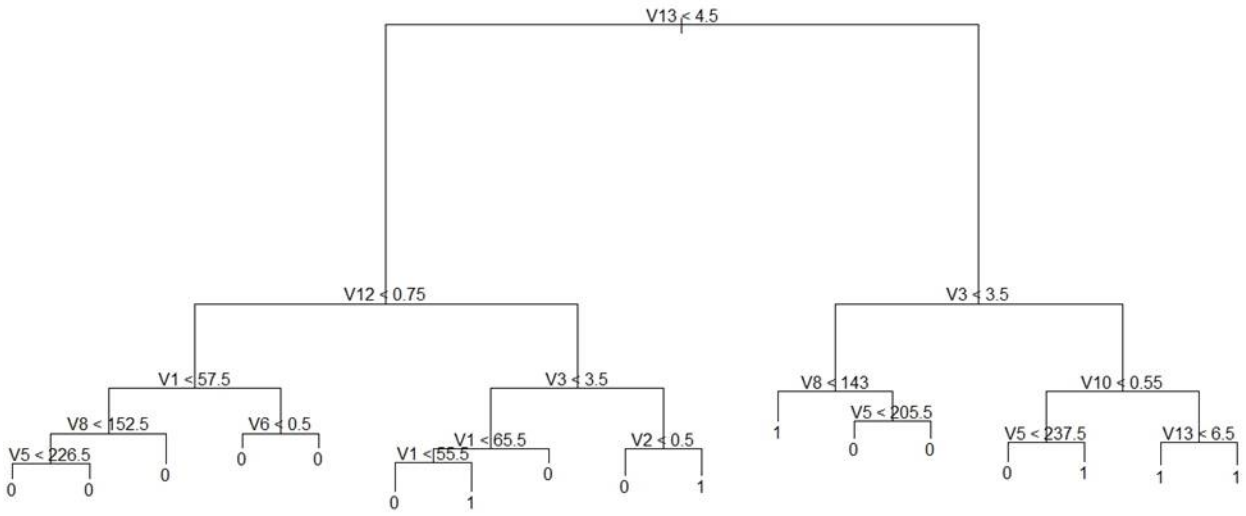
Για κάθε κόμβο i ενός δέντρου ταξινόμησης, έχουμε μια κατανομή πιθανότητας p_{ik} για τις τάξεις του προβλήματος. Τα φύλλα ή οι τελικοί κόμβοι του δέντρου δίνουν ένα τυχαίο δείγμα n_{ik} από μια **Πολυωνυμική Κατανομή** που καθορίζεται από τα p_{ik}. Επομένως, η απόκλιση D ενός δέντρου ορίζεται ως το άθροισμα των αποκλίσεων όλων των φύλλων [16],

$$D_i = -2 \sum_k n_{ik} \log(p_{ik})$$

Η βασική ιδέα είναι η ελαχιστοποίηση, μέσω της διαδικασίας του κλαδέματος, της ποσότητας $(D + a \times (\#nodes))$ όπου a είναι η παράμετρος **cost-complexity**. Η συνολική απόκλιση D ταυτίζεται με την έννοια της Μη καθαρότητας του κόμβου,[2]

[2]Η ετερογένεια της ομάδας ενός κόμβου.

δηλαδή του **Node Impurity** το οποίο βασίζεται στο γνωστό δείκτη Gini που ορίζεται ως $1 - \sum_k {p_{ik}}^2$.



Σχήμα 2.1: Το πλήρες δέντρο ταξινόμησης CART των δεδομένων cleveland.processed.

Τέλος, στην εφαρμογή έχουμε μόνο primary splits διότι δεν έχουμε ελλιπή δεδομένα (missing data). Στην περίπτωση των ελλιπών δεδομένων το κριτήριο διαχωρισμού παραμένει η μεγιστοποίηση της ποσότητας

$$\Delta I = p(A)I(A) - p(A_L)I(A_L) - p(A_R)I(A_R)$$

ωστόσο οι όροι $p(A_L)I(A_L)$ και $p(A_R)I(A_R)$ τροποποιούνται. Οι δείκτες μη-καθαρότητας $I(A_R)$ και $I(A_L)$ υπολογίζονται από τις παρατηρήσεις οι οποίες δεν έχουν ελλιπείς τιμές για καμία μεταβλητή. Ομοίως οι πιθανότητες $p(A_L)$ και $p(A_R)$ υπολογίζονται από τις πλήρεις παρατηρήσεις, αλλά στη συνέχεια τροποποιούνται ώστε ν'αθροίζουν στην πιθανότητα $p(A)$.

Διαδικασία pruning «κλαδέματος»

Στην εισαγωγή του παρόντος κεφαλαίου αναφέρθηκε ότι το μειονέκτημα του πλήρους δέντρου ειναι ότι παρουσιάζει overfitting ενώ και η μορφή του είναι ιδιαίτερα πολύπλοκη. Συνεπώς η προσπάθεια επικεντρώνεται στην απλοποίηση του πλήρους δέντρου ή διαφορετικά στο «κλάδεμα» του. Αυτή η διαδικασία παράγει μέσω του πλήρους δέντρου, μια σειρά από λιγότερο πολύπλοκα δέντρα [5].

Στόχος του κλαδέματος είναι η εύρεση του δέντρου με την σωστή διάσταση, το οποίο είναι προφανώς υποδέντρο του πλήρους δέντρου. Το ενδιαφέρον επικεντρώνεται στο κριτήριο που θα χρησιμοποιηθεί για τον καθορισμό της βέλτιστης διάστασης.

Έστω $T_1, T_2, \ldots, T_k$ οι τελικοί κόμβοι ενός δέντρου Τ. Υπενθυμίζεται ότι,

$$|T| = \text{πλήθος τελικών κόμβων}$$

και το ρίσκο του δέντρου Τ ως

$$R(T) = \sum_{i=1}^{k} P(T_i)R(T_i)$$

Έστω a κάποιος θετικός πραγματικός αριθμός, ο οποίος μετρά το κόστος που συνεπάγεται η προσθήκη μιας ακόμη μεταβλητής στο μοντέλο. Ο αριθμός a καλείται παράμετρος πολυπλοκότητας (Complexity Parameter). Με $R(T_0)$ συμβολίζεται το ρίσκο του δέντρου χωρίς διαχωρισμούς (splits). Υπενθυμίζεται ακόμη ότι

$$R_a(T) = R(T) + a|T|$$

είναι το κόστος του δέντρου και συμβολίζουμε με T_a το «παρακλάδι» του πλήρους μοντέλου με το ελάχιστο κόστος. Προφανώς το πλήρες μοντέλο/δέντρο συμβολίζεται ως T_∞ ενώ εκείνο χωρίς κανένα split με T_0.

Τότε αποδεικνύονται[3] τα εξής:

[3] *Classification and Regression Trees* των L. Breiman, J.H. Friedman, R.A. Olshen και C.J Stone, Εκδόσεις Wadsworth, Belmont CA, 1983.

1. Εάν T_1 και T_2 είναι υποδέντρα του T με $R_a(T_1) = R_a(T_2)$, τότε είτε το T_1 είναι υποδέντρο του T_2, είτε το T_2 είναι υποδέντρο του T_1. Δηλαδή ισχύει $|T_1| < |T_2|$ ή $|T_1| > |T_2|$.

2. Εάν $a > \beta$ τότε είτε $T_a = T_\beta$ είτε το T_a είναι αυστηρά υποδέντρο του T_β.

3. Με δεδομένους τους αριθμούς $a_1, a_2, \ldots, a_m$ οι ποσότητες $T_{a_1}, \ldots, T_{a_m}$ και $R(T_{a_1}), \ldots, R(T_{a_m})$ μπορούν να υπολογιστούν.

Με βάση το πρώτο αποτέλεσμα έχουμε ότι η ποσότητα $R_a(T)$ ελαχιστοποιείται για $T = T_a$. Επειδή οποιαδήποτε ακολουθία εμφωλευμένων δέντρων του T έχει το πολύ $|T|$ τελικούς κόμβους, με βάση το δεύτερο αποτέλεσμα όλες οι δυνατές τιμές του a μπορούν να ταξινομηθούν σε m διαστήματα ($m \leq |T|$).

$$\begin{aligned} I_1 &= [0, a_1] \\ I_2 &= (a_1, a_2] \\ &\vdots \\ I_m &= (a_{m-1}, \infty] \end{aligned}$$

όπου όλες οι τιμές $a \in I_i$ αφορούν στο ίδιο δέντρο T_a.

Επιλογή βέλτιστου δέντρου

Για την επιλογή της βέλτιστης παραμέτρου πολυπλοκότητας a και επομένως του βέλτιστου δέντρου, χρησιμοποιείται η μέθοδος Cross-validation [15] με τα ακόλουθα βήματα:

1. Προσαρμόζουμε το πλήρες μοντέλο στα δεδομένα και υπολογίζουμε τα διαστήματα $I_1, I_2, \ldots, I_m$.
   Θέτουμε

$$\begin{aligned} \beta_1 &= 0 \\ \beta_2 &= \sqrt{a_1 a_2} \\ \beta_3 &= \sqrt{a_2 a_3} \\ &\vdots \\ \beta_{m-1} &= \sqrt{a_{m-2} a_{m-1}} \\ \beta_m &= \infty \end{aligned}$$

2. Διαμερίζουμε τα δεδομένα σε s σύνολα $G_1, G_2, \ldots, G_s$ μεγέθους s/n το καθένα και για κάθε σύνολο ξεχωριστά[4]:

[4] Έχει βρεθεί ότι η τιμή $s = 10$ επαρκεί, ωστόσο μπορεί να χρησιμοποιηθεί και άλλη τιμή.

- Προσαρμόζουμε το πλήρες μοντέλο στα δεδομένα εξαιρώντας κάθε φορά ένα σύνολο G_i και καθορίζουμε τις τιμές $T_{\beta_1}, T_{\beta_2}, \ldots, T_{\beta_m}$ του μικρότερου (κατά G_i) συνόλου δεδομένων.
- Υπολογίζουμε την προβλεπόμενη τάξη για κάθε παρατήρηση του G_i σύμφωνα με καθένα από τα μοντέλα T_{β_j} για $1 \leq j \leq m$.
- Από το προηγούμενο βήμα υπολογίζουμε το ρίσκο για κάθε άτομο.

3. Αθροίζουμε μέσα στο G_i για να πάρουμε το εκτιμώμενο ρίσκο για κάθε β_j. Για εκείνη την παράμετρο πολυπλοκότητας β με το μικρότερο ρίσκο υπολογίζουμε το T_β για το πλήρες σετ δεδομένων. Το T_β είναι το βέλτιστο δέντρο που τελικά επιλέγουμε.

Επανερχόμαστε στο σετ δεδομένων cleveland.processed.data προκειμένου να εφαρμόσουμε τη διαδικασία **Pruning**. Μέσω της συνάρτησης prune.tree αξιολογείται ο βαθμός απλοποίησης του πλήρους δέντρου χωρίς να γίνει καμία έκπτωση στην καλή προσαρμογή του μοντέλου στα δεδομένα.

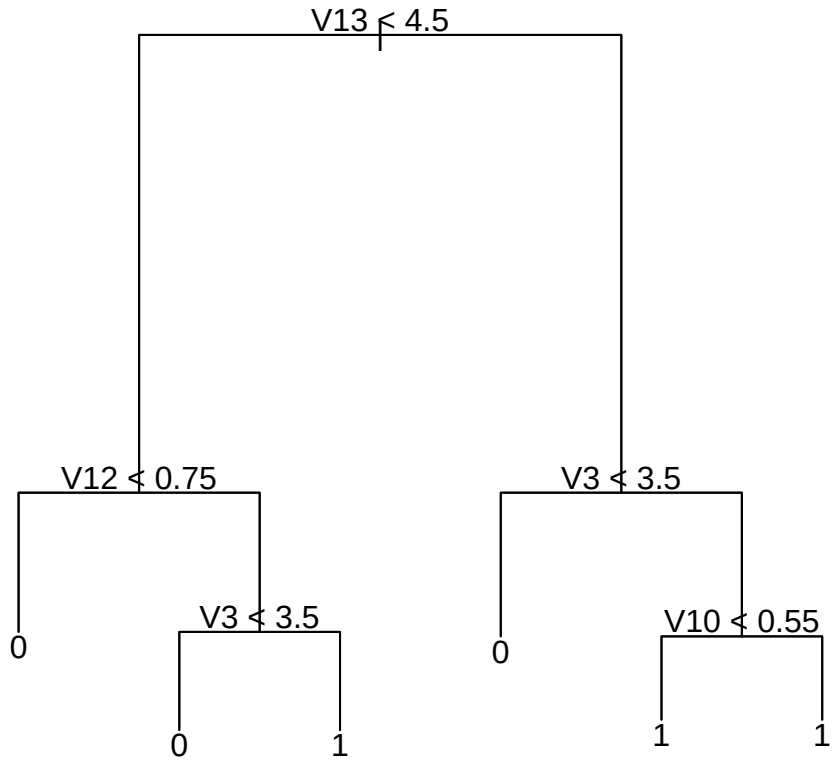
```
prune.tree(heartmodel1, k = 14.170738, method = "deviance")
```

Το όρισμα k είναι η Cost Complexity παράμετρος η οποία υπολογίζεται με τη βοήθεια της 10-απλής Cross Validation, ως μέθοδος μπορεί να επιλεγεί είτε η απόκλιση είτε η misclass δηλαδή ο αριθμός των λανθασμένων ταξινομήσεων ή συνολική απώλεια. Το όρισμα loss είναι η 0-1 απώλεια.

```
cv.tree(heartmodel1, FUN = prune.tree, K = 10)

$size
17 16 15 14 13 11 10  9  8  7  6  5  4  3  2  1

$dev

c(578.2334, 544.6557, 527.6433, 478.6138, 464.9161, 417.3316, 418.5508, 377.715,

374.1351, 370.4903, 355.678, 356.4324, 339.1545, 343.0823, 343.287, 423.6683)

$k

c(4.516096, 4.762948, 5.511279, 6.0389, 6.773185, 7.291829, 8.590597, 9.359694,

9.591402, 14.170738, 16.077469, 18.945102, 26.873822, 27.748096, 88.383928)
```

Πραγματοποιήθηκε 10-απλή **Cross Validation** και υπολογίστηκαν οι αποκλίσεις (deviance) σε συνάρτηση με τις τιμές των διαφόρων παραμέτρων cost-complexity k που δίνονται στα διανύσματα. Το μέγεθος των δέντρων αναφέρεται στο πλήθος των φύλλων/τελικών κόμβων.

Μια «μικρή» απόκλιση ισούται με 355.678 για $k = 14.170738$ άρα δοκιμάζουμε την τελευταία στο κλάδεμα του αρχικού πλήρους μοντέλου.



Σχήμα 2.2: Το κλαδεμένο δέντρο έχει μόνο 6 τελικούς κόμβους.

```
> prune.tree(heartmodel1, k = 14.170738, method = "deviance")
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 303 418.00 0 ( 0.54125 0.45875 )
   2) V13 < 4.5 167 176.60 0 ( 0.77844 0.22156 )
     4) V12 < 0.75 118  81.86 0 ( 0.88983 0.11017 ) *
     5) V12 > 0.75 49  67.91 0 ( 0.51020 0.48980 )
      10) V3 < 3.5 29  32.05 0 ( 0.75862 0.24138 ) *
      11) V3 > 3.5 20  16.91 1 ( 0.15000 0.85000 ) *
   3) V13 > 4.5 136 153.00 1 ( 0.25000 0.75000 )
     6) V3 < 3.5 45  62.18 0 ( 0.53333 0.46667 ) *
     7) V3 > 3.5 91  63.02 1 ( 0.10989 0.89011 )
      14) V10 < 0.55 22  28.84 1 ( 0.36364 0.63636 ) *
      15) V10 > 0.55 69  18.11 1 ( 0.02899 0.97101 ) *

g<-prune.tree(heartmodel1, k = 14.170738, method = "deviance")
plot(g, y = NULL, type = "proportional");text(g)
```

```
summary(heartmodel2)

Classification tree:
snip.tree(tree = heartmodel1, nodes = c(14L, 10L, 15L, 11L, 6L,
4L))
Variables actually used in tree construction:
[1] "V13" "V12" "V3"  "V10"
Number of terminal nodes:  6
Residual mean deviance:  0.8079 = 240 / 297
Misclassification error rate: 0.1782 = 54 / 303
```

Ο ρυθμός λανθασμένης ταξινόμησης αυξήθηκε ελάχιστα εξ'αιτίας του κλαδέματος και ισούται με 0.1782. Την ίδια πορεία είχε και η συνολική απόκλιση D η οποία ήταν ίση με 0.5475 και αυξήθηκε στο 0.8079.

Κεφάλαιο 3

Δέντρα Παλινδρόμησης

3.1 Ορισμός

Στα δέντρα ταξινόμησης ένα από τα κριτήρια που χρησιμοποιούνται προκειμένου ν'αποφασιστεί ποιά μεταβλητή δίνει τον καλύτερο διαχωρισμό είναι ο δείκτης Gini. Στα δέντρα παλινδρόμησης (anova method) το κριτήριο διαχωρισμού είναι το

$$SS_T - (SS_L + SS_R)$$

όπου $SS_T = \sum(y_i - \overline{y})^2$, δηλαδή το άθροισμα των τετραγώνων του κόμβου. Ομοίως, οι ποσότητες SS_L και SS_R είναι τα αθροίσματα τετραγώνων για τον αριστερό και δεξί θυγατρικό κόμβο αντίστοιχα. Δηλαδή το κριτήριο επιλογής του διαχωρισμού είναι η μεγιστοποίηση του αθροίσματος τετραγώνων μεταξύ των ομάδων στην απλή anova. Η απόκριση είναι η μέση τιμή του κόμβου ενώ το σφάλμα του ισούται με τη διασπορά της y. Τέλος, το σφάλμα της πρόβλεψης για την επόμενη παρατήρηση ισούται με $y_{new} - \overline{y}$.

Προέλευση των δεδομένων

Τα δεδομένα treedata.csv[1]αφορούν στο πρόβλημα του προσδιορισμού της αφθονίας διαφόρων ειδών δέντρων και προέρχονται από τον προστατευόμενο εθνικό δρυμό Great Smoky Mountains των Η.Π.Α..

Η μεταβλητή απόκρισης είναι η cover η οποία μετρά την αφθονία μετρημένη ως η οριζόντια κάλυψη, δηλαδή η έκταση της σκιάς που δημιουργείται κάτω από το ηλιακό φως. Οι τιμές της είναι διακριτές και κυμαίνονται από το 1 έως το 10. Η τιμή 1 αντιστοιχεί σε ίχνος σκιάς, η τιμή 2 σε 0-1% σκίασης, η τιμή 3 σε 1-2%, η τιμή 4 σε 2-5%, η τιμή 5 σε 5-10%, η τιμή 6 σε 10-25%, η τιμή 7 σε 25-50%, η τιμή 8 σε 50-75%, η τιμή 9 σε 75-95% και τέλος η τιμή 10 σε 95-100%.

Προκειμένου οι περιβαλλοντολόγοι να μετρήσουν τα μεγέθη φυτικών πληθυσμών σε μια περιοχή ενδιαφέροντος, επιλέγουν και μελετούν εκτενώς κάποια plots τα οποία είναι δείγματα πολύ μικρής έκτασης. Επομένως χρησιμοποιούν τις πληροφορίες των plots ούτως ώστε να γενικεύσουν τα αποτελέσματα για την ευρύτερη περιοχή ενδιαφέροντος.

Το σετ δεδομένων περιλαμβάνει τις μετρήσεις για 8971 plots σε 12 ανεξάρτητες μεταβλητές, οι οποίες είναι οι plotID (μοναδικός κωδικός για κάθε χωρική μονάδα),

[1]Διατίθενται στην ιστοσελίδα http://plantecology.syr.edu/fridley/bio793/cart.html

plotsize (μέγεθος σε τ.μ.), date (ημερομηνία καταγραφής του είδους), spcode (μοναδικός επταψήφιος κωδικός για κάθε είδος), species (το πλήρες όνομα του είδους), utme (γεωγραφική συντεταγμένη x), utmn (γεωγραφική συντεταγμένη y), elev (υψόμετρο σε μέτρα), tci (διαφορετικά water potential), streamdist (απόσταση σε μέτρα του plot από το πλησιέστερο ρυάκι), disturb (η επιβάρρυνση του plot μεταβλητή με 4 δυνατές τιμές - από το παρθένο έδαφος VIRGIN, την εγκατάσταση SETTLE, την ήπια εκμετάλλευση για ξυλεία LTSEL, έως την εμπορική του εκμετάλλευσή του για ξυλεία CORPLOG) και τέλος η beers (διαφορετικά heat load index με συνεχείς τιμές από το 0 (θερμότερο) έως το 2 (ψυχρότερο)).

3.2 Εφαρμογή με δεδομένα δέντρων

Πακέτο rpart Εφαρμόζουμε το μοντέλο με ανεξάρτητες μεταβλητές τις utme, utmn, elev, tci, streamdist και beers.

Συγκεκριμένα χρησιμοποιηθηκαν οι εντολές,

```
fit <- rpart(cover~utme+utmn+elev+tci+streamdist+beers,
method="anova", data=rt)
fit
printcp(fit)
plotcp(fit)
summary(fit)
plot(fit, uniform=TRUE,main="Regression Tree for cover")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```

Ως μέθοδος επιλέχθηκε η anova αφού πρόκειται για μοντέλο παλινδρόμησης.

```
n= 8971

node), split, n, deviance, yval
      * denotes terminal node

1) root 8971 39735.670 4.006019
  2) elev< 1157.5 6875 29605.550 3.867636 *
  3) elev>=1157.5 2096  9566.634 4.459924 *
```

Η ανάλυση έδωσε ένα δέντρο με έναν διαχωρισμό και τρεις συνολικά κόμβους, εκ των οποίων οι δύο είναι τελικοί.

```
Regression tree:
rpart(formula = cover ~ utme + utmn + elev + tci +
streamdist + beers, data = rt, method = "anova")

Variables actually used in tree construction:
[1] elev

Root node error: 39736/8971 = 4.4293

n= 8971

        CP nsplit rel error  xerror     xstd
1 0.014181      0   1.00000 1.00023 0.011850
2 0.010000      1   0.98582 0.98657 0.011815
```

Το σχετικό σφάλμα **rel error** ισούται με $1 - R^2$, όπου R^2 είναι ο συντελεστής προσδιορισμού, και το **error** σχετίζεται με το στατιστικό PRESS. Παρατηρούμε ότι το μοναδικό split που γίνεται δε βελτιώνει σημαντικά την προσαρμογή του μοντέλου και ελαττώνει ελάχιστα το σφάλμα της cross validation. Συνεπώς εφαρμόζουμε ξανά το μοντέλο μεταβάλλοντας το κατώφλι για την τιμή cp από 0.01 που είναι η default τιμή σε 0.002 και τότε έχουμε τα ακόλουθα αποτελέσματα.

```
fit1<-rpart(cover~utme+utmn+elev+tci+streamdist+beers,data=rt,
method="anova", control = rpart.control(cp = 0.002))
fit1
printcp(fit1)
plotcp(fit1)
summary(fit1)


n= 8971

node), split, n, deviance, yval
      * denotes terminal node

  1) root 8971 39735.67000 4.006019
    2) elev< 1157.5 6875 29605.55000 3.867636
      4) elev< 642.8 2946 11359.08000 3.704684
        8) utme>=228670 2901 10766.12000 3.675284
         16) utme< 252110.5 2581  9068.76600 3.608679
           32) utme>=242914.5 1087  3425.27700 3.311868
             64) elev< 615.9 909  2813.19700 3.238724 *
             65) elev>=615.9 178   582.38200 3.685393
              130) elev>=624.45 157   403.42680 3.343949 *
              131) elev< 624.45 21    23.80952 6.238095 *
           33) utme< 242914.5 1494  5478.05400 3.824632 *
         17) utme>=252110.5 320  1593.55000 4.212500 *
        9) utme< 228670 45   428.80000 5.600000 *
      5) elev>=642.8 3929 18109.59000 3.989819
       10) utme>=291254 428  2016.96300 3.490654
         20) streamdist>=174.45 67   244.65670 2.462687 *
         21) streamdist< 174.45 361  1688.36600 3.681440 *
       11) utme< 291254 3501 15972.95000 4.050843
         22) utme< 238424 355  1076.69300 3.512676 *
         23) utme>=238424 3146 14781.84000 4.111570
           46) streamdist>=40.615 2683 12420.27000 4.037272 *
           47) streamdist< 40.615 463  2260.92900 4.542117 *
    3) elev>=1157.5 2096  9566.63400 4.459924
      6) utmn< 3944782 1830  8132.94000 4.384153 *
      7) utmn>=3944782 266  1350.90600 4.981203
       14) tci< 3.722733 17    54.47059 2.823529 *
       15) tci>=3.722733 249  1211.88800 5.128514 *
```

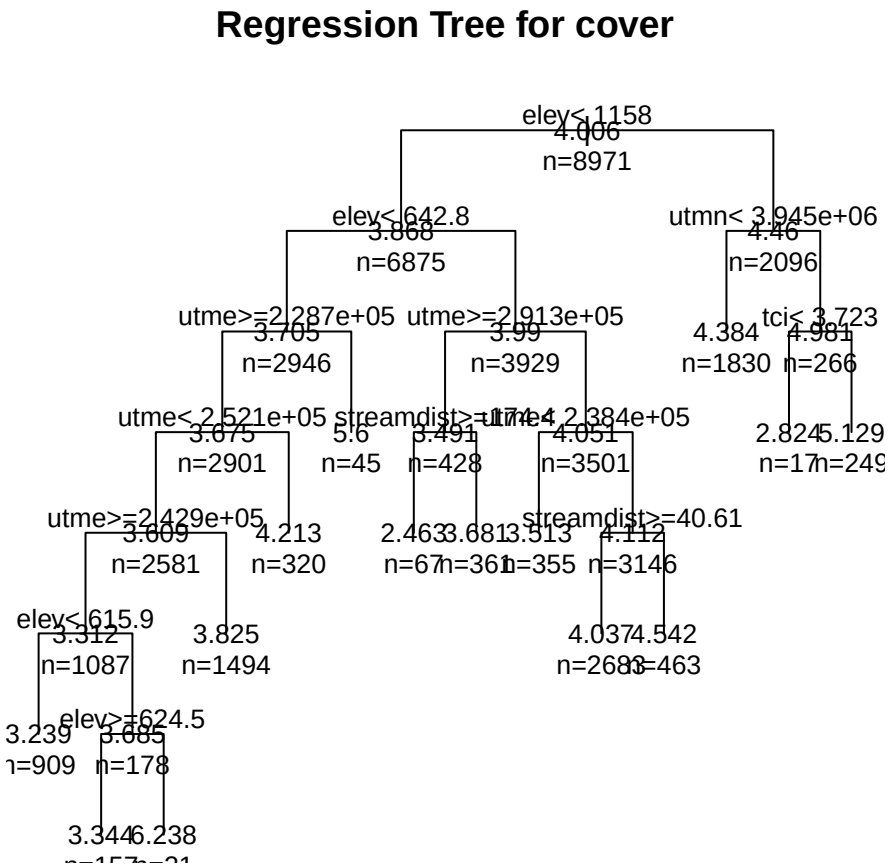
Κατ'αναλογία με τα δέντρα ταξινόμησης, σε κάθε γραμμή δίνεται ο αριθμός του εκάστοτε κόμβου, ο διαχωρισμός, το πλήθος των παρατηρήσεων ανά κόμβο, το διορθωμένο άθροισμα τετραγώνων / διασπορά της y (deviance) και η μέση τιμή των παρατηρήσεων εντός του κόμβου. Με αστερίσκο σημειώνονται οι τελικοί κόμβοι.

```
Regression tree:
```

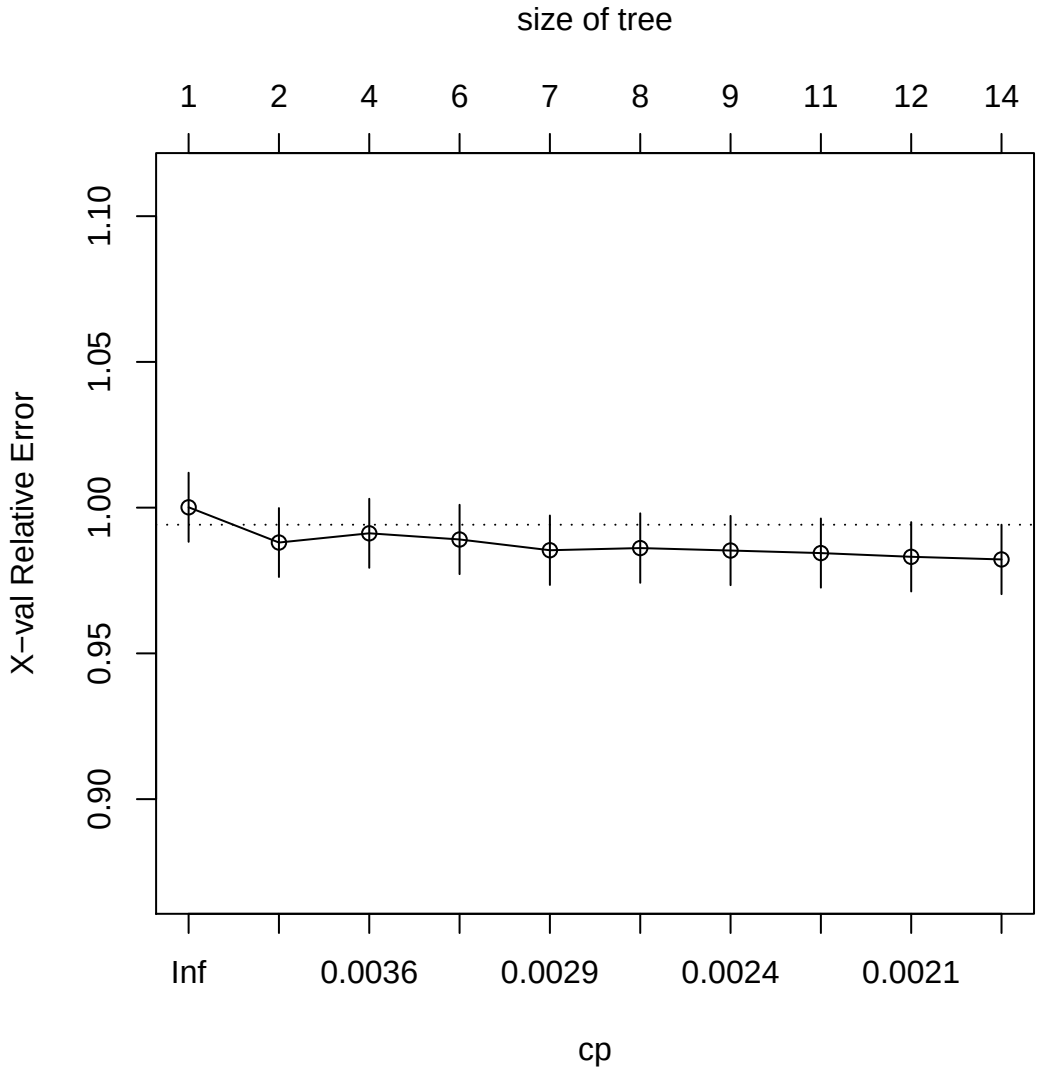
```
rpart(formula = cover ~ utme + utmn + elev + tci + streamdist +
beers,data=rt,method="anova",control=rpart.control(cp = 0.002))

Variables actually used in tree construction:
[1] elev       streamdist tci        utme       utmn

Root node error: 39736/8971 = 4.4293

n= 8971

          CP nsplit rel error  xerror     xstd
1  0.0141810      0   1.00000 1.00013 0.011849
2  0.0037880      1   0.98582 0.98804 0.011821
3  0.0033879      3   0.97824 0.99120 0.011839
4  0.0030119      5   0.97147 0.98909 0.011895
5  0.0028795      6   0.96846 0.98539 0.011910
6  0.0025327      7   0.96558 0.98614 0.011914
7  0.0023259      8   0.96304 0.98529 0.011882
8  0.0021125     10   0.95839 0.98442 0.011859
9  0.0021056     11   0.95628 0.98314 0.011894
10 0.0020000     13   0.95207 0.98224 0.011912
```

Το δέντρο παλινδρόμησης αποτελείται από 13 διαχωρισμούς, 14 τελικούς κόμβους ενώ από τις 6 ανεξάρτητες μεταβλητές τελικά χρησιμοποιήθηκαν στην ανάλυση οι 5 (elev, streamdist, tci, utme και utmn). Το πρώτο split φαίνεται να βελτιώνει περισσότερο την προσαρμογή του μοντέλου, αφού τότε σημειώνεται η «δραστικότερη» μείωση του rel error. Το τελευταίο split προσφέρει ελάχιστα στην καλύτερη προσαρμογή του μοντέλου και μάλιστα αυξάνει ελαφρά το σφάλμα της cross validation.

Για κάθε τιμή της CP μεταξύ 0.0038 και 0.0142, το βέλτιστο μοντέλο αποτελείται από έναν διαχωρισμό, για κάθε τιμή της CP μεταξύ 0.0034 και 0.0038 το βέλτιστο μοντέλο αποτελείται από 3 διαχωρισμούς κ.ο.κ.



Σχήμα 3.1: Το δέντρο παλινδρόμησης με τις 5 εξαρτημένες μεταβλητές. Σε κάθε κόμβο αναγράφεται η μέση τιμή των παρατηρήσεων του.



Σχήμα 3.2: Το δέντρο του σχήματος 3.1 είναι αρκετά πολύπλοκο. Επομένως προτείνεται να υποστεί pruning ούτως ώστε να περιλαμβάνει μόνο 7 τελικούς κόμβους αντί για 14, αφού οι πλεονάζοντες δεν συνεισφέρουν στην ελάττωση του rel error, δηλαδή στην άυξηση του R^2.

```
Node number 1: 8971 observations,    complexity param=0.01418102
  mean=4.006019, MSE=4.429347
  left son=2 (6875 obs) right son=3 (2096 obs)
  Primary splits:
   elev  < 1157.5 to the left,  improve=0.014181020, (0 missing)
   utme  < 260378 to the left,  improve=0.006210418, (0 missing)
   utmn  < 3941705 to the right, improve=0.004254300, (0 missing)
   beers < 0.2082226 to the left,  improve=0.003965862, (0 missing)
   tci   < 4.518608 to the right, improve=0.001270782, (0 missing)
  Surrogate splits:
   streamdist < 669.6 to the left,  agree=0.770, adj=0.017, (0 split)
   beers      < 1.99992 to the left,  agree=0.769, adj=0.010, (0 split)
   utme       < 311400.5 to the left,  agree=0.767, adj=0.002, (0 split)
```

```
   tci        < 3.053798 to the right, agree=0.767, adj=0.001, (0 split)

Node number 2: 6875 observations,    complexity param=0.003788012
  mean=3.867636, MSE=4.306262
  left son=4 (2946 obs) right son=5 (3929 obs)
  Primary splits:
 elev       < 642.8 to the left,  improve=0.004623490, (0 missing)
 utme       < 228670 to the right, improve=0.004591658, (0 missing)
 beers      < 0.2093909 to the left,  improve=0.004570857, (0 missing)
 utmn       < 3941778 to the right, improve=0.003033020, (0 missing)
 streamdist < 60.415 to the right, improve=0.001457844, (0 missing)
 Surrogate splits:
 utmn       < 3941842 to the right, agree=0.745, adj=0.405, (0 split)
 utme       < 239398.5 to the left,  agree=0.675, adj=0.242, (0 split)
 beers      < 0.1364979 to the left,  agree=0.612, adj=0.094, (0 split)
 streamdist < 531.7 to the right, agree=0.600, adj=0.066, (0 split)
 tci        < 7.504867 to the right, agree=0.592, adj=0.049, (0 split)

Node number 3: 2096 observations,    complexity param=0.002105602
  mean=4.459924, MSE=4.564234
  left son=6 (1830 obs) right son=7 (266 obs)
  Primary splits:
 utmn       < 3944782 to the left,  improve=0.008653737, (0 missing)
 elev       < 1364.5 to the left,  improve=0.008419121, (0 missing)
 tci        < 5.596267 to the left,  improve=0.006991079, (0 missing)
 utme       < 305218 to the right, improve=0.006725772, (0 missing)
 streamdist < 610.6 to the left,  improve=0.003570608, (14 missing)
  Surrogate splits:
      elev < 1758 to the left,  agree=0.914, adj=0.323, (0 split)
      tci  < 3.010473 to the right, agree=0.875, adj=0.019, (0 split)

Node number 4: 2946 observations,    complexity param=0.003788012
  mean=3.704684, MSE=3.855762
  left son=8 (2901 obs) right son=9 (45 obs)
  Primary splits:
      utme  < 228670 to the right, improve=0.01445166, (0 missing)
      beers < 0.2393186 to the left,  improve=0.01108149, (0 missing)
      elev  < 313.95 to the right, improve=0.01055597, (0 missing)
      utmn  < 3940278 to the right, improve=0.00822455, (0 missing)
      tci   < 6.275865 to the right, improve=0.00386489, (0 missing)
  Surrogate splits:
elev       < 313.95 to the right, agree=0.994, adj=0.622, (0 split)
streamdist < 851.05 to the left,  agree=0.988, adj=0.244, (0 split)

Node number 5: 3929 observations,    complexity param=0.003011907
  mean=3.989819, MSE=4.609212
  left son=10 (428 obs) right son=11 (3501 obs)
  Primary splits:
   utme       < 291254 to the right, improve=0.006608660, (0 missing)
   streamdist < 40.615 to the right, improve=0.006241007, (0 missing)
   tci        < 6.713293 to the left,  improve=0.003760180, (0 missing)
   utmn       < 3938251 to the left,  improve=0.001435233, (0 missing)
   beers      < 0.7289352 to the left,  improve=0.001364077, (0 missing)
  Surrogate splits:
      utmn < 3956502 to the right, agree=0.897, adj=0.058, (0 split)
      tci  < 19.93 to the right, agree=0.893, adj=0.019, (0 split)
```

```
Node number 6: 1830 observations
  mean=4.384153, MSE=4.44423

Node number 7: 266 observations,    complexity param=0.002105602
  mean=4.981203, MSE=5.078594
  left son=14 (17 obs) right son=15 (249 obs)
  Primary splits:
   tci        < 3.722733 to the left,  improve=0.06258605, (0 missing)
   beers      < 0.04587005 to the left,  improve=0.03467616, (0 missing)
   utme       < 279021.5 to the left,  improve=0.01318961, (0 missing)
   streamdist < 552  to the left,  improve=0.01242515, (0 missing)
   elev       < 1244.5 to the left,  improve=0.01127290, (0 missing)
  Surrogate splits:
      utmn < 3957386 to the right, agree=0.944, adj=0.118, (0 split)

Node number 8: 2901 observations,    complexity param=0.003387855
  mean=3.675284, MSE=3.711175
  left son=16 (2581 obs) right son=17 (320 obs)
  Primary splits:
 utme       < 252110.5 to the left,  improve=0.009641572, (0 missing)
 beers      < 0.2393186 to the left,  improve=0.009633354, (0 missing)
 utmn       < 3949185 to the left,  improve=0.006225164, (0 missing)
 tci        < 6.275865 to the right, improve=0.004622665, (0 missing)
 streamdist < 274.65 to the right, improve=0.003712651, (0 missing)
  Surrogate splits:
      utmn  < 3950117 to the left,  agree=0.940, adj=0.456, (0 split)
      beers < 1.998653 to the left,  agree=0.893, adj=0.031, (0 split)
      tci   < 14.16 to the left,  agree=0.891, adj=0.016, (0 split)
      elev  < 639.55 to the left,  agree=0.891, adj=0.009, (0 split)

Node number 9: 45 observations
  mean=5.6, MSE=9.528889

Node number 10: 428 observations,    complexity param=0.002112466
  mean=3.490654, MSE=4.712529
  left son=20 (67 obs) right son=21 (361 obs)
  Primary splits:
 streamdist < 174.45 to the right, improve=0.041617160, (0 missing)
 utmn       < 3934079 to the left,  improve=0.019503240, (0 missing)
 tci        < 9.676192 to the left,  improve=0.015050850, (0 missing)
 beers      < 0.488933 to the left,  improve=0.009754094, (0 missing)
 utme       < 292904.5 to the left,  improve=0.009707592, (0 missing)
  Surrogate splits:
  utme  < 313715 to the right, agree=0.888, adj=0.284, (0 split)
  beers < 1.997021 to the right, agree=0.874, adj=0.194, (0 split)
  utmn  < 3946146 to the right, agree=0.867, adj=0.149, (0 split)

Node number 11: 3501 observations,    complexity param=0.002879483
  mean=4.050843, MSE=4.562396
  left son=22 (355 obs) right son=23 (3146 obs)
  Primary splits:
   utme       < 238424 to the left,  improve=0.007163249, (0 missing)
   streamdist < 107.95 to the right, improve=0.007105469, (0 missing)
   tci        < 6.713293 to the left,  improve=0.003658379, (0 missing)
   utmn       < 3938251 to the left,  improve=0.003542249, (0 missing)
   beers      < 0.7289352 to the left,  improve=0.001421292, (0 missing)
```

```
Node number 14: 17 observations
  mean=2.823529, MSE=3.204152

Node number 15: 249 observations
  mean=5.128514, MSE=4.867018

Node number 16: 2581 observations,    complexity param=0.003387855
  mean=3.608679, MSE=3.513664
  left son=32 (1087 obs) right son=33 (1494 obs)
  Primary splits:
  utme       < 242914.5 to the right, improve=0.018242300, (0 missing)
  beers      < 0.2990105 to the left,  improve=0.011881620, (0 missing)
  utmn       < 3946321 to the right, improve=0.006430964, (0 missing)
  tci        < 6.998647 to the right, improve=0.005655161, (0 missing)
  streamdist < 17.07  to the left,  improve=0.003428043, (0 missing)
  Surrogate splits:
  beers      < 0.2393186 to the left,  agree=0.650, adj=0.169, (0 split)
  utmn       < 3946321 to the right, agree=0.646, adj=0.160, (0 split)
  elev       < 544.35 to the right, agree=0.646, adj=0.159, (0 split)
  tci        < 5.257648 to the right, agree=0.602, adj=0.055, (0 split)
  streamdist < 675.8 to the right, agree=0.591, adj=0.029, (0 split)

Node number 17: 320 observations
  mean=4.2125, MSE=4.979844

Node number 20: 67 observations
  mean=2.462687, MSE=3.651593

Node number 21: 361 observations
  mean=3.68144, MSE=4.676913

Node number 22: 355 observations
  mean=3.512676, MSE=3.032938

Node number 23: 3146 observations,    complexity param=0.002532668
  mean=4.11157, MSE=4.698614
  left son=46 (2683 obs) right son=47 (463 obs)
  Primary splits:
  streamdist < 40.615 to the right, improve=0.006808171, (0 missing)
  utmn       < 3937966 to the left,  improve=0.004790859, (0 missing)
  utme       < 272873.5 to the left,  improve=0.004070157, (0 missing)
  tci        < 6.713293 to the left,  improve=0.002842007, (0 missing)
  beers      < 1.902979 to the right, improve=0.001746733, (0 missing)
  Surrogate splits:
  tci   < 11.125 to the left,  agree=0.888, adj=0.240, (0 split)
  beers < 1.999958 to the left,  agree=0.859, adj=0.039, (0 split)
  elev  < 644.4 to the right, agree=0.855, adj=0.017, (0 split)

Node number 32: 1087 observations,    complexity param=0.002325916
  mean=3.311868, MSE=3.151129
  left son=64 (909 obs) right son=65 (178 obs)
  Primary splits:
 elev       < 615.9  to the left,  improve=0.008670238, (0 missing)
 streamdist < 33.84  to the left,  improve=0.007450325, (0 missing)
 tci        < 7.613668 to the right, improve=0.006079286, (0 missing)
 beers      < 1.550484 to the left,  improve=0.005224724, (0 missing)
 utmn       < 3946735 to the right, improve=0.004388856, (0 missing)
```

```
  Surrogate splits:
 streamdist < 574.35 to the left,  agree=0.852, adj=0.096, (0 split)
 tci        < 3.346606 to the right, agree=0.840, adj=0.022, (0 split)
 utmn       < 3936784 to the right, agree=0.839, adj=0.017, (0 split)

Node number 33: 1494 observations
  mean=3.824632, MSE=3.666703

Node number 46: 2683 observations
  mean=4.037272, MSE=4.629248

Node number 47: 463 observations
  mean=4.542117, MSE=4.883215

Node number 64: 909 observations
  mean=3.238724, MSE=3.094826

Node number 65: 178 observations,    complexity param=0.002325916
  mean=3.685393, MSE=3.271809
  left son=130 (157 obs) right son=131 (21 obs)
  Primary splits:
 elev       < 624.45 to the right, improve=0.26639860, (0 missing)
 utme       < 245848.5 to the right, improve=0.12534190, (0 missing)
 tci        < 4.788561 to the left,  improve=0.05535554, (0 missing)
 beers      < 1.186194 to the right, improve=0.05535554, (0 missing)
 streamdist < 364.55 to the right, improve=0.04676413, (0 missing)
  Surrogate splits:
      utme < 245848.5 to the right, agree=0.921, adj=0.333, (0 split)

Node number 130: 157 observations
  mean=3.343949, MSE=2.569597

Node number 131: 21 observations
  mean=6.238095, MSE=1.133787
```

Παρατήρηση Οι βελτιώσεις (improve) που αναγράφονται σε κάθε μη τελικό κόμβο, είναι οι ποσοστιαίες μεταβολές στο άθροισμα τετραγώνων για τον εκάστοτε διαχωρισμό, οι οποίες είναι ίσες με $1 - (SS_R + SS_L)/SS_{parent}$. Για τη διαδικασία του pruning επιλέγουμε από το cptable την τιμή cp που αντιστοιχεί σε μικρό σφάλμα xerror, η οποία είναι η 0.0023259.

```
pfit<- prune(fit, cp=0.0023259)
pfit
n= 8971

node), split, n, deviance, yval
      * denotes terminal node

1) root 8971 39735.670 4.006019
  2) elev< 1157.5 6875 29605.550 3.867636 *
  3) elev>=1157.5 2096  9566.634 4.459924 *
```

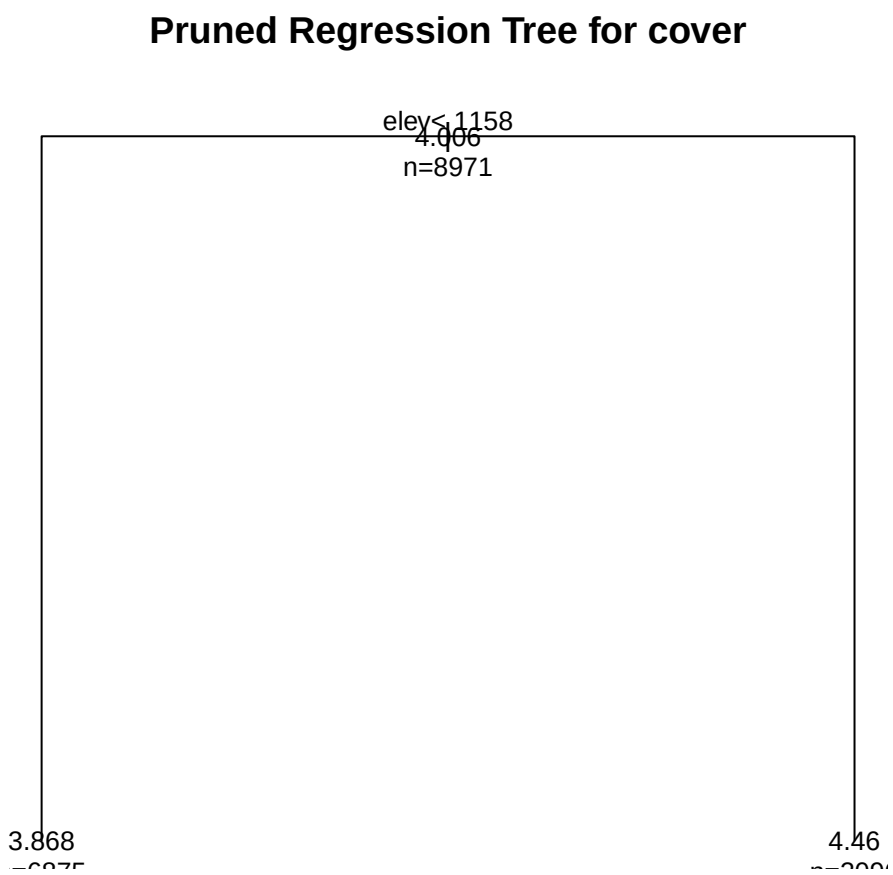


Figure 3.3: Το pruned δέντρο.

Κεφάλαιο 4

Πλεονεκτήματα και Μειονεκτήματα των CART

4.1 Μειονέκτημα των CART

Ένα σοβαρό μειονέκτημα που παρουσιάζουν τα δέντρα ταξινόμησης είναι η ευαισθησία τους σε μικρές μεταβολές στα δεδομένα εκπαίδευσης. Όταν παραλείπονται μερικές παρατηρήσεις ή όταν μεταβάλλεται, προς τα πάνω ή προς τα κάτω, ο λόγος για τα κόστη λανθασμένης ταξινόμησης, είναι πιθανό να αλλάξουν και οι μεταβλητές διαχωρισμού. Κατά συνέπεια προκύπτουν διαφορετικά δέντρα/μοντέλα. Στις προηγούμενες ενότητες χρησιμοποιήθηκε η απώλεια $0-1$ δηλαδή σε περίπτωση λανθασμένης ταξινόμησης ($L(1|0) = L(0|1) = 1$) το κόστος ισούται με 1.

Ωστόσο σε ιατρικά θέματα είναι προφανές ότι η **εσφαλμένα αρνητική** ταξινόμηση έχει χειρότερες συνέπειες απ'ότι η **εσφαλμένα θετική**. Έστω ότι ο λόγος για τα κόστη λανθασμένης ταξινόμησης είναι ο $C(0|1) = 4C(1|0)$ ή ο $C(0|1) = 10C(1|0)$. Τότε επαναλαμβάνοντας την ανάλυση για τα δεδομένα cleveland.processed.data αρχικά με τον 0-1 και κατόπιν με τους υπόλοιπους πίνακες απώλειας έχουμε, μέσω του πακέτου rpart, τα εξής,

0-1 Πίνακας Απώλειας

```
fit1 <- rpart(yfactor~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13, data=heartdata,
parms=list(split="gini",loss=matrix(c(0,1,1,0), byrow=TRUE, nrow=2)),
control=rpart.control(cp=0.001))

fit1
n= 303

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 303 139 0 (0.54125413 0.45874587)
   2) V13< 4.5 167  37 0 (0.77844311 0.22155689)
     4) V12< 0.75 118  13 0 (0.88983051 0.11016949)
       8) V4< 155.5 111   9 0 (0.91891892 0.08108108) *
       9) V4>=155.5 7   3 1 (0.42857143 0.57142857) *
     5) V12>=0.75 49  24 0 (0.51020408 0.48979592)
      10) V3< 3.5 29   7 0 (0.75862069 0.24137931) *
```

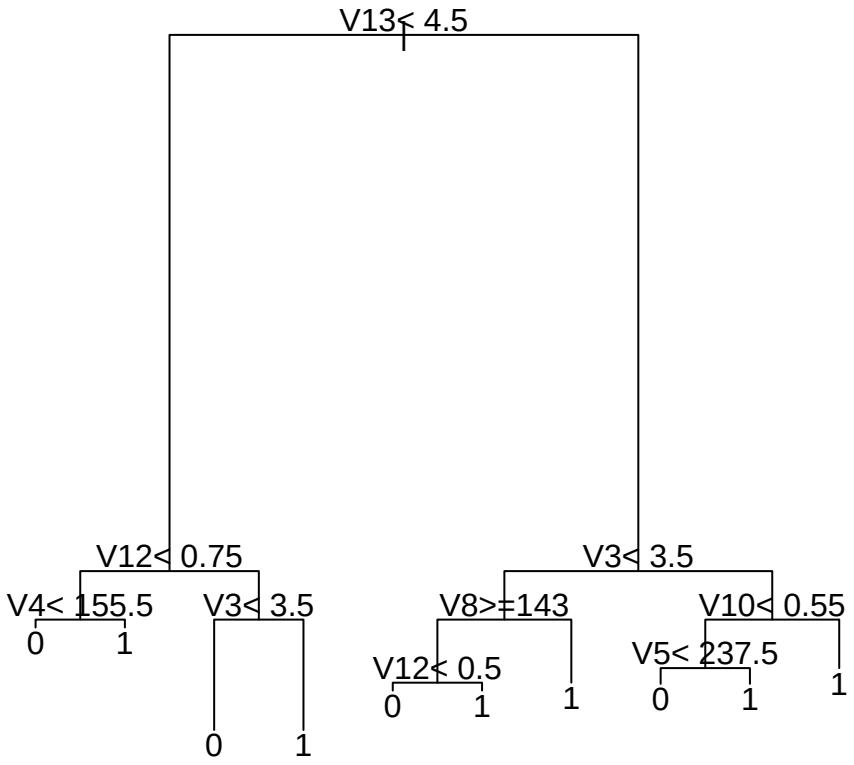
```
      11) V3>=3.5 20   3 1 (0.15000000 0.85000000) *
   3) V13>=4.5 136  34 1 (0.25000000 0.75000000)
     6) V3< 3.5 45  21 0 (0.53333333 0.46666667)
      12) V8>=143 33  11 0 (0.66666667 0.33333333)
        24) V12< 0.5 22   5 0 (0.77272727 0.22727273) *
        25) V12>=0.5 11   5 1 (0.45454545 0.54545455) *
      13) V8< 143 12   2 1 (0.16666667 0.83333333) *
     7) V3>=3.5 91  10 1 (0.10989011 0.89010989)
      14) V10< 0.55 22   8 1 (0.36363636 0.63636364)
        28) V5< 237.5 12   5 0 (0.58333333 0.41666667) *
        29) V5>=237.5 10   1 1 (0.10000000 0.90000000) *
      15) V10>=0.55 69   2 1 (0.02898551 0.97101449) *

summary(fit1)
Call:
rpart(formula = yfactor ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
    V8 + V9 + V10 + V11 + V12 + V13, data = heartdata, parms = list(split = "gini",
    loss = matrix(c(0, 1, 1, 0), byrow = TRUE, nrow = 2)),
    control = rpart.control(cp = 0.001))


  n= 303

           CP nsplit rel error    xerror       xstd
1 0.489208633      0 1.0000000 1.0000000 0.06240124
2 0.050359712      1 0.5107914 0.5899281 0.05563742
3 0.039568345      3 0.4100719 0.5323741 0.05380188
4 0.007194245      5 0.3309353 0.3956835 0.04826931
5 0.001000000      9 0.3021583 0.4532374 0.05082060

Variable importance
V13  V3  V8  V9 V10 V12  V2  V4 V11  V5  V1  V7
 25  18  14  10  10   7   6   4   2   1   1   1
```

Βλέπουμε ότι προέκυψε δέντρο 19 κόμβων εκ των οποίων οι 10 είναι τελικοί με 9 διαχωρισμούς. Επιλέχθηκαν για τους διαχωρισμούς οι μεταβλητές $V_{13}, V_{12}, V_4, V_3, V_8, V_{10}, V_5$.

Η παράμετρος **Cost Complexity** χρησιμεύει στην επιλογή του βέλτιστου μεγέθους του δέντρου [6]. Αν το κόστος της προσθήκης μιας ακόμα μεταβλητής στο δέντρο, υπολογίζοντας από τον τρέχοντα κόμβο, είναι μεγαλύτερο από την τιμή cp τότε το δέντρο δεν επεκτείνεται. Αν το cp ήταν ίσο με το μηδέν, θα δημιουργούνταν το πιο πολύπλοκο δέντρο με το μεγαλύτερο δυνατό « βάθος ».

Το σφάλμα του μητρικού κόμβου (Root Node Error) είναι ίσο με 0.45874587 και χρησιμεύει στον υπολογισμό δύο μέτρων προβλεπτικής απόδοσης, όταν λαμβάνουμε υπόψιν τις τιμές που εμφανίζονται στις στήλες rel error και xerror, αλλά σε συνάρτηση με την παράμετρο πολυπλοκότητας CP. Συγκεκριμένα έχουμε,

- $0.3021583 \times 0.45874587 = 0.1386$ δηλαδή το σφάλμα υπολογισμένο στο training δείγμα ισούται με 13.86%.
- $0.4532374 \times 0.45874587 = 0.2079$ δηλαδή το **Cross Validation σφάλμα** εφαρμόζοντας 10-πλή CV είναι ίσο με 20.79%. Είναι ένα πιο αντικειμενικό μέτρο της προβλεπτικής ακρίβειας.



Σχήμα 4.1: Το πλήρες δέντρο με την 0-1 απώλεια.

Πίνακας Απώλειας $C(0|1) = 4C(1|0)$

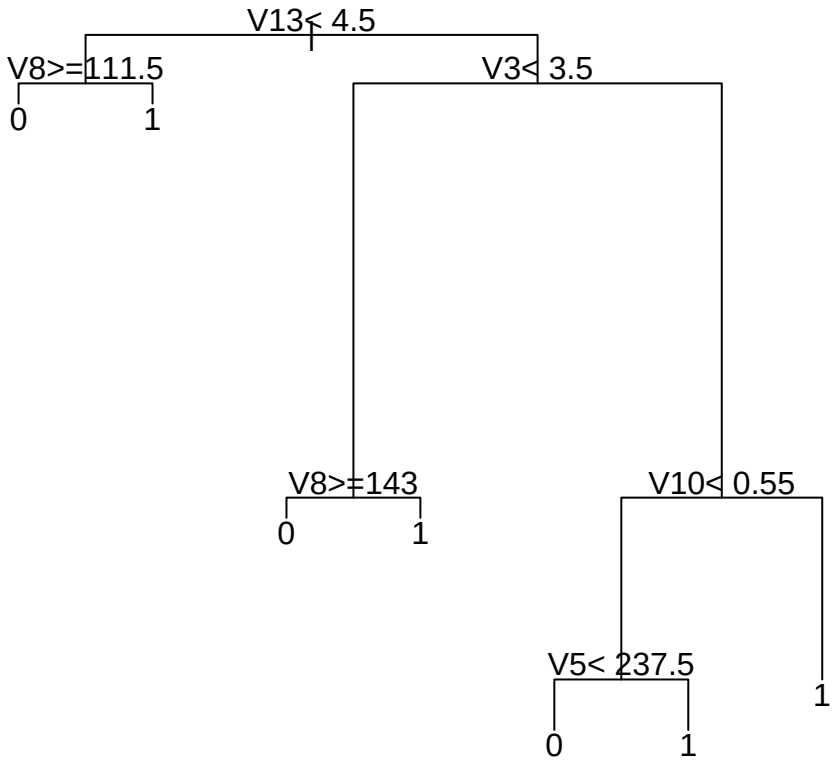
```
fit2 <- rpart(yfactor~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13, data=heartdata,
parms=list(split="gini",loss=matrix(c(0,4,1,0), byrow=TRUE, nrow=2)),
control=rpart.control(cp=0.001))
fit2
n= 303

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 303 139 0 (0.54125413 0.45874587)
   2) V13< 4.5 167  37 0 (0.77844311 0.22155689)
     4) V8>=111.5 160  31 0 (0.80625000 0.19375000) *
     5) V8< 111.5 7   4 1 (0.14285714 0.85714286) *
   3) V13>=4.5 136 102 0 (0.25000000 0.75000000)
     6) V3< 3.5 45  21 0 (0.53333333 0.46666667)
      12) V8>=143 33  11 0 (0.66666667 0.33333333) *
```

```
      13) V8< 143 12   8 1 (0.16666667 0.83333333) *
     7) V3>=3.5 91  40 1 (0.10989011 0.89010989)
      14) V10< 0.55 22  14 0 (0.36363636 0.63636364)
        28) V5< 237.5 12   5 0 (0.58333333 0.41666667) *
        29) V5>=237.5 10   4 1 (0.10000000 0.90000000) *
      15) V10>=0.55 69   8 1 (0.02898551 0.97101449) *

summary(fit2)
Call:
rpart(formula = yfactor ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
    V8 + V9 + V10 + V11 + V12 + V13, data = heartdata, parms = list(split = "gini",
    loss = matrix(c(0, 4, 1, 0), byrow = TRUE, nrow = 2)),
    control = rpart.control(cp = 0.001))
  n= 303

          CP nsplit rel error   xerror      xstd
1 0.14748201      0 1.0000000 4.000000 0.2496050
2 0.12949640      2 0.7050360 2.525180 0.2220210
3 0.03597122      3 0.5755396 2.388489 0.2191854
4 0.01438849      4 0.5395683 2.035971 0.2078477
5 0.00100000      6 0.5107914 2.064748 0.2089002

Variable importance
V13  V3  V8 V10  V9  V2  V5 V11  V4 V12  V7  V1
 23  21  17  15   8   6   3   2   2   1   1   1
```

Βλέπουμε ότι προέκυψε δέντρο 13 κόμβων εκ των οποίων οι 7 είναι τελικοί με 6 διαχωρισμούς. Επιλέχθηκαν για τους διαχωρισμούς μόνο οι μεταβλητές $V_{13}, V_3, V_8, V_{10}, V_5$.

Το σφάλμα του μητρικού κόμβου (Root Node Error) είναι ίσο με 0.45874587 και χρησιμεύει στον υπολογισμό δύο μέτρων προβλεπτικής απόδοσης, όταν λαμβάνουμε υπόψιν τις τιμές που εμφανίζονται στις στήλες rel error και xerror, αλλά σε συνάρτηση με την παράμετρο πολυπλοκότητας CP. Συγκεκριμένα έχουμε,

- $0.5107914 \times 0.45874587 = 0.2343$ δηλαδή το σφάλμα υπολογισμένο στο training δείγμα ισούται με 23.43%.
- $2.064748 \times 0.45874587 = 0.947$ δηλαδή το **Cross Validation σφάλμα** εφαρμόζοντας 10-πλή CV είναι ίσο με το ακραία υψηλό 94.7%. (Είναι ένα πιο αντικειμενικό μέτρο της προβλεπτικής ακρίβειας.)



Σχήμα 4.2: Το πλήρες δέντρο με τον πίνακα απώλειας $C(0|1) = 4C(1|0)$.

Πίνακας Απώλειας $C(0|1) = 10C(1|0)$

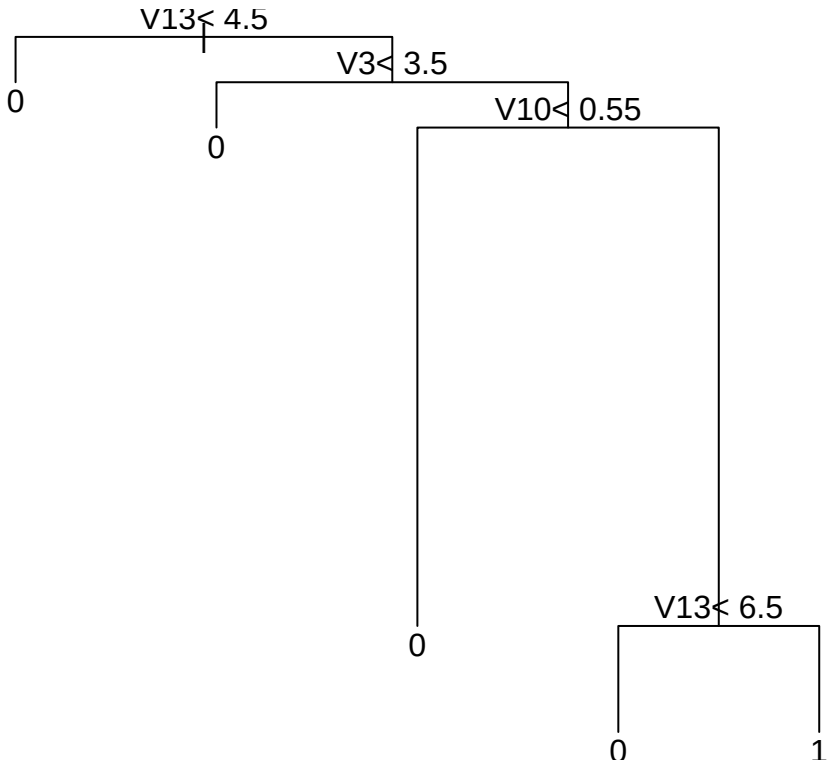
```
fit3 <- rpart(yfactor~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13, data=heartdata,
parms=list(split="gini",loss=matrix(c(0,10,1,0), byrow=TRUE, nrow=2)),
control=rpart.control(cp=0.001))
fit3
n= 303

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 303 139 0 (0.54125413 0.45874587)
   2) V13< 4.5 167  37 0 (0.77844311 0.22155689) *
   3) V13>=4.5 136 102 0 (0.25000000 0.75000000)
     6) V3< 3.5 45  21 0 (0.53333333 0.46666667) *
     7) V3>=3.5 91  81 0 (0.10989011 0.89010989)
      14) V10< 0.55 22  14 0 (0.36363636 0.63636364) *
      15) V10>=0.55 69  20 1 (0.02898551 0.97101449)
```

```
        30) V13< 6.5 12  10 0 (0.16666667 0.83333333) *
        31) V13>=6.5 57   0 1 (0.00000000 1.00000000) *

summary(fit3)
Call:
rpart(formula = yfactor ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
    V8 + V9 + V10 + V11 + V12 + V13, data = heartdata, parms = list(split = "gini",
    loss = matrix(c(0, 10, 1, 0), byrow = TRUE, nrow = 2)),
    control = rpart.control(cp = 0.001))
  n= 303

          CP nsplit rel error    xerror      xstd
1 0.11270983      0 1.0000000 10.000000 0.6240124
2 0.07194245      3 0.6618705  6.748201 0.5757295
3 0.00100000      4 0.5899281  6.338129 0.5641596

Variable importance
V13 V10  V3  V8  V9  V2 V11  V4  V1
 26  25  21  12   6   4   3   3   1
```

Βλέπουμε ότι προέκυψε δέντρο κόμβων εκ των οποίων οι είναι τελικοί με διαχωρισμούς. Επιλέχθηκαν για τους διαχωρισμούς μόνο οι μεταβλητές V_{13}, V_3, V_{10}.

Το σφάλμα του μητρικού κόμβου (Root Node Error) είναι ίσο με 0.45874587 και χρησιμεύει στον υπολογισμό δύο μέτρων προβλεπτικής απόδοσης, όταν λαμβάνουμε υπόψιν τις τιμές που εμφανίζονται στις στήλες rel error και xerror, αλλά σε συνάρτηση με την παράμετρο πολυπλοκότητας CP. Συγκεκριμένα έχουμε,

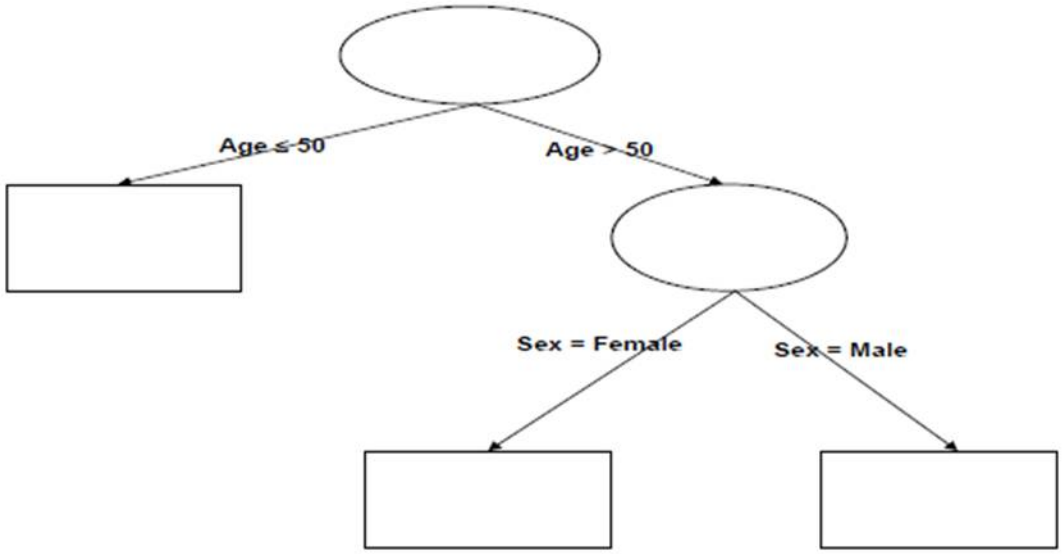
- $0.5899281 \times 0.45874587 = 0.27$ δηλαδή το σφάλμα υπολογισμένο στο training δείγμα ισούται με 27%.
- $6.338129 \times 0.45874587 = 2.91$ δηλαδή το **Cross Validation σφάλμα** εφαρμόζοντας 10-πλή CV είναι ακραίο.



Σχήμα 4.3: Το πλήρες δέντρο με τον $C(0|1) = 10C(1|0)$ πίνακα απώλειας.

4.2 Πλεονέκτηματα των CART

- Τα CART είναι **μη-παραμετρικά** κατά συνέπεια δεν προϋποθέτουν τα δεδομένα ν'ακολουθούν την Κανονική κατανομή.
- Πολύ συχνά στα δεδομένα υπάρχουν πολύπλοκες αλληλεπιδράσεις οι οποίες μοντελοποιούνται δύσκολα, ενώ όταν είναι και πολλές καθιστούν αδύνατη τη μοντελοποίηση.



Σχήμα 4.4: Το δέντρο δίνει έμφαση στην αλληλεπίδραση μεταξύ ηλικίας και φύλου. Αν χρησιμοποιούνταν ως διερευνητικό εργαλείο, τότε θα πρότεινε έναν όρο αλληλεπίδρασης μεταξύ των 2 μεταβλητών στο μοντέλο.

- Τα δέντρα CART μπορούν να χειριστούν όλων των ειδών τα δεδομένα (συνεχή, κατηγορικά, δϋαδικά, διατακτικά), χωρίς να απαιτούν κάποιον μετασχηματισμό.
- Είναι αποτελεσματικά στην αντιμετώπιση των **outliers** καθώς ο διαχωριστικός αλγόριθμος τα απομονώνει σε ξεχωριστό κόμβο.
- Συχνά τίθεται το ζήτημα επιλογής επεξηγηματικών μεταβλητών που είναι πολύ δύσκολο ιδιαίτερα όταν το πλήθος των μεταβλητών είναι μεγάλο. Όμως τα δέντρα δεν απαιτούν την προεπιλογή των μεταβλητών αφού ο αλγόριθμος ξεχωρίζει τις πιο σημαντικές αγνοώντας τις υπόλοιπες.
- Τα CART παράγουν μοντέλα που ερμηνεύονται εύκολα.

Κεφάλαιο 5

Εισαγωγή στις Boosting και Bagging μεθόδους

5.1 Εισαγωγή

Η μέθοδος Boosting η οποία βασίζεται σε ένα σύνολο ατομικών μοντέλων, ανήκει στις μεθόδους μηχανικής μάθησης (machine learning) και χρησιμοποιείται την τελευταία δεκαετία. Η μηχανική μάθηση, η οποία είναι κλάδος της τεχνητής νοημοσύνης, ασχολείται με το σχεδιασμό και την ανάπτυξη αλγορίθμων, με στόχο την αποτύπωση προτύπων και προβλέψεων που αποτελούν και τα χαρακτηριστικά του μηχανισμού που γέννησε τα δεδομένα. Η βασική ιδέα του Boosting είναι ο συνδυασμός των αποτελεσμάτων πολλών « αδύναμων » αλγορίθμων μάθησης. Δηλαδή ο συνδυασμός αλγορίθμων των οποίων το σφάλμα (error rate) είναι ελαφρώς καλύτερο από την τυχαία επιλογή.

Ο αλγόριθμος AdaBoost είναι η πιό δημοφιλής διαδικασία boosting καθώς είναι ιδιαίτερα αποτελεσματικός στις περισσότερες περιπτώσεις. Η στατιστική ανάλυση του AdaBoost μέσω του εκθετικού κριτηρίου έδειξε ότι εκτιμά τους λογαριθμημένους λόγους των πιθανοτήτων για κάθε τάξη. Υπάρχουν αρκετές εφαρμογές του boosting όπως ο Gradient Boosting (GB) του Friedman, στοχαστικές παραλλαγές του GB (SGB) κ.α. Με την ελαχιστοποίηση διαφορετικών συναρτήσεων απώλειας, η παρούσα μεθοδολογία μπορεί ν'αντιμετωπίσει εξίσου αποτελεσματικά προβλήματα ταξινόμησης και παλινδρόμησης. Το σημαντικό χαρακτηριστικό της είναι ότι μπορεί ν'αποτυπώσει με ακρίβεια τη μη γραμμική δομή των δεδομένων με το να προσθέτει προσαρμόζοντας κάποιο σετ βασικών αλγορίθμων μάθησης.

Μεταξύ των πολλών επιστημονικών περιοχών στις οποίες εφαρμόζεται η μέθοδος boosting είναι και η χημεία [5]. Καθώς συγκεντρώνει πολλές επιθυμητές ιδιότητες, πολλοί χημικοί επικεντρώνονται σ'αυτήν για την αντιμετώπιση θεμάτων της βιοπληροφορικής, της φασματοσκοπικής μεθόδου εγγύς υπερύθρου (near infrared spectroscopy), της φασματομετρίας μαζών (mass spectrometry analysis), της πρόβλεψης πρωτεϊνικών δομών και λειτουργιών, των μοντέλων QSAR/QSPR quantitative structure activity/property relationship, κ.α.

5.2 Η βασική ιδέα του boosting

Οι αλγόριθμοι boosting [6] χτίζουν πολλαπλά μοντέλα από ένα σύνολο δεδομένων, χρησιμοποιώντας και άλλους τρόπους κατασκευής μοντέλων, όπως είναι τα δέντρα αποφάσεων, τα οποία δεν δίνουν κατ'ανάγκη κάποιο εξαιρετικά καλό μοντέλο. Η βασική ιδέα του boosting είναι η σύνδεση κάθε παρατήρησης του συνόλου δεδομένων με ένα βάρος. Χτίζεται λοιπόν ένα πλήθος μοντέλων και τα βαρη **ενισχύονται**, εξ'ού και ο αγγλικός όρος **boosted**, εαν ένα μοντέλο ταξινομεί λανθασμένα την παρατήρηση.

Τα βάρη των παρατηρήσεων γενικά παρουσιάζουν διακυμάνσεις προς τα πάνω και προς τα κάτω από το ένα μοντέλο στο επόμενο. Το τελικό μοντέλο είναι αθροιστικό, κατασκευασμένο από μια ακολουθία μοντέλων, με το αποτέλεσμα καθενός εξ'αυτών να έχει λάβει κάποιο βάρος. Τα πλεονεκτήματα της μεθόδου boosting είναι ότι απαιτείται μικρή προσαρμογή (tuning) και ότι η μόνη υπόθεση που κάνουμε για τον κατασκευαστή του μοντέλου είναι ότι πρέπει να είναι σχετικά αδύναμος. Τα μειονεκτήματα του boosting είναι ότι μπορεί να αποτύχει είτε όταν τα δεδομένα δεν είναι επαρκή, είτε όταν τα αδύναμα μοντέλα είναι εξαιρετικά πολύπλοκα και επιπλέον ότι είναι ευαίσθητο στα τυχαία σφάλματα.

Σ'ένα πρόβλημα ταξινόμησης στοχεύουμε στη δημιουργία ενός κανόνα που δίνει τις πιο ακριβείς προβλέψεις για τα επόμενα **test sets**. Το test set είναι ένα σύνολο δεδομένων με τη βοήθεια του οποίου **αξιολογούμε** την ισχύ και τη χρησιμότητα μιας προβλεπτικής σχέσης. Ενώ τα δεδομένα που χρησιμοποιούμε για την **κατασκευή ή ανακάλυψη** μιας προβλεπτικής σχέσης καλούνται **training set**. Τα δυο αυτά είδη δεδομένων είναι εντελώς ανεξάρτητα μεταξύ τους, αλλά ακολουθούν την ίδια κατανομή πιθανότητας. Τα test sets και τα training sets έχουν εφαρμογή στην τεχνητή νοημοσύνη, στη μηχανική μάθηση και στη στατιστική.

Προκειμένου να εφαρμοστεί η μέθοδος, δυο είναι τα βασικά ερωτήματα που πρέπει ν'απαντηθούν. Πρώτον, με ποιόν τρόπο θα πρέπει να επιλεγεί ο κάθε αδύναμος ταξινομητής, και δεύτερον, μετά τη συλλογή πολλών τέτοιων ταξινομητών, με ποιον τρόπο θα συνδυαστούν σε έναν.

5.3 Η βασική ιδέα του bagging

Η τεχνική του **B**ootstrap **Agg**regat**ing** χρησιμεύει στη βελτίωση της προβλεπτικής ικανότητας των δέντρων ταξινόμησης ή παλινδρόμησης. Αποτελεί μια μέθοδο « ψήφου » όπου οι αδύναμοι μηχανισμοί εκμάθησης διαφοροποιούνται εκπαιδεύοντάς τους σε training σύνολα δεδομένων που διαφέρουν ελάχιστα μεταξύ τους. Με τη μέθοδο Bootstrap προκύπτουν τα L δείγματα ως εξής: από το σύνολο δεδομένων εκπαίδευσης X μεγέθους N λαμβάνεται ένα τυχαίο δείγμα επίσης μεγέθους N, με επανάθεση.

Λόγω της δειγματοληψίας με επανάθεση είναι πιθανό ορισμένες παρατηρήσεις να επιλεγούν περισσότερες από μια φορές και άλλες να μην επιλεγούν ποτέ. Κατόπιν οι αδύναμοι μηχανισμοί εκμάθησης d_j εκπαιδεύονται στα L δείγματα $X_j, j = 1 \ldots, L$ χρησιμοποιώντας μια ασταθή διαδικασία εκμάθησης και στο τέλος υπολογίζεται ο μέσος όρος. Η μέθοδος Bagging εφαρμόζεται σε προβλήματα ταξινόμησης και παλινδρόμησης με τη διαφορά ότι στη δεύτερη περίπτωση, για λόγους ανθεκτικότητας, λαμβάνεται αντί του μέσου όρου η διάμεσος κατά το συνδυασμό προβλέψεων.

Αλγοριθμικά η διαδικασία έχει ως εξής [7],

1. Έστω n το πλήθος των παρατηρήσεων του συνόλου εκπαίδευσης. Για κάθε μια από τις t επαναλήψεις:

   (α') Πάρτε δείγμα μεγέθους n με επανάθεση από το σύνολο εκπαίδευσης.

   (β') Εφαρμόστε τον αλγόριθμο εκμάθησης στο δείγμα.

   (γ') Αποθήκευση του μοντέλου που προκύπτει.

2. Για καθένα από τα t στο πλήθος μοντέλα (περίπτωση ταξινόμησης),

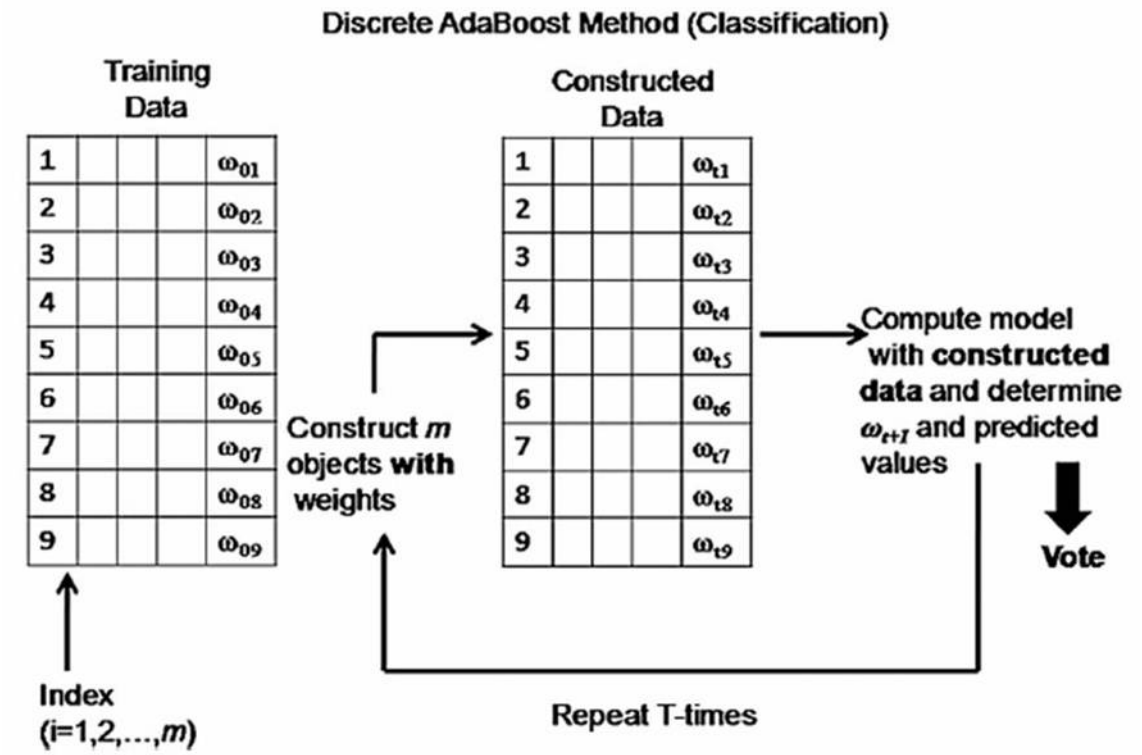
   (α') Προβλέψτε την τάξη με τη χρήση του μοντέλου.

   (β') Επιστρέψτε την τάξη που έλαβε τις περισσότερες προβλέψεις.

Κεφάλαιο 6

Boosting και Ταξινόμηση

Η συνήθης μέθοδος boosting που εφαρμόζεται σε προβλήματα ταξινόμησης είναι ο αλγόριθμος AdaBoost των Freund και Schapire. Υπάρχουν 3 είδη αλγορίθμων AdaBoost: ο discrete AdaBoost, ο real AdaBoost και ο gentle AdaBoost. Ακόμη ο αλγόριθμος AdaBoost.MH εφαρμόζεται σε προβλήματα ταξινόμησης πολλαπλών τάξεων.



Σχήμα 6.1: Σχηματική απεικόνιση της διαδικασίας boosting για την ταξινόμηση.

Έστω ότι έχουμε ένα training σετ δεδομένων που αποτελείται από N παρατηρήσεις οι οποίες ανήκουν σε δυο τάξεις. Οι δυο τάξεις παίρνουν τις τιμές -1 και 1, δηλαδή $y \in \{-1, 1\}$.

Ο discrete AdaBoost αλγόριθμος αποτελείται από τα ακόλουθα βήματα [5]:

Α. Αρχικά αναθέστε ίσα βάρη για κάθε μια παρατήρηση του αρχικού training σετ:

$$w_i{}^1 = 1/N, i = 1, 2, \ldots, N$$

Β. Για τις επαναλήψεις $t = 1, 2, \ldots, T$:

1. Επιλέξτε ένα σετ δεδομένων N παρατηρήσεων απ'το αρχικό training set με τη χρήση επαναδειγματοληψίας με βάρη. Η πιθανότητα επιλογής μιας παρατήρησης εξαρτάται απ'το βάρος που της έχει ανατεθεί. Μια παρατήρηση με μεγάλο βάρος έχει μεγαλύτερη πιθανότητα να επιλεγεί.

2. Βρείτε απ'το σετ δεδομένων της επαναδειγματοληψίας μια συνάρτηση εκμάθησης $f(x)$.

3. Εφαρμόστε τη συνάρτηση εκμάθησης $f(x)$ στο αυθεντικό training set. Εαν μια παρατήρηση ταξινομηθεί λάνθασμένα, το σφάλμα της θα είναι ίσο με 1 διαφορετικά θα ισούται με μηδέν.

4. Υπολογίστε το άθροισμα των σφαλμάτων με τα βάρη τους για όλες τις παρατηρήσεις του training δείγματος.

$$err^t = \sum_{i=1}^{N} (w_i{}^t \times err_i{}^t)$$

Ο δείκτης εμπιστοσύνης της $f(x)$ υπολογίζεται με τον ακόλουθο τρόπο:

$$c^t = log((1 - err^t)/err^t)$$

Είναι προφανές ότι όσο μικρότερο είναι το σφάλμα err^t της συνάρτησης εκμάθησης $f(x)$ πάνω στο training sample, τόσο μεγαλύτερος είναι ο δείκτης εμπιστοσύνης της.

5. Αναβαθμίστε το βάρος όλων των αρχικών training παρατηρήσεων

$$w_i{}^{t+1} = w_i{}^t exp(c^t err_i{}^t)$$

Τα βάρη των σωστά ταξινομημένων παρατηρήσεων δεν αλλάζουν, ενώ εκείνα των λανθασμένων παρατηρήσεων αυξάνονται.

6. Επανακανονικοποιείστε τα w ούτως ώστε $\sum_{i=1}^{N} w_i{}^{t+1} = 1$

7. $T = t + 1$ εαν $err < 0.5$ και $t < T$, επαναλάβετε τα βήματα (1)-(6). Διαφορετικά, σταματήστε και θέστε $T = t - 1$. Μετά από T επαναλήψεις του βήματος Β, έχουμε T συναρτήσεις εκμάθησης $f^t(x), t = 1, 2, \ldots, T$.

Η απόδοση του Discrete AdaBoost αξιολογείται μέσω του test set. Για μια παρατήρηση j του test set, η τελική πρόβλεψη είναι η συνδυασμένη πρόβλεψη που δίνουν οι T τιμές $f^t(x)$. Κάθε πρόβλεψη πολλαπλασιάζεται με το δείκτη εμπιστοσύνης της αντίστοιχης συνάρτησης εκμάθησης $f^t(x)$. Όσο μεγαλύτερος είναι ο δείκτης εμπιστοσύνης, τόσο σημαντικότερος είναι ο ρόλος της αντίστοιχης συνάρτησης εκμάθησης στην τελική αποφαση. Συγκεκριμένα η πρόβλεψη δίνεται απ'τον τύπο

$$y_j = sign(\sum_{t=1}^{T} c^t f^t(x_j))$$

Πλεονεκτήματα του αλγορίθμου AdaBoost

- Ο προγραμματισμός του είναι γρήγορος, απλός και εύκολος.
- Η μοναδική παράμετρος που προσαρμόζεται είναι το πλήθος των επαναλήψεων T.
- Δεν απαιτείται εκ των προτέρων γνώση του αδύναμου ταξινομητή άρα μπορεί να εφαρμοστεί με οποιαδήποτε μέθοδο αδύναμης ταξινόμησης.

- Εντοπίζει outliers αφού επικεντρώνεται σε παρατηρήσεις οι οποίες ταξινομούνται δυσκολότερα. Οι παρατηρήσεις με τα υψηλότερα βάρη συχνά αποδεικνύεται ότι είναι άτυπα σημεία (outliers). Στην περίπτωση μας τα άτυπα σημεία υπάρχουν είτε λόγω λάθους στην ονομασία τους στο training δείγμα, είτε λόγω πραγματικής δυσκολίας στην ταξινόμηση τους.

Από τον παραπάνω αλγόριθμο προκύπτουν ενδιαφέροντα συμπεράσματα. Κάθε αδύναμος μηχανισμός εκμάθησης προκύπτει παίρνοντας δείγματα από την επαναληπτικά ανανεούμενη κατανομή ολόκληρου του training συνόλου δεδομένων. Κατά συνέπεια ο AdaBoost διατηρεί την κατανομή ή το σύνολο των βαρών πάνω στο αρχικό training σύνολο έστω Z και προσαρμόζει αυτά τα βάρη μετά την εκμάθηση του κάθε ταξινομητή από τους αδύναμους μηχανισμούς εκμάθησης.

Οι προσαρμογές αυξάνουν τα βάρη των παρατηρήσεων που ταξινομήθηκαν λανθασμένα από τις αδύναμες συναρτήσεις εκμάθησης και ελαττώνουν τα βάρη των σωστά ταξινομημένων παρατηρήσεων. Άρα βλέπουμε τους συνεχόμενους ταξινομητές να εστιάζουν σταδιακά στις παρατηρήσεις που είναι δύσκολο να ταξινομηθούν.

Από τη σχέση για τον υπολογισμό του δείκτη εμπιστοσύνης $c^t = log((1-err^t)/err^t)$ βλέπουμε ότι όσο υψηλότερη είναι η τιμή του, τόσο πιο ακριβής είναι η αντίστοιχη αδύναμη συνάρτηση εκμάθησης. Μια μεγάλη τιμή του δείκτη καταδεικνύει ότι ο αντίστοιχος μηχανισμός εκμάθησης έχει σημαντικότερο ρόλο στην τελική λήψη της απόφασης. Ουσιαστικά ο AdaBoost υιοθετεί τη στρατηγική της Weighted Majority Voting προκειμένου ν'αναθέσει βάρη σε μια σειρά κανόνων εκμάθησης. Η βασική ιδέα της εν λόγω στρατηγικής είναι αποτελεσματική και καθαρά διαισθητική. Εκείνοι οι ταξινομητές οι οποίοι ήταν ιδιαιτέρως αποδοτικοί κατά τη διάρκεια της εκπαίδευσης, ανταμοίβονται με υψηλότερα βάρη ψήφου σε σχέση με τους υπόλοιπους.

Σημειώνουμε ότι ο δείκτης εμπιστοσύνης είναι μεγαλύτερος του μηδενός, επομένως είναι εξασφαλισμένο ότι ο αδύναμος μηχανισμός εκμάθησης (weak learner) είναι καλύτερος από την τυχαία επιλογή. Η εξίσωση

$$w_i{}^{t+1} = w_i{}^t exp(c^t err_i{}^t)$$

περιγράφει τον κανόνα αναβάθμισης της κατανομής του training set. Κάθε αναβάθμιση της κατανομής του training set ελέγχεται από τα βάρη των training παρατηρήσεων. Τα βάρη της κατανομής των σωστά ταξινομημένων παρατηρήσεων μένουν αμετάβλητα, ενώ τα αντίστοιχα βάρη των λανθασμένα ταξινομημένων παρατηρήσεων ενισχύονται. Κατά συνέπεια, ο AdaBoost επανάληψη στην επανάληψη, αναβαθμίζει την κατανομή των training παρατηρήσεων προκειμένου να δοθεί έμφαση στις παρατηρήσεις που έχουν αυξανόμενη δυσκολία στην ταξινόμηση. Όταν επιτευχθεί η T επανάληψη που έχουμε θέσει, ο AdaBoost είναι έτοιμος να ταξινομήσει τις test παρατηρήσεις.

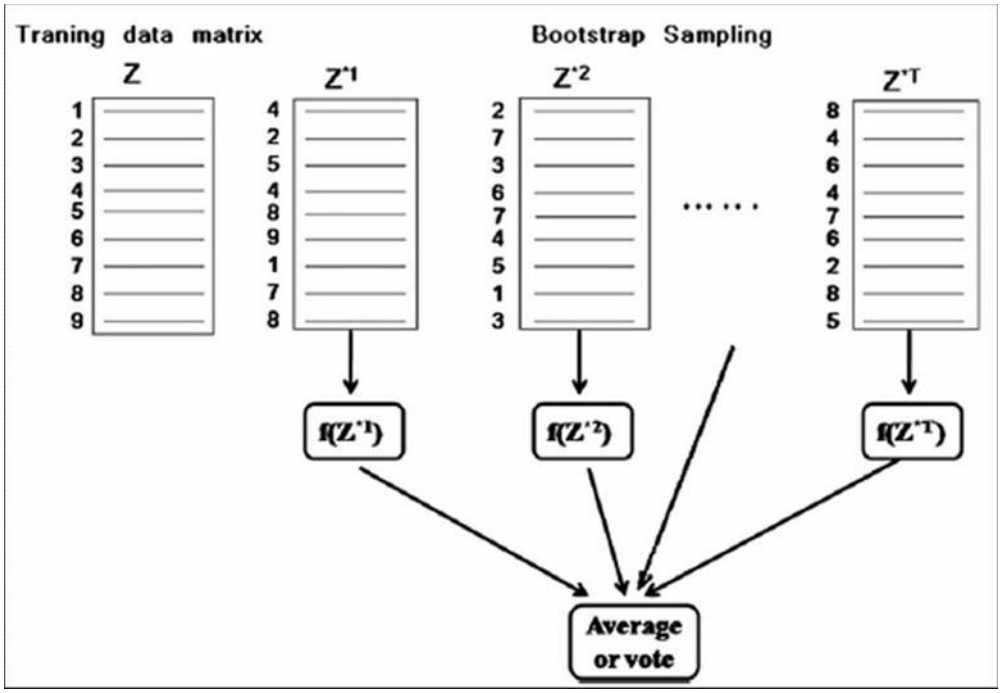
Η τεχνική του Bagging

Οι αλγόριθμοι Boosting χρησιμοποιούν την εν λόγω τεχνική, επομένως οφείλουμε ν'αναφερθούμε σε αυτήν. Η τεχνική Bagging [5] είναι εξίσου δημοφιλής με τη μέθοδο Boosting, η οποία επίσης δημιουργεί ένα σύνολο ταξινομητών με επαναδειγματοληψία των δεδομένων, οι οποίοι συνδυάζονται με τη μέθοδο του majority voting παρόλο

που έχουν ουσιαστικές διαφορές. Η ονομασία της προήλθε ως ακολούθως **B**ootstrap **agg**regat**ing**.

Πρόκειται για μια Bootstrap μέθοδο η οποία δημιουργεί παρατηρήσεις για το σύνολό της, με την εκπαίδευση κάθε ταξινομητή πάνω σε μια τυχαία ανακατανομή των δεδομένων εκπαίδευσης (training set). Κατά συνέπεια ενσωματώνει τα οφέλη τόσο της bootstrap όσο και της aggregating προσέγγισης. Στο σχήμα 6.2 φαίνεται καθαρά ότι πρώτον, παράγονται πολλά bootstrap δείγματα από το αρχικό training σετ. Ένα bootstrap δείγμα παράγεται με τυχαία δειγματοληψία με επανάθεση.

Στο πρόβλημα της παλινδρόμησης, έστω ότι θέλουμε να προσαρμόσουμε ένα μοντέλο στα training δεδομένα Z, ώστε να λάβουμε την πρόβλεψη $f(x)$ για την τιμή x. Από την τυχαία δειγματοληψία με επανάθεση δημιουργούνται N δείγματα, όπου $Z^* = \{(x_1{}^*, y_1{}^*), (x_2{}^*, y_2{}^*), \ldots, (x_N{}^*, y_N{}^*)\}$. Αξίζει να σημειωθεί ότι κάποιες από τις παρατηρήσεις του αρχικού δείγματος ενδέχεται να επαναλαμβάνονται στο training set που προκύπτει, ενώ κάποιες άλλες ενδέχεται να μην εμφανιστούν καθόλου. Από στοιχεία προσομοιώσεων έχει βρεθεί ότι περίπου το 37% των παρατηρήσεων δεν εμφανίζεται στο bootstrap δείγμα.



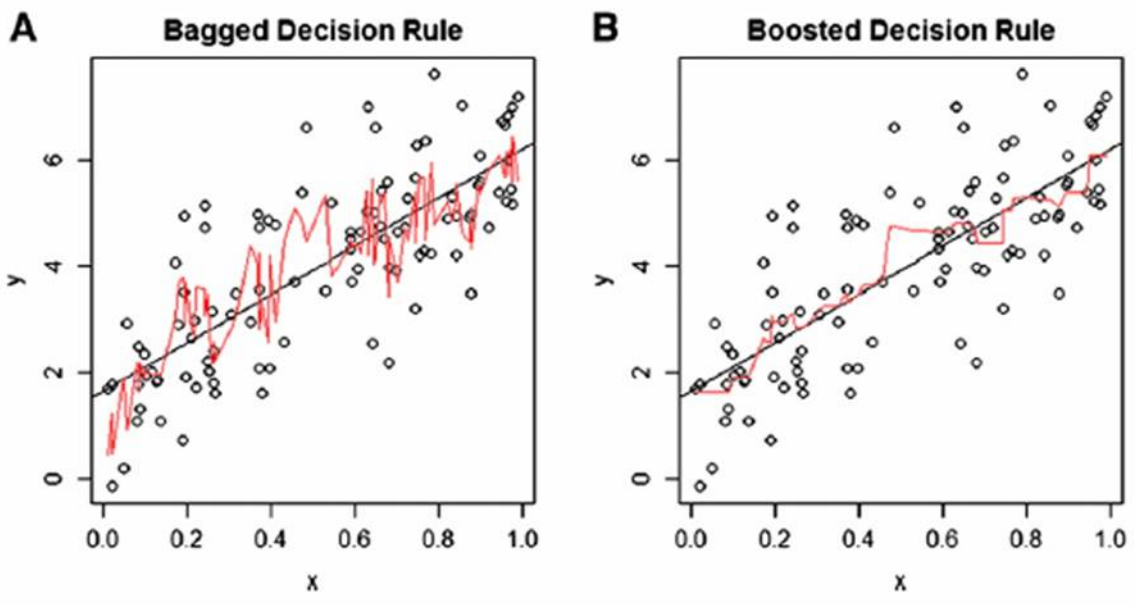
Σχήμα 6.2: Η σχηματική απεικόνιση της διαδικασίας Bagging. T: το πλήθος των επαναλήψεων. Στην παλινδρόμηση παίρνουμε το μέσο όρο T τιμών πρόβλεψης. Στην ταξινόμηση δίνεται ψήφος για T τιμές πρόβλεψης.

Η βασική ιδέα της προσέγγισης aggregating είναι ουσιαστικά ο συνδυασμός πολλαπλών μοντέλων. Για κάθε bootstrap σετ $Z^{*t}, t = 1, 2, \ldots, T$, προσαρμόζεται ένα μοντέλο το οποίο δίνει μια πρόβλεψη $f(x)$. Η bagging εκτίμηση, δηλαδή ο μέσος όρος των προβλέψεων για μια συλλογή bootstrap παρατηρήσεων, δίνεται από την ακόλουθη σχέση,

$$\hat{f}_{bag}(x) = \frac{1}{T}\sum_{t=1}^{T}\hat{f}^{*t}(x)$$

Είναι άξιο λόγου ότι ο συνδυασμός μοντέλων σε επίπεδο πληθυσμού δε χειροτερεύει ποτέ το μέσο άθροισμα τετραγώνων των σφαλμάτων. Σύμφωνα με τον Breiman το bagging είναι ιδιαίτερα αποτελεσματικό σε ασταθείς αλγορίθμους εκμάθησης, στους οποίους μια μικρή μεταβολή στα δεδομένα έχει ως αποτέλεσμα μια μεγάλη μεταβολή στα αποτελέσματα. Το bagging μπορεί να μειώσει σημαντικά τη μεταβλητότητα ασταθών διαδικασιών όπως είναι τα δέντρα αποφάσεων και τα νευρωνικά δίκτυα, οδηγώντας σε μια βελτιωμένη πρόβλεψη, διότι ο μέσος όρος μειώνει τη μεταβλητότητα και αφήνει ως έχει την μεροληψία.

Προκειμένου να γίνει κατανοητό πως το bagging βρίσκει την πραγματική συνάρτηση παλινδρόμησης δίνεται το ακόλουθο απλό παράδειγμα με μια ανεξάρτητη μεταβλητή. Έστω $y_i = 2 + 4x_i + \epsilon_i, i = 1, 2, \ldots, n$ όπου ϵ_i είναι το σφάλμα που ακολουθεί την $N(0, 1)$ και κάθε x_i ακολουθεί την ομοιόμορφη U[0,1]. Όλες οι τιμές x_i είναι ανεξάρτητες απ'όλες τις τιμές ϵ_i. Η πραγματική συνάρτηση παλινδρόμησης (ο στόχος) είναι φυσικά η $F_{true}(x_i) = 2 + 4x_i, x_i \in [0, 1]$. Χρησιμοποιούμε stumps, δηλαδή απλά δέντρα με δύο τελικούς κόμβους και έναν μόνο εσωτερικό κόμβο/ρίζα, ως διαδικασία εκμάθησης και υπολογίζουμε το μέσο όρο των προβλέψεων 50 επαναλήψεων.



Σχήμα 6.3: Ένα απλό παράδειγμα παλινδρόμησης με μια μόνο μεταβλητή πρόβλεψης. (Α) Η εκτιμημένη με τη μέθοδο Bagging ευθεία παλινδρόμησης. (Β) Η εκτιμημένη με τη μέθοδο Boosting ευθεία παλινδρόμησης. Και οι δύο διαδικασίες έγιναν με 50 επαναλήψεις.

Είναι εμφανές ότι η εκτίμηση της μεθόδου Bagging είναι μια μάλλον αδρή και λανθασμένη προσέγγιση της συνάρτησης στόχου. Από το σχήμα φαίνεται ότι η μέθοδος Boosting είναι καταλληλότερη για τη μοντελοποίηση της πραγματικής ευθείας παλινδρόμησης.

6.1 Partial Least Squares Διαχωριστική Ανάλυση

Η PLS-DA είναι ευρέως διαδεδομένη τεχνική η οποία εφαρμόζεται σε προβλήματα ταξινόμησης υψηλών διαστάσεων. Συνήθως χρησιμοποιείται ως μια μέθοδος αναφοράς για τον έλεγχο άλλων τρόπων μοντελοποίησης. Η PLS αρχικά σχεδιάστηκε για συνεχείς αποκρίσεις και η υιοθέτησή της για την αντιμετώπιση προβλημάτων ταξινόμησης υψηλών διαστάσεων είναι σχετικά πρόσφατη. Οι Barker και Rayens το 2003 δικαιολόγησαν τη χρήση της PLS για ταξινόμηση συνδέοντάς την με τη γραμμική διαχωριστική ανάλυση του Fisher.

Η PLS-DA είναι στην πραγματικότητα η άμεση εφαρμογή της αρχικής PLS η οποία βρίσκει ένα γραμμικό μοντέλο παλινδρόμησης προβάλλοντας τις επεξηγηματικές μεταβλητές και την απόκριση σ'ένα καινούργιο χώρο. Με την PLS βρίσκουμε τις θεμελιώδεις σχέσεις μεταξύ των πινάκων X και Y, δηλαδή χρησιμοποιεί την προσέγγιση της κρυμμένης μεταβλητής για να μοντελοποιήσει τη συνδιακύμανση στους δυο χώρους. Το PLS μοντέλο προσπαθεί να βρει την πολυδιάστατη κατεύθυνση στο χώρο X που εξηγεί τη μέγιστη πολυδιάστατη κατεύθυνση της διακύμανσης στο χώρο Y [2].

$$X = TP^T + E$$

$$Y = UQ^T + F$$

όπου X είναι ο πίνακας $n \times m$ των επεξηγηματικών μεταβλητών, Y το διάνυσμα της απόκρισης, οι T και U είναι πίνακες διάστασης $n \times l$ και είναι προβολές του X και του Y αντίστοιχα. Οι $P_{m\times l}$ και $Q_{p\times l}$ είναι οι ορθογώνιοι πίνακες φορτίων, ενώ οι E και F είναι οι πίνακες των (ανεξάρτητων και ισόνομων Κανονικών) σφαλμάτων. Οι παραγοντοποιήσεις των X και Y γίνονται με στόχο τη μεγιστοποίηση της συνδιακύμανσης των T και U.

Αρχικά κατασκευάζονται οι κρυμμένες συνιστώσες μέσω της PLS παλινδρόμησης αντιμετωπίζοντας την κατηγορική απόκριση ως μια συνεχή μεταβλητή. Για τη δίτιμη απόκριση χρησιμοποιείται η $0-1$ ψευδομεταβλητή. Έστω $Y_{n\times q}$ και $X_{n\times p}$ η στήλη της απόκρισης και ο πίνακας με τις μεταβλητές πρόβλεψης αντίστοιχα. Έστω $T_{n\times K}$ οι κρυμμένες συνιστώσες που διέπουν το Y και τον X σύμφωνα με την PLS παλινδρόμηση. Τότε το PLS μοντέλο δίνεται από το τύπο

$$Y = TQ^T + F$$

και

$$X = TP^T + E$$

όπου $P_{p\times K}$ και $Q_{q\times K}$ είναι οι συντελεστές/φορτία και $E_{n\times p}$ και $F_{n\times q}$ τα σφάλματα. Οι κρυμμένες συνιστώσες T ορίζονται ως $T = XW$, όπου $W_{p\times K}$ είναι K διανύσματα κατεύθυνσης ($1 \leq K \leq \min(n, p)$). Βασικό μέλημα της PLS είναι η εύρεση των διανυσμάτων κατεύθυνσης. Το k-οστό διάνυσμα κατεύθυνσης $\hat{w}_k$ υπολογίζεται με την επίλυση του ακόλουθου προβλήματος βελτιστοποίησης

$$\max_w w^T M w$$

με $w^T w = 1$ και

$$w^T S_{XX} \hat{w}_l = 0$$

για $l = 1, \ldots, k-1$ όπου $M = X^T Y Y^T X$ και S_{XX} να είναι ο δειγματικός πίνακας συνδιακύμανσης των μεταβλητών πρόβλεψης [2].

Για την PLS η παραπάνω συνάρτηση μπορεί να ερμηνευτεί ως ακολούθως

$$\max_w cor^2(Y, Xw) var(Xw)$$

Αλλά αντί του προβλήματος βελτιστοποίησης τελικά επιλύεται το ακόλουθο πρόβλημα ελαχιστοποίησης ως προς τις ποσότητες w και c:

$$-k w^T M w + (1-k)(c-w)^T M (c-w) + \lambda_1 \|c\|_1 + \lambda_2 \|c\|_2$$

όπου $w^T w = 1$ και $M = X^T Y Y^T X$. Αυτή η μορφή επιβάλλει την L_1 ποινή σ'ένα αναπληρωματικό διάνυσμα κατεύθυνσης c και όχι στο αρχικό διάνυσμα κατεύθυνσης w, κρατώντας παράλληλα τα w και c κοντά μεταξύ τους. Η ποινή L_2 προλαμβάνει την πιθανή ιδιομορφία του πίνακα M [2].

Εαν η απόκριση είναι μονομεταβλητή, τότε το διάνυσμα που ελαχιστοποιεί την ως άνω σχέση είναι το

$$\hat{c} = (|Z| - \lambda_1/2)_+ sign(Z) = (|Z|)_+ sign(Z)$$

όπου $Z = \frac{X^T Y}{\|X^T Y\|}$, $(x)_+ = \max(0, x)$, $0 \leq k \leq 0.5$ και $\lambda_2 = \infty$.

Μετά την κατασκευή των κρυμμένων συνιστωσών, προσαρμόζουμε έναν ταξινομητή. Στο τελευταίο βήμα της PLS-DA έχουμε τη δυνατότητα να επιλέξουμε ανάμεσα σε πολλές μεθόδους ταξινόμησης γιατί ο αριθμός των κρυμμένων συνιστωσών K είναι συνήθως πολύ μικρότερος απ'το μέγεθος του δείγματος n. Οι συνήθεις επιλογές είναι γραμμικοί ταξινομητές οι οποίοι ερμηνεύονται ευκολότερα, όπως η Γραμμική Διαχωριστική Ανάλυση (LDA) και η Λογιστική παλινδρόμηση.

Έστω ότι με $\hat{\beta}^{LC}$ συμβολίζουμε τις εκτιμήσεις των συντελεστών των κρυμμένων συνιστωσών κάποιου γραμμικού ταξινομητή. Τότε οι εκτιμήσεις των συντελεστών των μεταβλητών πρόβλεψης X υπολογίζονται από τη σχέση $\hat{\beta} = W\hat{\beta}^{LC}$ επειδή $T\hat{\beta}^{LC} = XW\hat{\beta}^{LC} = X\hat{\beta}$.

Η PLS-DA στην R

Το λογισμικό που χρησιμοποιείται είναι το πακέτο DiscriMiner. Το σύνολο δεδομένων processed.cleveland.data[1] περιείχε αρχικά 76 μεταβλητές ενώ το επεξεργασμένο με το οποίο δουλεύουμε περιέχει τις εξής 13,

[1]Είναι διαθέσιμο στην ιστοσελίδα http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data του τμήματος Μηχανικής Μάθησης του πανεπιστημίου Irvine. Δημιουργήθηκε από τον Robert Detrano, M.D., Ph.D του ιδρύματος Cleveland Clinic Foundation και η δωρεά του στο τμήμα έγινε από τον David W. Aha.

X_1 Ηλικία σε έτη (age)

X_2 Φύλο (sex)

X_3 Τύπος πόνου στο στήθος (1: τυπική στηθάγχη, 2: ατυπική στηθάγχη, 3: άλλος πόνος, 4: ασυμπτωματικός) (cp)

X_4 Πίεση του ασθενούς κατά την εισαγωγή του στο νοσοκομείο (trestbps)

X_5 Τιμή της χοληστερίνης (chol)

X_6 Τιμή του σακχάρου στο αίμα (1: fasting blood sugar$>$120 mg/dl, 0: fasting blood sugar$\leq$120 mg/dl) (fbs)

X_7 Ηλεκτροκαρδιογραφικά αποτελέσματα (0: κανονικά, 1: εμφάνιση ανωμαλιών, 2: πιθανή υπερτροφία αριστερής κοιλίας) (restecg)

X_8 Μέγιστος καρδιακός ρυθμός (thalach)

X_9 Στηθάγχη που προκλήθηκε λόγω άσκησης (1: ναι, 0: όχι) (exang)

X_{10} Κατάσπαση του τμήματος ST στο ηλεκτροκαρδιογράφημα που προκλήθηκε κατά την άσκηση (oldpeak)

X_{11} Κλίση του τμήματος ST κατά την άσκηση (1: κλίση προς τα επάνω, 2: επίπεδη, 3: κλίση προς τα κάτω) (slope)

X_{12} Πλήθος βασικών αγγείων που χρωματίστηκαν κατά την ακτινοσκόπηση, λαμβάνει τιμές από 0 έως 3 (ca)

X_{13} Αποτελέσματα σαρωτή θαλλίου (3: κανονικά, 6: μόνιμο πρόβλημα, 7: αναστρέψιμο πρόβλημα) (thal)

Η απόκριση Y δίνει τη διάγνωση του ασθενούς. Οι τιμές 1, 2, 3 και 4 δηλώνουν την παρουσία καρδιακού νοσήματος ενώ η τιμή 0 δηλώνει την απουσία του. Οι παρατηρήσεις είναι 303 ενώ δεν είναι διαθέσιμες συνολικά 6 τιμές (missing values). Οι ελλιπείς τιμές αντικαθίστανται με τιμές που προκύπτουν από τη γραμμική παρεμβολή κάθε στήλης ξεχωριστά με τη συνάρτηση na.approx() του πακέτου zoo της R. Με x συμβολίζουμε το πλαίσιο δεδομένων των μεταβλητών πρόβλεψης και με y το διάνυσμα των $0-1$ αποκρίσεων. Ακόμη ζητήθηκε να γίνει η διαμέριση του συνόλου δεδομένων, ώστε το 75% εξ'αυτών να είναι τα δεδομένα εκπαίδευσης και το υπόλοιπο 25% να είναι τα δεδομένα ελέγχου. Καθώς οι παρατηρήσεις δεν εμφάνιζαν κάποια διάταξη επιλέχθηκαν οι πρώτες 227 ως training δεδομένα και οι τελευταίες 76 ως test δεδομένα.

```
modelplsDA <- plsDA(x, y, autosel = FALSE, comps = 2, validation = "learntest",
learn = c(1:227), test = c(228:303))

$functions
```

```
                0          1
INTERCEPT    1.580919  -0.580919
V1          -0.001510   0.001510
V2          -0.156573   0.156573
V3          -0.105352   0.105352
V4          -0.001150   0.001150
V5          -0.000381   0.000381
V6           0.083246  -0.083246
V7          -0.034208   0.034208
V8           0.002179  -0.002179
V9          -0.163540   0.163540
V10         -0.042305   0.042305
V11         -0.057624   0.057624
V12         -0.101825   0.101825
V13         -0.061126   0.061126
```

Ο παραπάνω πίνακας περιέχει τους συντελεστές των διαχωριστικών συναρτήσεων. Συγκεκριμένα η διαχωριστική συνάρτηση του υγιούς πληθυσμού είναι η

$$\begin{aligned} f_0 &= 1.58 - 0.002X_1 - 0.16X_2 - 0.105X_3 - 0.001X_4 - 0.0004X_5 + 0.08X_6 - 0.034X_7 \\ &+ 0.0022X_8 - 0.164X_9 - 0.04X_{10} - 0.058X_{11} - 0.102X_{12} - 0.0611X_{13} \end{aligned}$$

ενώ η αντίστοιχη συνάρτηση για τον πληθυσμό που νοσεί είναι η

$$\begin{aligned} f_1 &= -0.581 + 0.0015X_1 + 0.157X_2 + 0.105X_3 + 0.001X_4 + 0.0004X_5 - 0.083X_6 \\ &+ 0.034X_7 - 0.0022X_8 + 0.164X_9 + 0.042X_{10} + 0.058X_{11} + 0.102X_{12} + 0.061X_{13} \end{aligned}$$

Κάθε καινούργια παρατήρηση $x = (x_1, x_2, \ldots, x_{13})$ κατατάσσεται στην ομάδα όπου παρατηρείται το μέγιστο σκορ. Δηλαδή αν $f_0(x) > f_1(x)$ τότε η παρατήρηση προέρχεται απ'τον υγιή πληθυσμό, ενώ αν $f_0(x) < f_1(x)$ τότε προέρχεται απ'τον πάσχοντα πληθυσμό. Επομένως αν θέσουμε $Z = f_0 - f_1$, τότε για $Z > 0$ κατατάσσεται μια παρατήρηση ως υγιές άτομο, διαφορετικά κατατάσσεται ως πάσχων με τη συνάρτηση να λαμβάνει την ακόλουθη μορφή,

$$\begin{aligned} Z &= f_0 - f_1 \\ &= 2.161 - 0.0035X_1 - 0.317X_2 - 0.21X_3 - 0.002X_4 - 0.0008X_5 + 0.163X_6 - 0.068X_7 \\ &+ 0.0044X_8 - 0.328X_9 - 0.082X_{10} - 0.116X_{11} - 0.204X_{12} - 0.1221X_{13} \end{aligned}$$

Ακολουθούν οι υπολογισμοί των σκορ για τις πρώτες 6 test παρατηρήσεις,

```
$scores
             0          1
228  0.9153275  0.08467247
229  0.3193418  0.68065824
230  0.4023203  0.59767971
231  0.9522508  0.04774917
232  0.3596312  0.64036881
233  0.4676467  0.53235328
...
```

οι οποίοι δίνουν τις εξής προβλέψεις

```
$classification
[1] 0 1 1 0 1 1
Levels: 0 1

$confusion
        predicted
original   0   1
       0  34   3
       1   9  30
```

Για τα 76 άτομα του test συνόλου δεδομένων ισχύει ότι το μοντέλο PLS-DA κατέταξε σωστά 34 άτομα ως υγιή ενώ ήταν υγιή και 30 άτομα ως πάσχοντα ενώ όντως έπασχαν από το καρδιακό νόσημα. Αντίθετα κατέταξε λανθασμένα 3 άτομα ως πάσχοντα ενώ ήταν υγιή και άλλα 9 ως υγιή ενώ στην πραγματικότητα έπασχαν.

```
$error_rate
[1] 0.1578947
```

Επομένως ο ρυθμός λανθασμένης ταξινόμησης (**misclassification rate**) των **Test** δεδομένων ισούται με 0.1578947.

```
modelplsDA_notrain<-plsDA(x, y, autosel = FALSE, comps = 2, validation = NULL,
learn = NULL, test = NULL)

$error_rate
[1] 0.1485149
```

Όταν δε λαμβάνεται υπόψιν η διαμέριση σε δεδομένα εκπαίδευσης και ελέγχου, ο ρυθμός λανθασμένης ταξινόμησης της **Cross Validation** ισούται με 0.1485149. Ακολουθείται η leave-one-out cross validation η οποία είναι η K-fold Cross Validation με το K να ισούται με το πλήθος των παρατηρήσεων ($K = 303$). Ακολουθούν οι συντελεστές προσδιορισμού R^2 των συνιστωσών t_1 και t_2,

```
modelplsDA$R2
         R2X    R2Xcum        R2Y    R2Ycum
t1 0.2354895 0.2354895 0.52989046 0.5298905
t2 0.0977754 0.3332649 0.02915521 0.5590457
```

Για κάθε μεταβλητή υπολογίζεται το στατιστικό VIP (Variable Importance in the Projection) ,

$$VIP_{jk} = \sqrt{\frac{n \sum_{l=1}^{k} {W_{jl}}^{*2} \times SS_l(Y)}{\sum_{l=1}^{k} SS_l(Y)}}$$

για $1 \leq j \leq 13$, $1 \leq k \leq 2$, $n = 13$ και ${W_{jk}}^*$ είναι το j-οστό στοιχείο του ${W_{(k)}}^*$, όπου ${W_{(k)}}^*$ είναι η k-οστή στήλη του πίνακα των βαρών των μεταβλητών X, $W_{13\times 2}$. Με W^* συμβολίζεται η ποσότητα $W(P^T W)^{-1}$ όπου P είναι ο πίνακας των φορτίων και είναι διάστασης 13×2.

```
modelplsDA$VIP
            t1        t2
V1  0.75120522 0.8063949
V2  0.72705229 0.7670300
V3  1.23389756 1.2293427
V4  0.51547564 0.5145048
V5  0.36389032 0.3550815
V6  0.05970085 0.2865296
V7  0.50010343 0.4883022
V8  1.22657387 1.2258124
V9  1.23757429 1.2050251
V10 1.16833289 1.1610502
V11 1.04387993 1.0573083
V12 1.32770407 1.3068000
V13 1.57455544 1.5525483
```

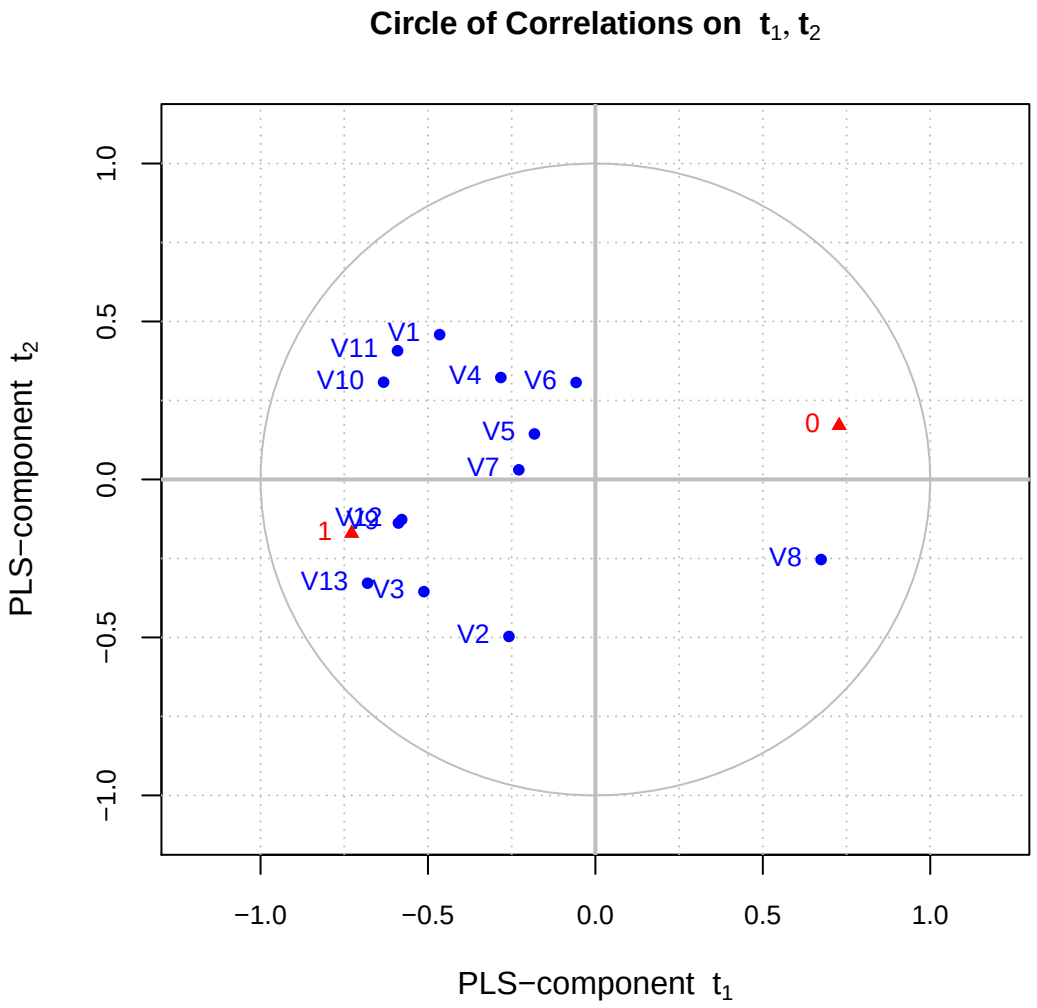
Ο πίνακας με τους συντελεστές γραμμικής συσχέτισης μεταξύ συνιστωσών και μεταβλητών ή διαφορετικά ο πίνακας των **φορτίων (loadings)**,

```
modelplsDA$comp_vars
             t1          t2
V1  -0.46514877  0.45840324
V2  -0.25826046 -0.49691782
V3  -0.51219045 -0.35510425
V4  -0.28237797  0.32265979
V5  -0.18208422  0.14403943
V6  -0.05760088  0.30693589
V7  -0.22871485  0.03056422
V8   0.67405261 -0.25353238
V9  -0.58832231 -0.13802044
V10 -0.63259006  0.30781595
V11 -0.59090067  0.40730874
V12 -0.57848640 -0.12695343
V13 -0.68043463 -0.32836451
```

Σημειώνεται ότι το άθροισμα των τετραγώνων των συντελεστών συσχέτισης (μεταξύ μιας μεταβλητής και για τις 2 συνιστώσες) ισούται με 1. Κατά συνέπεια τα τετράγωνα των φορτίων ερμηνεύονται ως η αναλογία της μεταβλητότητας των $V_1, \ldots, V_{13}$ που εξηγείται από τις συνιστώσες.

Οι συντελεστές γραμμικής συσχέτισης μεταξύ συνιστωσών και ομάδων ($Y = 0, 1$)

```
modelplsDA$comp_group
          t1        t2
0  0.7279358  0.170749
1 -0.7279358 -0.170749
```



Σχήμα 6.4:

Το γράφημα απεικονίζει τις συσχετίσεις/φορτία των μεταβλητών με τις συνιστώσες t_1, t_2. Οι μεταβλητές απεικονίζονται ως σημεία του χώρου των συνιστωσών, με συντεταγμένες τα φορτία τους. Το άθροισμα των τετραγώνων των φορτίων για κάθε μεταβλητή ισούται με 1. Κατά συνέπεια όταν τα δεδομένα εκπροσωπούνται τέλεια από μόνο δυο συνιστώσες, το άθροισμα των τετραγώνων των φορτίων ισούται με 1 και άρα σ'αυτήν την περίπτωση τα φορτία θα εμφανίζονταν πάνω στον κύκλο των συσχετίσεων. Όταν απαιτείται να υπάρχουν στην ανάλυση περισσότερες από δυο συνιστώσες, οι μεταβλητές θα εμφανίζονταν εντός του κύκλου των συσχετίσεων. Μεταβλητές που προσεγγίζουν αρκετά τον κύκλο είναι πιο σημαντικές στο να ερμηνεύσουν τις συνιστώσες από ότι οι υπόλοιπες. Αντίθετα, μεταβλητές που προσεγγίζουν το κέντρο του κύκλου είναι λιγότερο σημαντικές για τις δυο συνιστώσες.

6.2 Μέθοδοι Adaptive Boosting

Ο αλγόριθμος AdaBoost (**Ada**ptive **Boost**ing) δημιουργεί ένα σύνολο αδύναμων βασικών ταξινομητών οι οποίοι συνδυάζονται για να δώσουν έναν συνολικά ισχυρό ταξινομητή. Οι πιο δημοφιλείς αδύναμοι βασικοί μηχανισμοί εκμάθησης είναι τα δέντρα ταξινόμησης ή παλινδρόμησης. Σε κάθε βήμα της διαδικασίας, ο AdaBoost επιχειρεί να βρει ένα βέλτιστο ταξινομητή με βάση την τρέχουσα κατανομή των βαρών των παρατηρήσεων. Αν μια παρατήρηση ταξινομηθεί λανθασμένα χρησιμοποιώντας τη συγκεκριμένη κατανομή βαρών, τότε η παρατήρηση θα λάβει μεγαλύτερο βάρος στην επόμενη επανάληψη. Αντίθετα, οι σωστά ταξινομημένες - υπό την ίδια κατανομή βαρών - παρατηρήσεις θα λάβουν μικρότερα βάρη στην επόμενη επανάληψη. Στο τελικό μοντέλο, στους ταξινομητές που προβλέπουν με ακρίβεια τα δεδομένα εκπαίδευσης ανατίθενται μεγαλύτερα βάρη, ενώ σε εκείνους που δίνουν φτωχές προβλέψεις ανατίθενται μικρότερα βάρη. Έτσι ο AdaBoost χρησιμοποιεί μια ακολουθία απλών ταξινομητών με βάρη, με τον καθένα να «μαθαίνει» και μια διαφορετική πτυχή των δεδομένων, ώστε να προκύψει ένας τελικός ταξινομητής ο οποίος είναι καλύτερος (σε όρους σφάλματος λανθασμένης ταξινόμησης) με υψηλή πιθανότητα από οποιονδήποτε άλλον.

Όπως αναφέρθηκε σε προηγούμενη ενότητα του παρόντος, υπάρχουν παραλλαγές του βασικού **Discrete AdaBoost** αλγορίθμου ο οποίος αναφέρεται και ως AdaBoost.M1. Το 2000 ο Friedman συνέδεσε τον AdaBoost.M1 με έννοιες της Στατιστικής όπως είναι οι συναρτήσεις απώλειας, η αθροιστική μοντελοποίηση και η λογιστική παλινδρόμηση. Συγκεκριμένα έδειξε ότι ο AdaBoost.M1 προσαρμόζει ένα προς τα εμπρός και κατά βήματα αθροιστικό μοντέλο λογιστικής παλινδρόμησης το οποίο ελαχιστοποιεί τη μέση τιμή της εκθετικής συνάρτησης απώλειας $E\left(e^{-yF(x)}\right)$, με $F(x)$ να είναι ο ενισχυμένος (boosted) ταξινομητής [11].

Ενώ έχει αποδειχθεί εμπειρικά ότι ο AdaBoost βελτιώνει την ακρίβεια της ταξινόμησης, παράγει σε κάθε βήμα ως αποτέλεσμα την προβλεπόμενη τάξη. Αυτό μπορεί να εμποδίσει τον αλγόριθμο από το να βρει ένα βέλτιστο μοντέλο ταξινόμησης. Για να ξεπεραστεί αυτό το εμπόδιο έχει προταθεί μια παραλλαγή του κλασικού AdaBoost, ο αλγόριθμος **Real AdaBoost**. Στον Real AdaBoost η εκτίμηση της πιθανότητας της τάξης μετατρέπεται με τη χρήση του λόγου half-log στην κλίμακα των πραγματικών αριθμών. Κατόπιν αυτή η τιμή χρησιμοποιείται προκειμένου να φανεί η συνεισφορά της παρατήρησης στο τελικό μοντέλο. Επιπλέον αναβαθμίζονται τα βάρη των παρατηρήσεων των επακόλουθων επαναλήψεων σύμφωνα με την Εκθετική συνάρτηση απώλειας του AdaBoost. Όπως και ο Discrete AdaBoost έτσι και ο Real AdaBoost αλγόριθμος επιχειρεί να ελαχιστοποιήσει την ποσότητα $E\left(e^{-yF(x)}\right)$. Αυτές οι τροποποιήσεις καθιστούν τον Real AdaBoost πιο αποτελεσματικό στην εύρεση ενός βέλτιστου μοντέλου ταξινόμησης σε σύγκριση με τον κλασικό Discrete. Εκτός από τον Real AdaBoost έχει προταθεί από τον Friedman μια περαιτέρω επέκταση, ο **Gentle AdaBoost**, ο οποίος ελαχιστοποιεί την Εκθετική συνάρτηση απώλειας του AdaBoost μέσω μιας ακολουθίας Newton βημάτων. Παρόλο που οι Real και Gentle αλγόριθμοι βελτιστοποιούν την ίδια συνάρτηση απώλειας και δίνουν τα ίδια αποτελέσματα όταν εφαρμόζονται στο ίδιο σύνολο δεδομένων, ο Gentle αλγόριθμος υπερτερεί από τη σκοπιά της αριθμητικής ανάλυσης αφού δε βασίζεται στον half-log λόγο [11].

Ο Discrete AdaBoost αλγόριθμος

INPUT	$Z = \{z_1, z_2, \ldots, z_N\}$, όπου $z_i = (x_i, y_i)$ τα δεδομένα εκπαίδευσης
	M, ο μέγιστος αριθμός ταξινομητών
OUTPUT	$H(x)$, ο ταξινομητής που ταιριάζει στα δεδομένα εκπαίδευσης

1. Αρχικές τιμές βαρών $w_i = \frac{1}{N}$, $i \in \{1, \ldots, N\}$
2. Για $m = 1, \ldots, M$
   (α') Προσαρμογή ταξινομητή $H_m(x)$ στα δεδομένα εκπαίδευσης με βάρη τα w_i
   (β') Έστω $err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq H_m(x_i))}{\sum_{i=1}^{N} w_i}$
   (γ') Υπολογισμός $a_m = 0.5 \log\left(\frac{1-err_m}{err_m}\right)$
   (δ') $w_i \exp\left(-a_m I(y_i \neq H_m(x_i))\right) \rightarrow w_i$, επανακανονικοποίηση $\sum_{i=1}^{N} w_i = 1$
3. OUTPUT $H(x) = sign\left(\sum_{m=1}^{M} a_m H_m(x)\right)$

Ο αλγόριθμος σταματά όταν $m = M$ ή όταν $err_m > 0.5$ διότι η δεύτερη συνθήκη σημαίνει πως είναι αδύνατο να χτιστεί ένα καλύτερο σύνολο με τη χρήση αυτών των αδύναμων ταξινομητών, ανεξάρτητα από την αύξηση στο πλήθος τους. Έτσι το M είναι ο μέγιστος αριθμός ταξινομητών που πρέπει να περιληφθούν στο σύνολο εκμάθησης.

Ο Real AdaBoost αλγόριθμος

Η Real παραλλαγή του αλγορίθμου βελτιστοποιεί τη συνάρτηση κόστους

$$E\left[\exp(-y(H(x) + H_m(x))\right]$$

σε σχέση με το $H_m(x)$. Η ονομασία του εξηγείται από το γεγονός ότι οι ταξινομητές παράγουν μια τιμή που ανήκει στο $\Re$. Η τιμή αυτή είναι η πιθανότητα μια δεδομένη παρατήρηση ν'ανήκει σε κάποια τάξη, λαμβάνοντας υπόψιν την τρέχουσα κατανομή των βαρών των δεδομένων εκπαίδευσης. Επομένως ο Real AdaBoost μπορεί να θεωρηθεί ως μια γενίκευση του Discrete αλγορίθμου.

INPUT	$Z = \{z_1, z_2, \ldots, z_N\}$, όπου $z_i = (x_i, y_i)$ τα δεδομένα εκπαίδευσης
	M, ο μέγιστος αριθμός ταξινομητών
OUTPUT	$H(x)$, ο ταξινομητής που ταιριάζει στα δεδομένα εκπαίδευσης

1. Αρχικές τιμές βαρών $w_i = \frac{1}{N}$, $i \in \{1, \ldots, N\}$
2. Για $m = 1, \ldots, M$
   (α') Προσαρμογή στα δεδομένα εκπαίδευσης της εκτίμησης για την πιθανότητα της τάξης $p_m(x) = \widehat{P}_w(y = 1|x) \in [0, 1]$ με βάρη τα w_i
   (β') Θέστε $H_m = \frac{1}{2} \log\left(\frac{1-p_m(x)}{p_m(x)}\right) \in \Re$
   (γ') Θέστε $w_i \exp\left(-y_i H_m(x_i)\right) \rightarrow w_i$, επανακανονικοποίηση $\sum_{i=1}^{N} w_i = 1$
3. OUTPUT $H(x) = sign\left(\sum_{m=1}^{M} H_m(x)\right)$

Στα βήματα 2(α) και 2(β) υπολογίζεται η πιθανότητα μια δεδομένη παρατήρηση να ανήκει σε μια τάξη.

Ο Gentle AdaBoost αλγόριθμος

Ο Gentle αλγόριθμος δεν προσαρμόζει έναν εκτιμητή της πιθανότητας μιας τάξης αλλά χρησιμοποιεί την παλινδρόμηση ελαχίστων τετραγώνων με βάρη προκειμένου να ελαχιστοποιήσει τη συνάρτηση

$$E\left[\exp(-yH(x))\right]$$

INPUT	$Z = \{z_1, z_2, \ldots, z_N\}$, όπου $z_i = (x_i, y_i)$ τα δεδομένα εκπαίδευσης M, ο μέγιστος αριθμός ταξινομητών
OUTPUT	$H(x)$, ο ταξινομητής που ταιριάζει στα δεδομένα εκπαίδευσης

1. Αρχικές τιμές βαρών $w_i = \frac{1}{N}$, $i \in \{1, \ldots, N\}$
2. Για $m = 1, \ldots, M$
   (α') Εκπαίδευση του $H_m(x)$ μέσω παλινδρόμησης ελαχίστων τετραγώνων των y_i στα x_i με βάρη τα w_i
   (β') Αναβάθμιση $H(x) = H(x) + H_m(x)$
   (γ') Αναβάθμιση $w_i \exp\left(-y_i H_m(x_i)\right) \rightarrow w_i$, επανακανονικοποίηση $\sum_{i=1}^{N} w_i = 1$
3. OUTPUT $H(x) = sign\left(\sum_{m=1}^{M} H_m(x)\right)$

Ο Stochastic Gradient Boosting αλγόριθμος

Η μέθοδος του Stochastic Gradient Boosting ενσωματώνει έναν τυχαίο μηχανισμό σε κάθε βήμα του boosting, με αποτέλεσμα να βελτιώνεται η απόδοση και η ταχύτητα στη δημιουργία του συνόλου. Βασίζεται στην αναζήτηση του ανάδελτα κατάβασης (Gradient Descent) για τη βελτιστοποίηση της συνάρτησης απώλειας, προκειμένου να καθοριστούν τόσο τα βάρη όσο και ο μηχανισμός εκμάθησης (learner) σε κάθε επανάληψη. Στην SGB μέθοδο, σε κάθε επανάληψη, ακολουθείται η στρατηγική των τυχαίων μεταθέσεων η οποία δίνει ένα καλύτερο σύνολο εκπαίδευσης. Ο SGB αλγόριθμος βασίζεται στην παράμετρο $\nu \in [0, 1]$ η οποία καλείται **ρυθμός εκμάθησης (learning rate)**. Τα βήματα του αλγορίθμου είναι τα ακόλουθα,

1. Θέστε $F(x) = 0$
2. Για $m = 1, \ldots, M$
   (α') Θέστε $w_i = -\frac{\delta L(y,g)}{\delta g}|_{g=F(x)}$
   (β') Προσαρμόστε $y = \eta(h_m(x))$ ως βασικό ταξινομητή με βάρη χρησιμοποιώντας τα $|w_i|$, με δεδομένα εκπαίδευσης π_m
   (γ') Υπολογίστε το βήμα αναζήτησης $a_m = \arg\min_a \sum_{i \in \pi_m} L\left(y_i, F(x) + a\eta(h_m(x_i))\right)$ (σε ορισμένες περιπτώσεις το βήμα παραλείπεται ή $a_m = 1$)
   (δ') Αναβαθμίστε $F(x) = F(x) + \nu a_m \eta(h_m(x))$

Ως συνάρτηση απώλειας μπορεί να χρησιμοποιηθεί είτε η Εκθετική, $L(y,f) = e^{-yf}$, είτε η Λογιστική, $L(y,f) = \log(1+e^{-yf})$. Η συνάρτηση η καθορίζει τον τύπο του boosting:

1. $\eta(x) = sign(x)$ στον Discrete αλγόριθμο
2. $\eta(x) = 0.5\log\left(\frac{x}{1-x}\right)$ στον Real αλγόριθμο
3. $\eta(x) = x$ στον Gentle αλγόριθμο

Στην **Εκθετική απώλεια** η λύση της μεθόδου αναζήτησης επί γραμμής (line search step solution) του τρίτου βήματος του SGB αλγορίθμου μπορεί να γραφτεί ως,

$$a_k = a_{k-1} - (\eta^T P(a_{k-1})\eta)^{-1}(y^T P(a_{k-1})\eta) \tag{6.1}$$

όπου $P(a_{k-1}) = diag(p_i(a_{k-1}))$, $p_i(a_{k-1}) = w_i e^{-ay_i\eta_i}$ και $w_i = e^{-y_i F(x_i)}$. Το τελικό $a_m = a_\infty$ χρησιμοποιείται για τον αλγόριθμο SGB.

Στη **Λογιστική συνάρτηση απώλειας** το τρίτο βήμα του αλγορίθμου μπορεί να γραφεί ως,

$$a_k = a_{k-1} - (\eta^T P(a_{k-1})(1-P(a_{k-1}))\eta)^{-1}(y^T P(a_{k-1})\eta) \tag{6.2}$$

όπου $P(a_{k-1}) = diag(p_i(a_{k-1}))$, $p_i(a_{k-1}) = \frac{w_i e^{-ay_i\eta_i}}{1+w_i e^{-ay_i\eta_i}}$ και $w_i = e^{-y_i F(x_i)}$.

Ο SGB αλγόριθμος προσαρμόζεται στις 3 παραλλαγές του AdaBoost ως εξής [11],

- **Discrete AdaBoost**: Στην εκθετική απώλεια θέτουμε

  $$L(y,g) = e^{-yg} \Rightarrow w_i = -y_i e^{-y_i F_i}$$

  και $\eta(x) = sign(x)$. Χρησιμοποιώντας την έκφραση (1.1) και με τη συγκεκριμένη τιμή της η, το πρόβλημα βελτιστοποίησης έχει λύση κλειστής μορφής η οποία είναι η $a_m = 0.5\log\left(\frac{1-err_m}{err_m}\right)$. Ουσιαστικά προσαρμόζεται ο αρχικός Discrete AdaBoost με τη στρατηγική της τυχαίας δειγματοληψίας σε κάθε επανάληψη. Στην περίπτωση της Λογιστικής συνάρτησης απώλειας βελτιστοποιείται η

  $$L(y,g) = \log\left(1+e^{-y_i g_i}\right) \Rightarrow w_i = \frac{-y_i e^{-y_i F(x_i)}}{1+e^{-y_i F(x_i)}}$$

  Τότε η λύση της (1.2) δεν είναι κλειστής μορφής.

- **Real AdaBoost**: Και στις δυο συναρτήσεις απώλειας $\eta(p) = \log\left(\frac{p}{1-p}\right)$, όπου $p \in [0,1]$ η εκτίμηση της πιθανότητας τάξης, με το βάρος να είναι ίδιο με την περίπτωση του Discrete αλγορίθμου.

- **Gentle AdaBoost**: Και στις δυο συναρτήσεις απώλειας $\eta(x) = x$. Απαιτείται απ'τον αλγόριθμο η προσαρμογή ενός μοντέλου παλινδρόμησης σε κάθε επανάληψη και όποτε $a_m = 1$ συμπίπτει με τον αρχικό Gentle αλγόριθμο. Όπως και στην περίπτωση του Real ο αλγόριθμος επιλύει άμεσα την αναζήτηση γραμμής.

Εφαρμογή στην R

Το λογισμικό που χρησιμοποιείται είναι το πακέτο ada, και το σύνολο δεδομένων είναι το γνωστό πλέον processed.cleveland.data. με τις 303 παρατηρήσεις, τις 13 επεξηγηματικές μεταβλητές και την απόκριση η οποία λαμβάνει την τιμή 0 όταν το άτομο είναι υγιές και την τιμή 1 στην περίπτωση που πάσχει από καρδιακό νόσημα.

Δέντρα ταξινόμησης ή παλινδρόμησης εφαρμόζονται, από τους boosting και τους SGB αλγορίθμους, ως βασικοί/αδύναμοι μηχανισμοί εκμάθησης (learners).

Για τον **Discrete AdaBoost** και με συνάρτηση απώλειας την *Εκθετική* έχουμε τα εξής[2],

```
model1<- ada(xlearn, ylearn, loss="exponential",
                      type="discrete",iter=50, nu=0.01, bag.frac=0.5,
                      model.coef=TRUE, verbose=TRUE)
model1<-addtest(model1,xtest,ytest)
```

Ο πίνακας xlearn διάστασης 227×13 περιέχει τα δεδομένα εκπαίδευσης και το διάνυσμα της δίτιμης απόκρισης είναι το ylearn. Ο πίνακας xtest διάστασης 76×13 και το αντίστοιχό του διάνυσμα απόκρισης ytest αποτελούν τα δεδομένα ελέγχου. Το πλήθος των επαναλήψεων ισούται με 50, η παράμετρος σμίκρυνσης ή **ρυθμός εκμάθησης** ν ισούται με 0.01 [5]. Η παράμετρος bag.frac αναφέρεται στο κλάσμα των παρατηρήσεων **out-of-bag** και επιτρέπει να χτίζεται κάθε δέντρο μέσω της στρατηγικής της τυχαίας μετάθεσης. Με το model.coef υπολογίζονται οι ποσότητες a_k (stageweights) της σχέσης (1.1).

```
Call:
ada(xlearn, y = ylearn, loss = "exponential", type = "discrete",
    iter = 50, nu = 0.01, bag.frac = 0.5, model.coef = TRUE,
    verbose = TRUE)

Loss: exponential Method: discrete   Iteration: 50

Final Confusion Matrix for Data:
          Final Prediction
True value   0   1
         0 119   8
         1  20  80

Train Error: 0.123

Out-Of-Bag Error:  0.123  iteration= 50

Additional Estimates of number of iterations:

train.err1 train.kap1 test.errs2 test.kaps2
        40         40         14         14
```

Ο ρυθμός λανθασμένης ταξινόμησης ισούται με 0.123 **(Misclassification Rate)**. Παρατηρούμε ότι δίνονται τρεις εκτιμήσεις για το πλήθος των επαναλήψεων. Θα

[2]Σημειώνεται ότι οι αλγόριθμοι στους οποίους γίνεται αναφορά στην παρούσα ενότητα αφορούν στην εκδοχή του Stochastic Gradient Boosting για ταξινόμηση.

μπορούσε να χρησιμοποιηθεί η Out-of-bag εκτίμηση για επαναλήψεις (μόνο που εδώ συμπίπτει με το 50), η εκτίμηση του training σφάλματος για 40 (ή η κάππα εκτίμηση σφάλματος για επίσης 40 επαναλήψεις). Τρέχοντας το μοντέλο για 40 επαναλήψεις προκύπτει ρυθμός λανθασμένης ταξινόμησης ίσος με 0.119.

Σχήμα 6.5: Οι ρυθμοί λανθασμένης ταξινόμησης (Misclassification Rates) για τα δεδομένα εκπαίδευσης και ελέγχου.

Παρατηρούμε ότι το σφάλμα εκπαίδευσης μειώνεται όσο αυξάνεται το πλήθος των επαναλήψεων. Επομένως φαίνεται πως το boosting είναι αποτελεσματικό στην εκμάθηση των χαρακτηριστικών των δεδομένων μας.

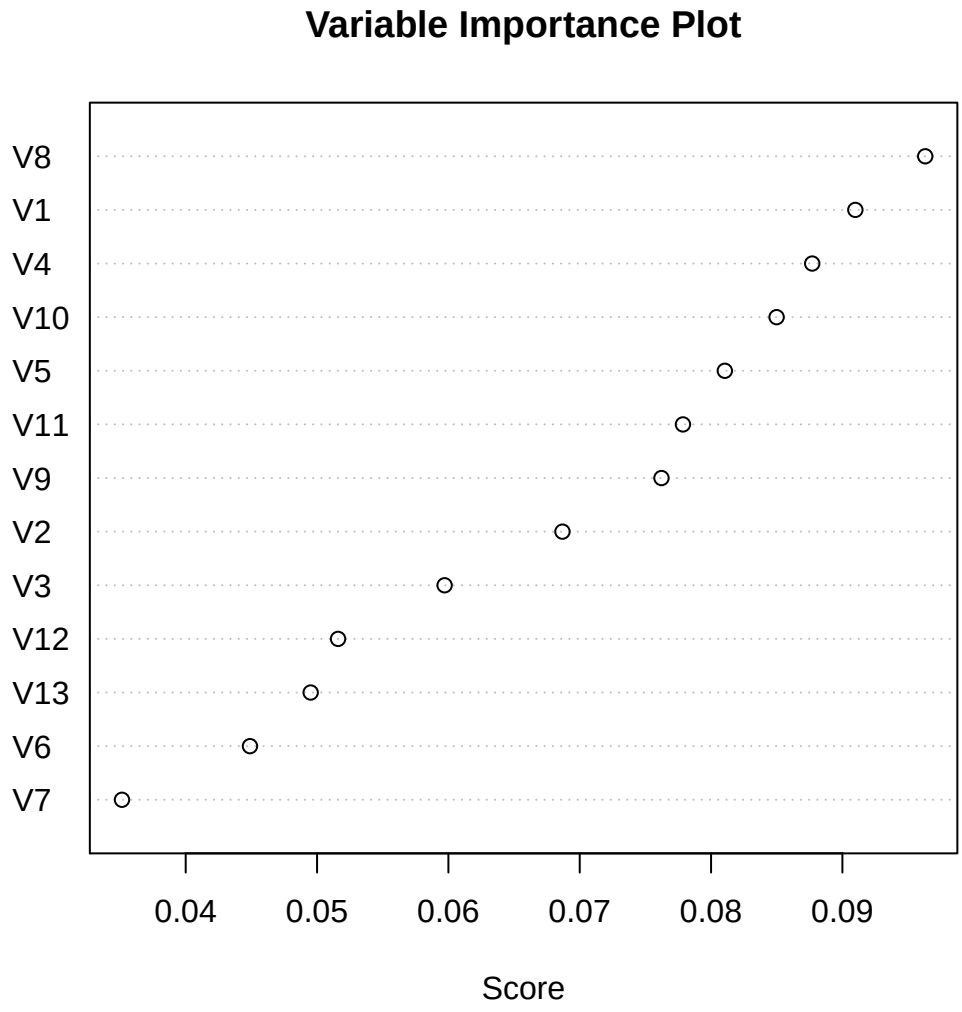
Για την κατάταξη καινούργιων παρατηρήσεων χρησιμοποιείται η συνάρτηση **predict** η οποία εκτός από την τιμή της απόκρισης, επιστρέφει και την εκτίμηση για την πιθανότητας της τάξης. Η τελευταία υπολογίζεται για οποιοδήποτε Boosting αλγόριθμο ως εξής,

$$\hat{P}(Y = 1|\boldsymbol{x}) = \frac{e^{2F(x)}}{1 + e^{2F(x)}}$$

```
varplot(model1)
vip<-varplot(model1,plot.it=FALSE,type="scores")
```

Ακολουθούν οι τιμές VIP σκορ για την αξιολόγηση της συνεισφοράς των μεταβλητών στο μοντέλο.

```
    V8     V1     V4    V10     V5    V11     V9     V2
0.0963 0.0910 0.0877 0.0850 0.0810 0.0779 0.0762 0.0687

  V3    V12    V13     V6     V7
0.0597 0.0516 0.0495 0.0449 0.0351
```



Σχήμα 6.6: Η γραφική αναπαράσταση των VIP σκορ των 13 μεταβλητών.

Η παρακάτω εντολή παρουσιάζει την απόδοση του boosted μοντέλου στα δεδομένα ελέγχου.

```
summary(model1,n.iter=14)
```

```
Call:
ada(xlearn, y = ylearn, loss = "exponential", type = "discrete",
    iter = 50, nu = 0.01, bag.frac = 0.5, model.coef = TRUE,
    verbose = TRUE)

Loss: exponential Method: discrete   Iteration: 14

Training Results

Accuracy: 0.855 Kappa: 0.704

Testing Results

Accuracy: 0.803 Kappa: 0.607
```

Το **στατιστικό Κάππα** είναι ένα μέτρο συμφωνίας μεταξύ της παρατηρούμενης και της προβλεπόμενης ταξινόμησης και ορίζεται ως εξής,

$$K = \frac{P(a) - P(e)}{1 - P(e)}$$

όπου $P(a)$ είναι η σχετική παρατηρούμενη συμφωνία μεταξύ βαθμολογητών και $P(e)$ η πιθανότητα τυχαίας συμφωνίας, χρησιμοποιώντας τα δεδομένα που παρατηρήθηκαν στον υπολογισμό των πιθανοτήτων κάθε άτομο να έχει επιλέξει τυχαία κάποια τάξη. Όταν $K = 1$ τότε σημαίνει πως υπάρχει απόλυτη συμφωνία μεταξύ των βαθμολογητών, αντίθετα όταν $K = 0$ η μόνη συμφωνία που υπάρχει μεταξύ τους είναι εκείνη που αποδίδεται στην τύχη.

Για τον **Real AdaBoost** και με συνάρτηση απώλειας την *Εκθετική* έχουμε,

```
Call:
ada(xlearn, y = ylearn, loss = "exponential", type = "real",
    iter = 50, nu = 0.01, bag.frac = 0.5, model.coef = TRUE,
    verbose = TRUE)

Loss: exponential Method: real   Iteration: 50

Final Confusion Matrix for Data:
          Final Prediction
True value   0   1
         0 118   9
         1  19  81

Train Error: 0.123

Out-Of-Bag Error:  0.128  iteration= 15

Additional Estimates of number of iterations:

train.err1 train.kap1 test.errs2 test.kaps2
        38         38          3          4
```

Το **Misclassification Rate** για τα δεδομένα εκπαίδευσης δε μεταβλήθηκε αφού είναι ίσο με 0.123.

Σχήμα 6.7: Από τη 12η περίπου επανάληψη αρχίζει να σταθεροποιείται το training σφάλμα.

Η απόδοση του boosted μοντέλου στα δεδομένα ελέγχου,

```
Call:
ada(xlearn, y = ylearn, loss = "exponential", type = "real",
    iter = 50, nu = 0.01, bag.frac = 0.5, model.coef = TRUE,
    verbose = TRUE)

Loss: exponential Method: real   Iteration: 3

Training Results

Accuracy: 0.824 Kappa: 0.648

Testing Results

Accuracy: 0.776 Kappa: 0.553
```

Τέλος ο **Gentle AdaBoost** με συνάρτηση απώλειας την *Εκθετική* δίνει τα ακόλουθα αποτελέσματα,

```
Call:
ada(xlearn, y = ylearn, loss = "exponential", type = "gentle",
    iter = 50, nu = 0.01, bag.frac = 0.5, model.coef = TRUE,
    verbose = TRUE)

Loss: exponential Method: gentle   Iteration: 50

Final Confusion Matrix for Data:
          Final Prediction
True value   0   1
         0 120   7
         1  19  81

Train Error: 0.115

Out-Of-Bag Error:  0.115  iteration= 47

Additional Estimates of number of iterations:

train.err1 train.kap1 test.errs2 test.kaps2
        26         26         19         19

Loss: exponential Method: gentle   Iteration: 19

Training Results

Accuracy: 0.877 Kappa: 0.747

Testing Results

Accuracy: 0.789 Kappa: 0.581
```

Αποτελεί τελικά την καλύτερη επιλογή ανάμεσα στους AdaBoost αλγόριθμους με misclassification rate ίσο με 0.115.

Σχήμα 6.8: Μετά την 25η επανάληψη το training σφάλμα παρουσιάζει κάμψη.

6.3 Bagging στην ταξινόμηση

Για θέματα ταξινόμησης ο αλγόριθμος Bagging έχει την ακόλουθη μορφή,

INPUT $Z = \{z_1, z_2, \ldots, z_N\}$, όπου $z_i = (x_i, y_i)$ τα δεδομένα εκπαίδευσης
B, ο αριθμός διάφορων εκδοχών του συνόλου εκπαίδευσης από δειγματοληψία

OUTPUT $H(x)$, ο ταξινομητής που ταιριάζει στα δεδομένα εκπαίδευσης

1. Για $n = 1, \ldots, B$

   (α') Πάρτε με επανάθεση $L \leq N$ δείγματα εκ του συνόλου εκπαίδευσης Z, συμβολίζοντας το n-οστό δείγμα με Z^{*n}

   (β') Για κάθε δείγμα Z^{*n}, εκπαιδεύστε τον ταξινομητή H_n

2. Ο τελικός ταξινομητής προκύπτει ως μια ψήφος των H_n με $n = \{1, \ldots, B\}$

$$H(x) = sign\left(\sum_{n=1}^{B} H_n(x)\right)$$

Εφαρμογή στην R

Το λογισμικό που χρησιμοποιείται είναι το πακέτο ipred, και το σύνολο δεδομένων το cleveland.processed.data.

```
model1<- ipredbagg(yfactor, X=x, nbagg=25,
control=rpart.control(minsplit=2, cp=0, xval=0),coob=TRUE,
ns=length(yfactor), keepX = TRUE)
```

Με yfactor συμβολίζεται το διάνυσμα της δίτιμης απόκρισης σε μορφή παράγοντα, x είναι το πλαίσιο δεδομένων των επεξηγηματικών και ζητήθηκαν 25 Bootstrap δείγματα.

```
Bagging classification trees with 25 bootstrap replications
Out-of-bag estimate of misclassification error:  0.2079

model2<- ipredbagg(yfactor, X=x, nbagg=50,
control=rpart.control(minsplit=2, cp=0, xval=0),coob=TRUE,
ns=length(yfactor), keepX = TRUE)

Bagging classification trees with 50 bootstrap replications
Out-of-bag estimate of misclassification error:  0.2277
```

Βλέπουμε ότι το **OOB** σφάλμα για 25 δείγματα ισούται με 0.2079. Αντίθετα όταν ζητούνται 50 δείγματα η ίδια ποσότητα ισούται με 0.2277.

Συμπεράσματα

Συγκρίνοντας τους ρυθμούς λανθασμένης ταξινόμησης των μεθόδων PLS-DA, Adaptive Boosting και Bagging προκύπτει ότι ο **Gentle AdaBoost** είναι η καλύτερη επιλογή (Misclassification Rate 0.115). Οι τρεις παραλλαγές του AdaBoost αλγορίθμου έδωσαν τα χαμηλότερα σφάλματα ταξινόμησης με τιμές από 0.115 έως 0.123.

Δεύτερη κατατάσσεται η PLS-DA με τον Cross Validated ρυθμό λανθασμένης ταξινόμησης να ισούται με 0.1485 και τον test ρυθμό να ισούται με 0.1579. Τέλος, το Bagging δίνει τιμή ίση με 0.2079.

Κεφάλαιο 7

Boosting και Παλινδρόμηση

Δύο τρόποι χρησιμοποιούνται για την προσαρμογή της μεθόδου boosting στα μοντέλα παλινδρόμησης. Ο πρώτος είναι το προς τα εμπρός και κατά στάδια αθροιστικό μοντέλο, το οποίο τροποποιεί τις τιμές στόχους προκειμένου να έχει αποτελεσματική προσαρμογή στα σφάλματα/υπόλοιπα. Ο δέυτερος τρόπος προέρχεται από την ήδη γνωστή ταξινόμηση, και συνίσταται στην αλλαγή των βαρών των παρατηρήσεων ώστε να δωθεί έμφαση σ'εκείνες στις οποίες η παλινδρόμηση στα προηγούμενα στάδια της διαδικασίας προσαρμογής, είχε φτωχά αποτελέσματα. Επειδή η τελευταία μέθοδος είναι παραπλήσια με το boosting για ταξινόμηση, επικεντρωνόμαστε στην παρουσίαση του **προς τα εμπρός και κατά στάδια αθροιστικού μοντέλου**.

Στη γενική περίπτωση, ένα αθροιστικό μοντέλο εκφράζεται ως εξής,

$$F(x) = \beta_1 f_1(x) + \beta_2 f_2(x) + \ldots + \beta_T f_T(x) = \sum_{t=1}^{T} \beta_t f_t(x) \tag{7.1}$$

όπου $\beta_t, t = 1, 2, \ldots, T$ είναι οι συντελεστές επέκτασης, συνήθως $\beta_t = 1$ και οι συναρτήσεις $f_t(x)$ συχνά επιλέγονται να είναι απλές. Στόχος της boosting μεθόδου είναι η εύρεση μιας κρυμμένης συνάρτησης $F(x)$ η οποία ελαχιστοποιεί κάποια συνάρτηση απώλειας, όπως για παράδειγμα είναι η απώλεια του τετραγώνου του σφάλματος, μεταξύ της απόκρισης y και των εκτιμώμενων τιμών $F(x)$. Τις περισότερες φορές η κρυμμένη συνάρτηση δεν είναι γνωστή σε εμάς. Συνεπώς η μέθοδος επιχειρεί να βρει μια απλή συνάρτηση $f(x)$ σε κάθε επανάληψη,

$$\underset{f(x)}{min} \sum_{i=1}^{N} \|y_i - F_{t-1}(x_i) - f_t(x_i)\|^2 \tag{7.2}$$

όπου t είναι το πλήθος των τιμών $f(x)$ που προστίθενται. Έτσι κάθε απλή συνάρτηση ($f(x)$) μπορεί να παραχθεί με την προσαρμογή προηγούμενων υπολοίπων, με το αρχικό training σύνολο X.

Ακολουθεί ο Boosting αλγόριθμος [5] ο οποίος στηρίζεται στην **προς τα εμπρός και κατά στάδια αθροιστική** στρατηγική:

1. Πρώτον προσαρμόστε ένα βασικό μοντέλο παλινδρόμησης $f(X)$ στα training δεδομένα, εκπεφρασμένο ως $\widehat{y}_1 = f_1(X)$.Υπολογίστε το υπόλοιπο: $y_{res} = y - v_1 \widehat{y}_1$

Όπου $0 < \upsilon < 1$ και υ είναι μια παράμετρος σμίκρυνσης η οποία μπορεί να είναι μια σταθερά ή όχι. Η παράμετρος σμίκρυνσης μπορεί ν'αποτρέψει αποτελεσματικά προβλήματα που σχετίζονται με το overfitting. Σε κάθε βήμα εξάγεται μόνο το υ τμήμα των προσαρμοσμένων τιμών.

2. Για $t = 2, \ldots, T$ επαναλάβατε τα ακόλουθα βήματα:
(α) Προσαρμόστε το τρέχων υπόλοιπο $y_{res,t-1}$ με τη χρήση του βασικού μοντέλου παλινδρόμησης $\widehat{y}_t = f_t(X)$.
(β) Αναβαθμίστε το υπόλοιπο

$$y_{res,t} = y_{res,t-1} - \upsilon_t \widehat{y}_t$$

Στο βήμα αυτό, μόνο το υ τμήμα του προσαρμοσμένου $\widehat{y}_t$, χρησιμοποιείται ως πληροφορία για την παλινδρόμηση.

3. Η τελική πρόβλεψη είναι η εξής

$$y_{pre} = \upsilon_1 \widehat{y}_1 + \upsilon_2 \widehat{y}_2 + \ldots + \upsilon_{T-1} \widehat{y}_{T-1} + \upsilon_T \widehat{y}_T = \sum_{t=1}^{T} \upsilon_t f_t(X)$$

Όντας ένα αθροιστικό μοντέλο, η βασική ιδέα του boosting είναι η διαδοχική κατασκευή αθροιστικών μοντέλων παλινδρόμησης, προσαρμώζοντας ένα βασικό μοντέλο παλινδρόμησης στα τρέχοντα υπόλοιπα τα οποία δεν επεξηγούνται από προηγούμενα μοντέλα.

Τα **τρέχοντα υπόλοιπα** είναι είναι το ανάδελτα (gradient) τη συνάρτησης απώλειας (π.χ. μια συνάρτηση τετραγωνικού σφάλματος) ελαχιστοποιημένο στο τρέχων βήμα. Αρχικά προσαρμόζεται ένα βασικό μοντέλο παλινδρόμησης και υπολογίζονται τα αρχικά υπόλοιπα. Ως εκ τούτου, επανάληψη με επανάληψη, το boosting λαμβάνει υπόψη του μόνο τα τρέχοντα υπόλοιπα του y και πάντα προσαρμόζει τα τρέχοντα υπόλοιπα του y με τα αρχικά X. Έτσι προστίθενται διαδοχικά μια σειρά βασικών μοντέλων παλινδρόμησης τα οποία εφαρμόζονται εύκολα. Κάθε μοντέλο σμικρύνεται με βάση κάποια παράμετρο σμίκρυνσης υ, η οποία λαμβάνει θετικές τιμές μικρότερες του 1. Η παράμετρος υ δρα ως ένα βάρος για το τρέχων μοντέλο το οποίο αποτρέπει ένα πιθανό overfitting περιορίζοντας τη διαδικασία προσαρμογής. Κάθε φορά χρησιμοποιείται ολόκληρος ο πίνακας X για την εξαγωγή πληροφορίας συσχετισμένης με τα υπόλοιπα, ωστόσο η εν λόγω πληροφορία δεν χρησιμοποιείται κατά 100%. Μονάχα ένα υ τμήμα της λαμβάνεται υπόψη, με το υπόλοιπο $1 - \upsilon$ να επιστρέφεται προς αποφυγή του overfitting. Η διαδοχική προσθήκη επαναλαμβάνεται αποφέροντας πολλά βασικά μοντέλα παλινδρόμησης με βάρη. Μόλις η διαδικασία φτάσει στον προκαθορισμένο αριθμό επαναλήψεων T, τότε γίνεται η πρόβλεψη με βάση το test σύνολο δεδομένων. Η τελική πρόβλεψη είναι ο συνδυασμός των βασικών μοντέλων παλινδρομησης με βάρη.

Η παραπάνω μέθοδος είναι γνωστή ως **Gradient Boosting (GB)**. Προκειμένου να βελτιωθεί η ακρίβεια και η ταχύτητα της εκτέλεσης του GB ο Friedman ενσωμάτωσε το ανάλογο της τυχαιοποίησης στο bagging μέσα στη μέθοδο και έτσι προέκυψε το **Stochastic Gradient Boosting (SGB)**. Σε κάθε επανάληψη του SGB, ένα κλάσμα

η των training δεδομένων υπόκειται σε δειγματοληψία χωρίς επανάθεση. Αυτά τα τυχαία επιλεγμένα υπο-δείγματα χρησιμοποιούνται αντί για τα πλήρη δείγματα για την προσαρμογή του μηχανισμού εκμάθησης (learner), και για τον υπολογισμό της αναβάθμισης του μοντέλου κατά την τρέχουσα επανάληψη. Ο υπόλοιπος αλγόριθμος είναι ίδιος με τον GB αλγόριθμο. Σύμφωνα με έρευνες, μια τυπική τιμή για την παράμετρο η είναι $1/2$ για μεγάλο αριθμό N των training παρατηρήσεων. Η εφαρμογή της δειγματοληψίας όχι μόνο μειώνει τον υπολογιστικό χρόνο κατά το ίδιο κλάσμα η, αλλά δίνει τις περισσότερες φορές και ένα πιό ακριβές μοντέλο.

Εκτός από τη συνάρτηση απώλειας τετραγωνικού σφάλματος, μπορούν να εφαρμοστούν και άλλες ανθεκτικές (robust) συναρτήσεις απώλειας με τα αντίστοιχα ανάδελτά τους όπως φαίνεται στον ακόλουθο πίνακα,

Συνάρτηση απώλειας		Ανάδελτα
Απώλεια τετραγωνικού σφάλματος	$1/2[y_i - F(x_i)]^2$	$y_i - F(x_i)$
Απώλεια απόλυτου σφάλματος	$\lvert y_i - F(x_i)\rvert$	$sign[y_i - F(x_i)]$
Συνάρτηση Huber	$1/2[y_i - F(x_i)]^2$ για $\lvert y_i - F(x_i)\rvert \leq \delta$	$y_i - F(x_i)$ για $\lvert y_i - F(x_i)\rvert \leq \delta$
	$\lvert y_i - F(x_i)\rvert$ για $\lvert y_i - F(x_i)\rvert > \delta$	$\delta(y_i - F(x_i))$ για $\lvert y_i - F(x_i)\rvert > \delta$

$*\delta$ είναι το a-ποσοστιαίο σημείο του $\lvert y_i - F(x_i)\rvert$

Για την απώλεια τετραγωνικού σφάλματος, το αρνητικό ανάδελτα είναι απλά το υπόλοιπο. Για την απώλεια απόλυτου σφάλματος, το αρνητικό ανάδελτα είναι το *sign* του υπόλοιπου. Επομένως, τα *sign* των τρεχόντων υπολοίπων προσαρμόζονται στον πίνακα X στο τρέχων βήμα.

Η βελτιστοποίηση της παραμέτρου είναι πολύ σημαντική όσον αφορά στην επιλογή του κατάλληλου μοντέλου. Εκτός από τις παραμέτρους της βασικής συνάρτησης $f(x)$ (υπάρχουν διαφορετικές παράμετροι για κάθε βασική συνάρτηση), τόσο η παράμετρος σμίκρυνσης ν όσο και το πλήθος των επαναλήψεων T ελέγχουν το βαθμό προσαρμογής στο μοντέλο του boosting. Στο πρόβλημα της ταξινόμησης στον AdaBoost γίνεται επιλογή μόνο για την παράμετρο T. Η επιλογή των παραμέτρων παίζει έναν κρίσιμο ρόλο στην αποδοτικότητα της μεθόδου και γίνεται είτε με την τεχνική cross validation είτε με την τεχνική του σετ ανεξάρτητης επικύρωσης. Ο έλεγχος της συνεισφοράς του μηχανισμού εκμάθησης μέσω του ν μπορεί να βοηθήσει στη μείωση της επιρροής του θορύβου που προκαλείται από άλλους παράγοντες. Κάτι τέτοιο βελτιώνει την προσαρμογή και την προβλεπτική ακρίβεια. Η βέλτιστη τιμή ν που επιτυγχάνει την καλύτερη απόδοση είναι συνήθως μικρή, μικρότερη του 0.1. Στην πράξη η κατάλληλη τιμή για την παράμετρο ν μπορεί να εκτιμηθεί μέσω της cross validation. Ο έλεγχος της τιμής T μπορεί να ρυθμίσει σε ποιό βαθμό μπορεί να ελαχιστοποιηθεί η αναμενόμενη απώλεια στα training δεδομένα. Μεγάλη τιμή του T οδηγεί σε overfitting και υποβαθμίζει τη μελλοντική απόδοση του μοντέλου. Η καλύτερη τιμή για το T εκτιμάται επίσης με τη βοήθεια της cross validation. Ένας πρακτικός τρόπος καθορισμού του T είναι να ελέγξουμε το σφάλμα της πρόβλεψης ως συνάρτηση του T σ'ένα σετ δεδομένων επικύρωσης.

Εφαρμογή με δεδομένα NIR

Μεταξύ των πολλών περιοχών στις οποίες η μέθοδος του Boosting βρίσκει εφαρμογή είναι και η χημεία, με τη φασματοσκοπική μέθοδο εγγύς υπερύθρου (near infrared

spectroscopy). Η εν λόγω μέθοδος εκμεταλλέυεται την ακτινοβολία του ηλεκτρομαγνητικού φάσματος που βρίσκεται κοντά στις υπέρυθρες (από περίπου 800nm έως 2500 nm) και έχει φαρμακευτικές, διαγνωστικές και άλλες εφαρμογές.

Το σύνολο δεδομένων gasoline[1] της εφαρμογής αποτελείται από NIR φάσματα τα οποία βρίσκονται κοντά στην υπέρυθρη ακτινοβολία 60 παρατηρήσεων βενζίνης. Η απόκριση octane δίνει τον αριθμό οκτανίων και οι 401 ανεξάρτητες μεταβλητές είναι το NIR φάσμα. Η διαμέριση των 60 παρατηρήσεων σε 75% training (κατασκευή μοντέλου) και σε 25% testing (αξιολόγηση μοντέλου) δεδομένα, έγινε με την ακόλουθη συνάρτηση,

```
dataframe<- gasoline

splitdf <- function(dataframe, seed=NULL) {
    if (!is.null(seed)) set.seed(seed)
    index <- 1:nrow(dataframe)
    trainindex <- sample(index, length(index)*0.75)
    trainset <- dataframe[trainindex, ]
    testset <- dataframe[-trainindex, ]
    list(trainset=trainset,testset=testset)
}

splits <- splitdf(dataframe, seed=NULL)


str(splits)

lapply(splits,nrow)

lapply(splits,head)

training <- splits$trainset

testing <- splits$testset
```

Τα δεδομένα NIR εμφανίζουν υψηλή πολυσυγγραμικότητα επομένως μια μέθοδος που χρησιμοποιείται σε δεδομένα τέτοιας φύσεως είναι η **Partial Least Squares (PLS)**, η οποία μειώνει σημαντικά τον αρχικό αριθμό των μεταβλητών. Έτσι παραμένει στην ανάλυση ένας μικρός αριθμός γραμμικά ασυσχέτιστων μεταβλητών οι οποίες καλούνται λανθάνουσες ή αφανείς (latent). Η PLS είναι η μέθοδος αναφοράς και γίνεται η σύγκρισή της με τις μεθόδους **Stochastic Gradient Boosting** και **Bagging**.

7.1 Partial Least Squares Παλινδρόμηση

Η πολυμεταβλητή παλινδρόμηση PLS όπως εξάλλου και η μέθοδος Κυρίων Συνιστωσών (**P**rincipal **C**omponent **R**egression) είναι ιδιαιτέρως δημοφιλείς στην αντιμετώπιση προβλημάτων από τις φυσικές επιστήμες. Διότι εκεί συχνά οι επεξηγηματικές

[1] Είναι διαθέσιμο με τη φόρτωση του πακέτου pls της R (http://cran.r-project.org/package=pls)

μεταβλητές είναι πολλές και ενδεχομένως συσχετισμένες, ενώ οι παρατηρήσεις είναι αναλογικά πολύ λίγες.

Στην PLS οι ζητούμενες λανθάνουσες μεταβλητές προκύπτουν επαναληπτικά. Ξεκινώντας με την Παραγοντοποίηση Ιδιαζουσών Τιμών (Singular Value Decomposition) του πίνακα X^TY, έτσι περιλαμβάνεται η πληροφορία για τη μεταβλητότητα τόσο του X όσο και του Y αλλά και η πληροφορία για τη μεταξύ τους συσχέτιση. Τα διανύσματα w και q είναι τα βάρη των πινάκων X και Y αντίστοιχα, απ'όπου παίρνουμε τα score διανύσματα t και u,

$$t = Xw = Ew \tag{7.3}$$

$$u = Yq = Fq \tag{7.4}$$

όπου οι αρχικές τιμές των E και F είναι οι X και Y αντίστοιχα. Το score διάνυσμα t συχνά κανονικοποιείται,

$$t = t/\sqrt{t^Tt} \tag{7.5}$$

Τα score u για το Y δεν έχουν κάποια χρησιμότητα στην παλινδρόμηση αλλά συχνά αποθηκεύονται για ερμηνευτικούς λόγους. Στη συνέχεια υπολογίζονται τα φορτία των παραγόντων (loadings)

$$p = E^Tt \tag{7.6}$$

$$q = F^Tt \tag{7.7}$$

Τέλος οι πίνακες με τα δεδομένα «ξεφουσκώνουν» καθώς η πληροφορία που σχετίζεται με τη συγκεκριμένη λανθάνουσα μεταβλητή, υπό τη μορφή των εξωτερικών γινομένων tp^T και tq^T, αφαιρείται από τους τρέχοντες πίνακες δεδομένων E και F

$$E_{n+1} = E_n - tp^T \tag{7.8}$$

$$F_{n+1} = F_n - tq^t \tag{7.9}$$

Η εκτίμηση της επόμενης συνιστώσας ξεκινά από την Παραγοντοποίηση SVD του πίνακα $E_{n+1}^TF_{n+1}$. Έπειτα από κάθε επανάληψη, τα διανύσματα w, t, p και q αποθηκεύονται ως στήλες στους πίνακες W, T, P και Q αντίστοιχα. Ωστόσο οι στήλες του πίνακα W δεν είναι άμεσα συγκρίσιμες γιατί προέκυψαν από τους διαδοχικά «ξεφουσκωμένους» πίνακες E και F. Ένας εναλλακτικός τρόπος αναπαράστασης των βαρών, ούτως ώστε όλες οι στήλες να σχετίζονται με τον αρχικό πίνακα X, δίνεται από τη σχέση

$$R = W(P^TW)^{-1} \tag{7.10}$$

Στην παρούσα φάση, αντί να έχουμε την παλινδρόμηση του Y πάνω στο X, χρησιμοποιούμε τα score T για τον υπολογισμό των συντελεστών παλινδρόμησης. Κατόπιν

τους μετατρέπουμε ξανά πολλαπλασιάζοντάς τους με τον πίνακα R ούτως ώστε να γίνει η επαναφορά στις αρχικές μεταβλητές X αφού $T = XR$,

$$B = R(T^T T)^{-1} T^T Y = RQ^T$$

Μόνο οι πρώτες a συνιστώσες χρησιμοποιούνται ενώ ο καθορισμός της βέλτιστης τιμής του a δίνεται από τη διαδικασία της **cross validation**.

Ακολουθεί ο **PLS αλγόριθμος** [10].

1. Υπολογίστε το διάνυσμα των βαρών $w = \frac{X^{'} y}{y^{'} y}$, και κανονικοποιήστε το $\bar{w} = \frac{w}{\|w\|}$.
2. Υπολογίστε το score διάνυσμα $t = X\bar{w}$.
3. Υπολογίστε τα διανύσματα/φορτία των y και X ως $c = \frac{y^{'} t}{t^{'} t}$ και $p = \frac{X^{'} t}{t^{'} t}$ αντίστοιχως.
4. Υπολογίστε τα υπόλοιπα του X και y ως $E = X - tp^{'}, f = y - tc$ αντιστοίχως.
5. Θέστε $X := E$, $y := f$ και επιστρέψτε στο βήμα 1 μέχρι να υπολογιστούν όλοι οι κύριοι παράγοντες.

Ακολουθώντας τα ως άνω βήματα, προκύπτει τελικά ένα μοντέλο παλινδρόμησης.

$$y = XW(P^{'} W)^{-1} C^{'} + f = Xb + f \tag{7.11}$$

Όπου $W \in \Re^{p \times k}$ ο πίνακας των βαρών που αποτελείται από τα διανύσματα των βαρών, $P \in \Re^{p \times k}$ ο πίνακας των φορτίων του X που αποτελείται από τα διανύσματα των φορτίων του X και $C \in \Re^{1 \times k}$ ο πίνακας των φορτίων για το y που αποτελείται από τα διανύσματα των φορτίων του y που προέκυψαν από τον αλγόριθμο. Τέλος $b \in \Re^{p \times 1}$ είναι το διάνυσμα των συντελεστών του μοντέλου παλινδρόμησης.

Η PLS στην R

Το λογισμικό που χρησιμοποιείται είναι το πακέτο pls.

```
dataframe<-gasoline

attach(training)
Y<-octane
X<-NIR
detach(training)

NIRpls<- plsr(Y ~ X, ncomp = 10, data = training, validation = "LOO")
```

Προσαρμόζουμε το μοντέλο στα training δεδομένα αρχικά με 10 συνιστώσες και η μέθοδος της cross validation που επελέγη είναι η leave-one-out.

```
summary(NIRpls)

Data:   X dimension: 45 401
        Y dimension: 45 1
Fit method: kernelpls
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 45 leave-one-out segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV           1.558    1.385   0.4222   0.2817   0.2682
adjCV        1.558    1.384   0.4195   0.2817   0.2677


5 comps  6 comps  7 comps  8 comps  9 comps  10 comps
0.2839   0.2623   0.2727   0.2804   0.2703    0.2549
0.2818   0.2611   0.2711   0.2788   0.2690    0.2533


TRAINING: % variance explained
   1 comps  2 comps  3 comps  4 comps  5 comps
X    72.88    80.35    88.12    96.04    96.45
Y    28.26    94.05    97.38    97.78    98.72

 6 comps  7 comps  8 comps  9 comps  10 comps
X  97.22    97.58    98.28    98.80     98.97
Y  98.86    99.08    99.17    99.24     99.39
```

Το αποτέλεσμα της cross validation είναι η ρίζα του μέσου τετραγωνικού σφάλματος πρόβλεψης (**R**oot **M**ean **S**quared **E**rror of **P**rediction RMSEP). Σε κάθε συνιστώσα δίνονται δυο cross validation εκτιμήσεις, η συνήθης CV και η adjCV εκτίμηση η οποία είναι διορθωμένη από μεροληψία. Οι τιμές αυτές είναι το ζητούμενο RMSEP σφάλμα εάν στην ανάλυση κρατούσαμε μαζί με την «τρέχουσα» συνιστώσα και όσες προηγούνται αυτής.

```
plot(RMSEP(NIRpls), legendpos = "topright")
```

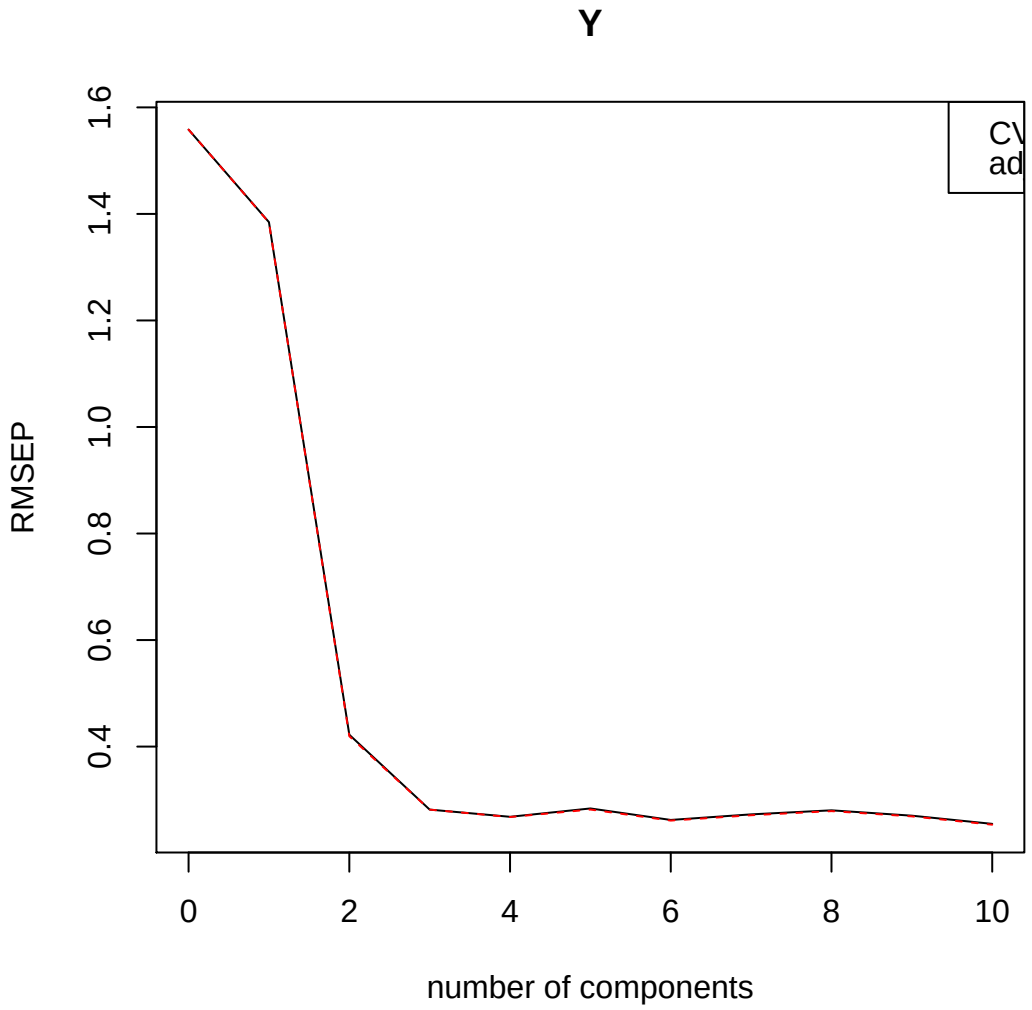
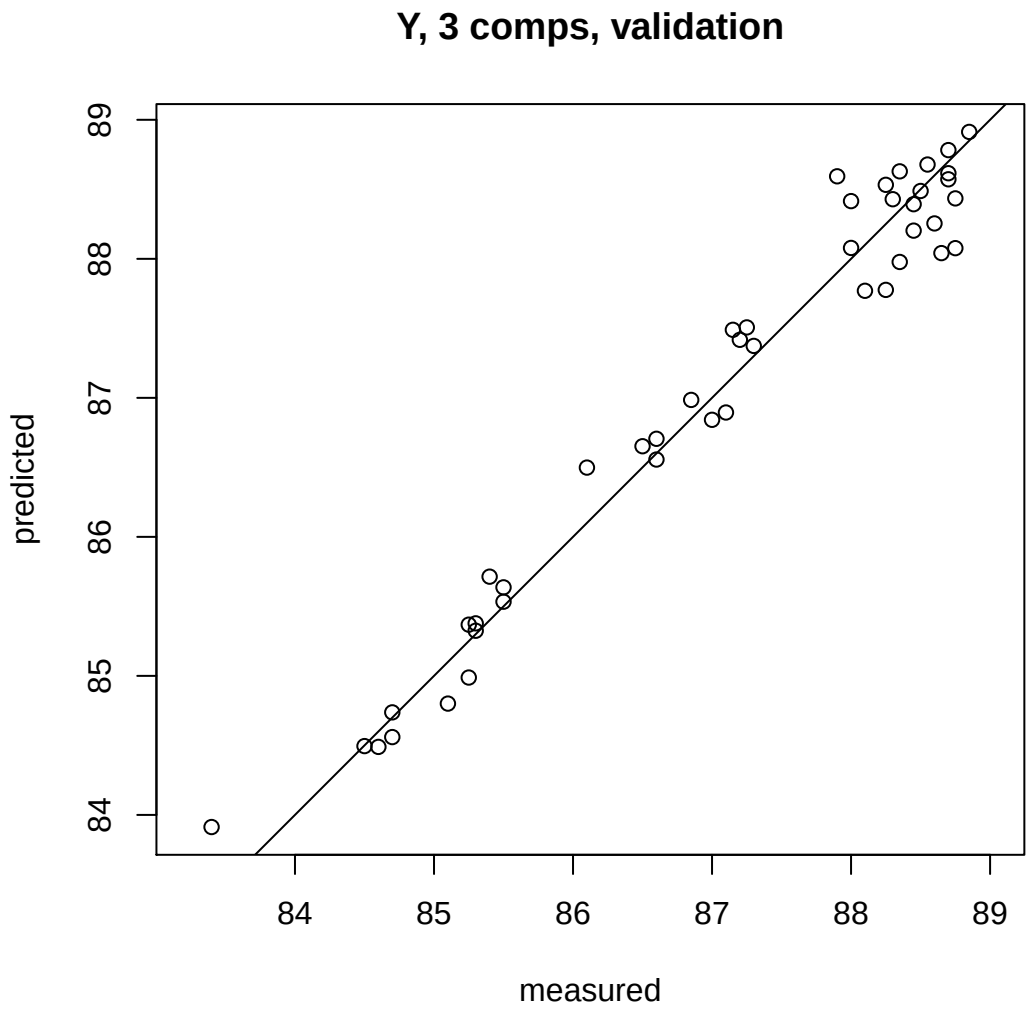


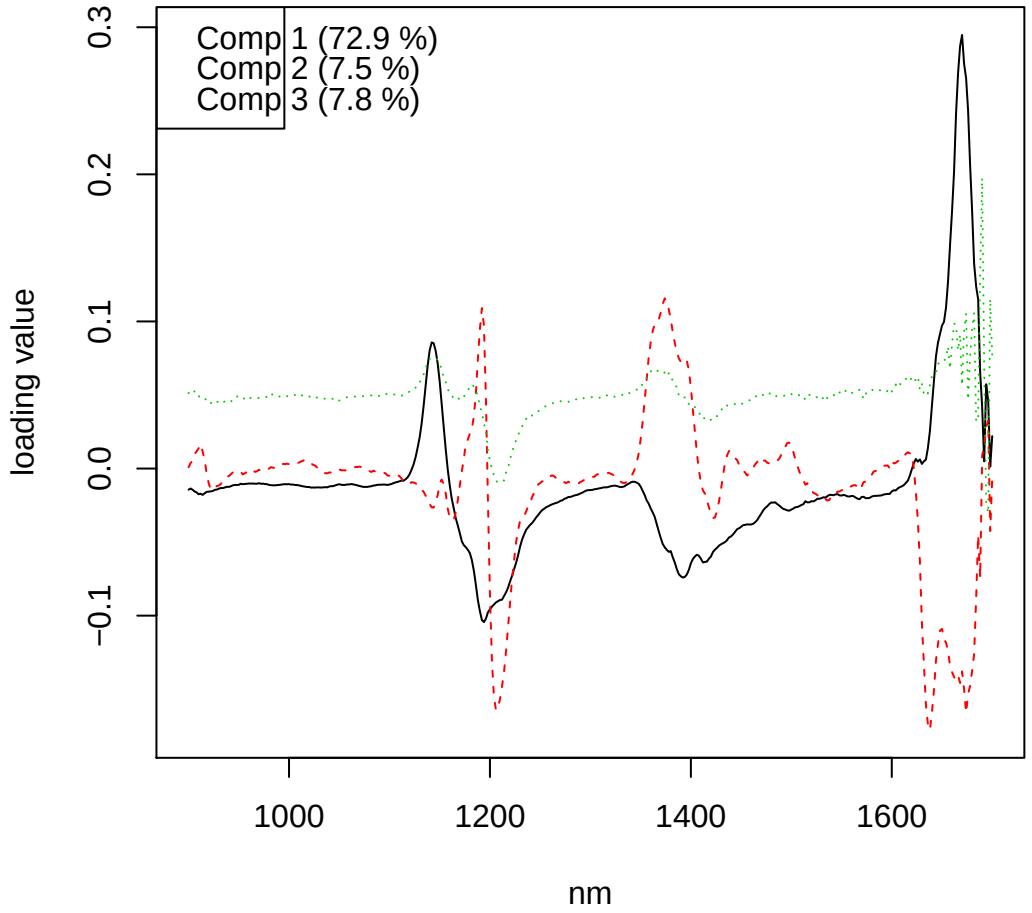
Figure 7.1: Τα εκτιμώμενα RMSEP ως συναρτήσεις του πλήθους των συνιστωσών. Τα κόκκινα ευθύγραμμα τμήματα είναι adjCV εκτιμήσεις ενώ τα μαύρα είναι οι συνήθεις CV εκτιμήσεις.

Από το σχήμα 7.1 είναι εμφανές ότι τρεις συνιστώσες είναι αρκετές διότι μέχρι τότε μειώνεται σημαντικά το σφάλμα. Άρα η τιμή του σφάλματος RMSEP ισούται με 0.2817.



Σχήμα 7.2: Cross validation προβλέψεις στο μοντέλο με τις 3 συνιστώσες vs οι πραγματικές μετρήσεις. Τα σημεία δεν εμφανίζουν κάποια καμπύλη και φαίνεται να είναι «κοντά» στην ευθεία.

```
plot(NIRpls, "loadings", comps = 1:3, legendpos = "topleft",
labels = "numbers", xlab = "nm")
```



Σχήμα 7.3: Το σχήμα με τα φορτία χρησιμοποιείται για ερμηνευτικούς λόγους, όπως για παράδειγμα η αναζήτηση γνωστών φασματικών προφίλ. Με μαύρο χρώμα απεικονίζεται η πρώτη συνιστώσα, με κόκκινο η δεύτερη και με πράσινο η τρίτη.

7.2 Stochastic Gradient Boosting

Όπως είδαμε στην ενότητα 6.2 η μέθοδος Stochastic Gradient Boosting, από εδώ και στο εξής SGB, είναι η βελτιωμένη εκδοχή της μεθόδου Gradient Boosting, GB. Αμφότερες αναπτύχθηκαν από τον Jerome H. Friedman το 2001 και το 2002 και αποτελούν την επέκταση της δουλειάς των Friedman, Hastie, και Tibshirani το 2000.

Η μέθοδος **GB** ή διαφορετικά η μέθοδος **Gradient Boosting Machine**, είναι ουσιαστικά η σύνδεση μεταξύ boosting και βελτιστοποίησης. Καταρχάς με δεδομένες τις τιμές του training δείγματος x_i, y_i για $i = 1, \ldots, N$, επιθυμούμε να βρούμε μια συνάρτηση παλινδρόμησης $\widehat{f}(x)$, η οποία ελαχιστοποιεί τη μέση τιμή κάποιας συνάρτησης απώλειας $\Psi(y, f)$, δηλαδή

$$\widehat{f}(x) = \arg\min_{f(x)} E_{y,x}\Psi(y,f(x)) \tag{7.12}$$

Μέσω του boosting, η $\widehat{f}(x)$ προσεγγίζεται από μια αθροιστική επέκταση της μορφής

$$f(x) = \sum_{m=0}^{M} \beta_m h(x;a_m) \tag{7.13}$$

όπου οι συναρτήσεις $h(x;a)$ οι οποίες είναι οι base learners συνήθως επιλέγονται να είναι απλές συναρτήσεις του x με παραμέτρους $a = (a_0, a_1, a_2, \ldots, a_M)$. Οι συντελεστές επέκτασης $\beta_0, \beta_1, \ldots, \beta_M$ και οι παράμετροι $a_0, a_1, \ldots, a_M$ προσαρμόζονται από κοινού στα training δεδομένα με τον προς τα εμπρός και κατά βήματα τρόπο. Η αρχή γίνεται με την ανάθεση μιας τυχαίας τιμής $f_0(x)$ και στη συνέχεια για $m = 1, 2, \ldots, M$ έχουμε

$$(\beta_m, a_m) = \arg\min_{\beta,a} \sum_{i=1}^{N} \Psi\left(y_i, f_{m-1}(x_i) + \beta h(x_i;a)\right) \tag{7.14}$$

και

$$f_m(x) = f_{m-1}(x) + \beta_m h(x;a_m) \tag{7.15}$$

Ο GB επιλύει προσεγγιστικά την 1.14 για οποιαδήποτε διαφορίσιμη συνάρτηση απώλειας $\Psi(y,f(x))$ σε δυο βήματα. Πρώτον, εφαρμόζεται η συνάρτηση $h(x;a)$ με τη μέθοδο των ελαχίστων τετραγώνων

$$a_m = \arg\min_{a,\rho} \sum_{i=1}^{N} \left[\widetilde{y}_{im} - \rho h(x_i;a)\right]^2 \tag{7.16}$$

στα τρέχοντα ψευδο-υπόλοιπα

$$\widetilde{y}_{im} = -\Big[\frac{\partial\Psi(y_i,f(x_i))}{\partial f(x_i)}\Big]_{f(x)=f_{m-1}(x)} \tag{7.17}$$

Τότε με δεδομένη την $h(x;a_m)$, η βέλτιστη τιμή του συντελεστή β_m δίνεται από τη σχέση

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} \Psi\left(y_i, f_{m-1}(x_i) + \beta h(x_i;a_m)\right) \tag{7.18}$$

Αυτή η στρατηγική υποκαθιστά ένα δύσκολο ενδεχομένως πρόβλημα βελτιστοποίησης συνάρτησης (σχέση 7.14) με κάποιο άλλο που βασίζεται στα ελάχιστα τετράγωνα (σχέση 7.16), ακολουθούμενο από τη βελτιστοποίηση μιας παραμέτρου (σχέση 7.18) βασισμένη στο γενικό κριτήριο απώλειας Ψ.

Ο Tree GB εξειδικεύει αυτήν την προσέγγιση στην περίπτωση που ο base learner $h(x;a)$ είναι ένα δέντρο παλινδρόμησης με L τελικούς κόμβους. Σε κάθε επανάληψη m, ένα δέντρο παλινδρόμησης διαμερίζει το χώρο του x σε L ξένα σύνολα R_{lm} με $l = 1,\ldots,L$ και δίνει την πρόβλεψη μιας σταθεράς ξεχωριστά για καθένα από αυτά.

$$h(x;\{R_{lm}\}_1^L) = \sum_{l=1}^{L} \bar{y}_{lm} I(x \in R_{lm}) \tag{7.19}$$

Όπου $\bar{y}_{lm} = E_{x_i \in R_{lm}}(\tilde{y}_{im})$ είναι η μέση τιμή της σχέσης 7.17 σε κάθε περιοχή R_{lm}. Οι παράμετροι αυτού του βασικού μηχανισμού εκμάθησης (base learner) είναι οι μεταβλητές διαχωρισμού και τα αντίστοιχα σημεία διαχωρισμού (splits) που ορίζουν το δέντρο, τα οποία με τη σειρά τους καθορίζουν τις αντίστοιχες περιοχές $\{R_{lm}\}_1^L$ της διαμέρισης κατά την m-επανάληψη. Το κριτήριο διαχωρισμού που χρησιμοποιείται είναι εκείνο των ελαχίστων τετραγώνων. Με τα δέντρα παλινδρόμησης η σχέση 7.18 μπορεί να επιλυθεί ξεχωριστά σε κάθε περιοχή R_{lm} που ορίζεται από τον αντίστοιχο τελικό κόμβο l του m-οστού δέντρου. Επειδή το δέντρο στη σχέση 7.19 δίνει την πρόβλεψη μιας σταθερής τιμής $\bar{y}_{lm}$ σε κάθε περιοχή R_{lm}, η λύση της 7.18 περιορίζεται σε μια απλή εκτίμηση με βάση το κριτήριο Ψ

$$\gamma_{lm} = \arg\min_{\gamma} \sum_{x_i \in R_{lm}} \Psi(y_i, f_{m-1}(x_i) + \gamma)$$

Η τρέχουσα προσέγγιση $f_{m-1}(x)$ αναβαθμίζεται ξεχωριστά σε κάθε περιοχή

$$f_m(x) = f_{m-1}(x) + \nu \cdot \gamma_{lm} I(x \in R_{lm})$$

Η παράμετρος σμίκρυνσης ν ($0 < \nu \leq 1$) ελέγχει το ρυθμό εκμάθησης της διαδικασίας. Εμπειρικά έχει βρεθεί ότι μικρές τιμές της παραμέτρου σμίκρυνσης (≤ 0.1) οδηγούν σε καλύτερο σφάλμα γενίκευσης.
Ακολουθεί ο αλγόριθμος GB [9],

Αρχικά θέτουμε η $\widehat{f}(x)$ να είναι σταθερά, $\widehat{f}(x) = \arg\min_{\gamma} \sum_{i=1}^{N} \Psi(y_i, \gamma)$.

Για $m = 1,\ldots,M$ εκτελέστε τα ακόλουθα βήματα:

1. Υπολογίστε για $i = 1, N$ το αρνητικό ανάδελτα $\tilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
2. $\{R_{lm}\}_1^L$ = δέντρο με L τελικούς κόμβους $(\{\tilde{y}_{im}, x_i\}_1^N)$
3. $\gamma_{lm} = \arg\min_{\gamma} \sum_{x_i \in R_{lm}} \Psi(y_i, f_{m-1}(x_i) + \gamma)$
4. $f_m(x) = f_{m-1}(x) + \nu \cdot \gamma_{lm} I(x \in R_{lm})$

Εκτός από το κριτήριο των ελαχίστων τετραγώνων, $\Psi(y,f) = (y-f)^2$, άλλες συναρτήσεις απώλειας που χρησιμοποιούνται είναι οι εξής

- ελάχιστη απόλυτη απόκλιση $\Psi(y,f) = |y-f|$
- Huber $\Psi(y,f) = (y-f)^2 I(|y-f| \leq \delta) + 2\delta(|y-f| - \delta/2) I(|y-f| > \delta)$

Stochastic Gradient Boosting

Η μέθοδος SGB δημιουργήθηκε όταν στον GB αλγόριθμο ενσωματώθηκε η τυχαιότητα. Συγκεκριμένα σε κάθε επανάληψη λαμβάνεται τυχαία και χωρίς επανάθεση ένα υποδείγμα των training δεδομένων. Κατόπιν χρησιμοποιείται το εν λόγω υποδείγμα αντί για το πλήρες training set, προκειμένου να γίνει η προσαρμογή του βασικού μηχανισμού εκμάθησης (βήμα 2) και να υπολογιστεί η αναβάθμιση του μοντέλου κατά την τρέχουσα επανάληψη (βήμα 3) [9].

Έστω $\{y_i, x_i\}_1^N$ ολόκληρο το training δείγμα και $\{\pi(i)\}_1^N$ μια τυχαία μετάθεση των ακεραίων $\{1, \ldots, N\}$. Τότε ένα τυχαία υποδείγμα μεγέθους $\tilde{N} < N$ συμβολίζεται με $\left\{y_{\pi(i)}, x_{\pi(i)}\right\}_1^{\tilde{N}}$. Ακολουθεί ο Stochastic Gradient Boosting αλγόριθμος [9],

Αρχικά θέτουμε η $\widehat{f}(x)$ να είναι σταθερά, $\widehat{f}(x) = \arg\min_\gamma \sum_{i=1}^N \Psi(y_i, \gamma)$.

Για $m = 1, \ldots, M$ εκτελέστε τα ακόλουθα βήματα:

1. $\{\pi(i)\}_1^N$ η τυχαία μετάθεση των ακεραίων $1, \ldots, N$
2. Υπολογίστε για $i = 1, \tilde{N}$ το αρνητικό ανάδελτα $\tilde{y}_{\pi(i)m} = -\left[\frac{\partial \Psi(y_{\pi(i)}, f(x_{\pi(i)}))}{\partial f(x_{\pi(i)})}\right]_{f(x)=f_{m-1}(x)}$
3. $\{R_{lm}\}_1^L$ = δέντρο με L τελικούς κόμβους $(\left\{\tilde{y}_{\pi(i)m}, x_{\pi(i)}\right\}_1^{\tilde{N}})$
4. $\gamma_{lm} = \arg\min_\gamma \sum_{x_{\pi(i)} \in R_{lm}} \Psi(y_{\pi(i)}, f_{m-1}(x_{\pi(i)}) + \gamma)$
5. $f_m(x) = f_{m-1}(x) + \upsilon \cdot \gamma_{lm} I(x \in R_{lm})$

Εάν θέσουμε $\tilde{N} = N$ τότε δεν έχουμε την τυχαιότητα και επανερχόμαστε στον GB αλγόριθμο. Όσο μικρότερο είναι το κλάσμα $k = \frac{\tilde{N}}{N}$, τόσο περισσότερο θα διαφέρουν τα τυχαία δείγματα που χρησιμοποιούνται στις διαδοχικές επαναλήψεις, επομένως τότε έχουμε μεγαλύτερη τυχαιότητα στη διαδικασία συνολικά. Για πολύ μικρές τιμές του k λιγοστεύουν τα διαθέσιμα δεδομένα για την εκπαίδευση του μηχανισμού εκμάθησης σε κάθε επανάληψη. Κάτι τέτοιο θα οδηγούσε σε αύξηση τη διακύμανσης που σχετίζεται με τους εκτιμητές των base learners ξεχωριστά.

Stochastic Gradient Boosting στην R

Το λογισμικό που χρησιμοποιείται είναι το πακέτο gbm. Καταρχάς λόγω του μεγάλου πλήθους των επεξηγηματικών μεταβλητών - το NIR φάσμα αποτελείται από 401 μεταβλητές - προτιμάται η συνάρτηση gbm.fit αντί της gbm. Στο σημείο αυτό πρέπει να αποφασιστούν οι τιμές που θα δοθούν στα ορίσματα της συνάρτησης.

Για τη **συνάρτηση απώλειας** σε προβλήματα παλινδρόμησης Ψ (όρισμα `distribution`) υπάρχει η επιλογή της Κανονικής κατανομής για την ελαχιστοποίηση του τετραγωνικού σφάλματος, και η επιλογή της κατανομής Laplace για την ελαχιστοποίηση του απόλυτου σφάλματος.

Το όρισμα `interaction.depth` αναφέρεται στο μέγιστο βάθος των αλληλεπιδράσεων των μεταβλητών. Η τιμή 1 υποδηλώνει ένα αθροιστικό μοντέλο, η τιμή 2 ένα μοντέλο με έως και 2 way αλληλεπιδράσεις κ.ο.κ. Επιπλέον, καθώς το πακέτο χτίζει ένα boosting σύνολο δέντρων με `interaction.depth` στο πλήθος επίπεδα, σε

κάθε δέντρο υπάρχουν $2^{interaction.depth}$ τελικοί κόμβοι. Παρόλο που το boosting μπορεί ν'αναπαραστήσει μόνο αθροιστικές αλληλεπιδράσεις, θεωρητικά η μόνη ελπίδα για την πλήρη « αποκρυπτογράφηση » της μη γραμμικής λειτουργίας N χαρακτηριστικών/μεταβλητών, είναι η κωδικοποίησή της ως δέντρα με N επίπεδα - κάτι που δικαιολογεί και την ονομασία του εν λόγω ορίσματος.

Το όρισμα `n.trees` δηλαδή το σύνολο των δέντρων που προσαρμόζονται, ισοδυναμεί με το **πλήθος των επαναλήψεων** και με το πλήθος των βασικών συναρτήσεων στην αθροιστική επέκταση. Η ανάθεση μιας τιμής σχετίζεται και με την επιλογή της **παραμέτρου σμίκρυνσης** (όρισμα `shrinkage`) η οποία είναι επίσης γνωστή και ως ρυθμός εκμάθησης ή step-size reduction. Γενικά μικρότερες τιμές της παραμέτρου σμίκρυνσης σχεδόν πάντα έχουν βελτιωμένη προβλεπτική ικανότητα. Δηλαδή θέτωντας `shrinkage=0.001` είναι σχεδόν βέβαιο ότι θα έχουμε ένα μοντέλο με καλύτερη προβλεπτική ικανότητα σε επόμενα δείγματα απ'ότι αν θέταμε `shrinkage=0.01`. Όμως μια μικρή τιμή για την παράμετρο σμίκρυνσης συνεπάγεται υψηλό υπολογιστικό κόστος, τόσο σε χρόνο όσο και σε μνήμη. Ένα μοντέλο με σμίκρυνση ίση με 0.001 απαιτεί 10 φορές περισσότερες επαναλήψεις συγκρινόμενο με το μοντέλο με παράμετρο σμίκρυνσης ίση με 0.01, αυξάνοντας έτσι το κόστος αποθήκευσης και τον υπολογιστικό χρόνο κατά έναν παράγοντα του 10. Εμπειρικά μπορεί να ειπωθεί ότι η σμίκρυνση πρέπει να κυμαίνεται από 0.001 εώς 0.01 για 3000 έως 10000 επαναλήψεις.

Ο ακέραιος αριθμός `nTrain` δεν είναι άλλος από το πλήθος των δεδομένων εκπαίδευσης. Οι παράμετροι `nTrain` και `train.fraction` είναι αμοιβαία αποκλειόμενες ενώ αν δε δοθεί καμμία τιμή και στις δύο τότε όλα τα δεδομένα λαμβάνονται υπόψιν για τη διαδικασία της εκπαίδευσης. Εάν στο όρισμα `train.fraction` ανατεθεί η τιμή μικρότερη του 1 τότε μόνο τα πρώτα `train.fraction`$\times$`nrow(gasoline)` δεδομένα χρησιμοποιούνται στην προσαρμογή του μοντέλου. Σημειώνεται ότι επειδή τα εν λόγω δεδομένα δεν εμφανίζουν καμία διάταξη, δε χρειάζεται ν'αναμιχθούν προηγουμένως. Εκείνα που έμειναν εκτός, μπορούν να χρησιμοποιηθούν στον υπολογισμό ενός αμερόληπτου εκτιμητή του **βέλτιστου αριθμού επαναλήψεων**. Το μειονέκτημα αυτής της προσέγγισης είναι ότι ένας σημαντικός αριθμός παρατηρήσεων προορίζεται για την εκτίμηση μιας μόνο παραμέτρου, μένοντας μ'ένα περιορισμένο σύνολο δεδομένων για την εκτίμηση του μοντέλου. Μετά την προσαρμογή του gbm μοντέλου, η συνάρτηση **gbm.perf** υπολογίζει την εκτίμηση για τον αριθμό των επαναλήψεων προσφέροντας τρεις επιλογές σχετικά με την ακολουθούμενη μέθοδο.

Πρώτον, την επιλογή ενός ανεξάρτητου συνόλου test δεδομένων, η οποία προφανώς προϋποθέτει `train.fraction` < 1. Δεύτερον, την επιλογή του **out-of-bag** εκτιμητή της βελτίωσης στην προβλεπτική ικανότητα του μοντέλου, θέτοντας `bag.fraction` να είναι μεγαλύτερο του μηδενός με προτεινόμενη τιμή το 0.5. Εκτιμά τη μείωση στην απόκλιση σ'εκείνες τις παρατηρήσεις που δεν λήφθηκαν υπόψιν κατά την επιλογή του επόμενου δέντρου παλινδρόμησης. Ο εκτιμητής out-of-bag υποεκτιμά τη μείωση στην απόκλιση, κατά συνέπεια δίνει σχεδόν πάντα μια πολύ συντηρητική εκτίμηση του βέλτιστου αριθμού επαναλήψεων. Η ιδέα γι'αυτή τη μέθοδο ήταν ν'αποφευθεί η μη αξιοποίηση ενός μεγάλου ανεξάρτητου συνόλου δεδομένων, που εκ των πραγμάτων περιορίζει τις διαθέσιμες πληροφορίες για την εκμάθηση του μοντέλου. Η τρίτη μέθοδος που προσφέρεται είναι η cross validation. Θέτωντας `cv.folds=5` εφαρμόζεται η 5απλή cross validation, δηλαδή το πρόγραμμα προσαρμόζει πέντε gbm.fit μοντέλα προκειμένου να υπολογιστεί η εκτίμηση του cross validation σφάλματος και στη συνέχεια προσαρμόζει ένα έκτο και τελικό gbm.fit μοντέλο με `n.trees` επαναλήψεις

χρησιμοποιώντας όλα τα δεδομένα.

Σχετικά με τα υπόλοιπα ορίσματα της συνάρτησης **gbm.fit** έχουμε το `n.minobsinnode` που είναι ο ελάχιστος αριθμός παρατηρήσεων στους τελικούς κόμβους των δέντρων.

Εφαρμογή

Στην εφαρμογή με τα NIR δεδομένα προσαρμόζουμε αρχικά ένα μοντέλο με τα ακόλουθα χαρακτηριστικά. Για την ελαχιστοποίηση του τετραγωνικού σφάλματος επιλέγεται η Κανονική κατανομή ως συνάρτηση απώλειας, η σμίκρυνση επιλέγεται αρχικά να είναι ίση με 0.05 αφού συνήθως οι τιμές από 0.001 έως το 0.1 είναι αποτελεσματικές, ενώ η αρχική τιμή που ανατίθεται στις επαναλήψεις είναι 1000. Μεριμνούμε και για τη δημιουργία ενός συνόλου δεδομένων εκπαίδευσης ώστε να είναι εφικτή αργότερα η εκτίμηση του βέλτιστου αριθμού επαναλήψεων.

```
model1<-gbm.fit(x,y,
offset = NULL,
misc = NULL,
distribution = "gaussian",
w = NULL,
var.monotone = NULL,
n.trees = 1000,
interaction.depth = 1,
n.minobsinnode = 3,
shrinkage = 0.05,
bag.fraction = 0.5,
nTrain = 45,
train.fraction = NULL,
keep.data = TRUE,
verbose = TRUE,
var.names = NULL,
response.name = NULL,
group = NULL)
```

```
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        2.1919          2.0262     0.0500    0.1209
     2        2.0527          1.8963     0.0500    0.1049
     3        1.9537          1.7701     0.0500    0.0855
     4        1.8156          1.6496     0.0500    0.1417
     5        1.7053          1.5405     0.0500    0.0817
     6        1.6612          1.6114     0.0500    0.0148
     7        1.5520          1.5091     0.0500    0.0954
     8        1.4703          1.4289     0.0500    0.0782
     9        1.4450          1.4912     0.0500   -0.0080
    10        1.4048          1.5134     0.0500    0.0225
    20        0.8412          1.1774     0.0500   -0.0019
    40        0.4125          0.7682     0.0500    0.0069
    60        0.2578          0.6707     0.0500    0.0082
    80        0.1811          0.6349     0.0500    0.0007
   100        0.1321          0.5958     0.0500   -0.0023
   120        0.1043          0.5786     0.0500   -0.0023
   140        0.0881          0.5847     0.0500    0.0000
   160        0.0733          0.5655     0.0500   -0.0019
   180        0.0635          0.5850     0.0500   -0.0017
   200        0.0537          0.5812     0.0500   -0.0002
```

```
  220        0.0459          0.5721     0.0500    0.0007
  240        0.0413          0.5660     0.0500   -0.0002
  260        0.0367          0.5641     0.0500   -0.0005
  280        0.0336          0.5727     0.0500   -0.0006
  300        0.0310          0.5699     0.0500   -0.0002
  320        0.0283          0.5615     0.0500   -0.0006
  340        0.0263          0.5537     0.0500   -0.0002
  360        0.0255          0.5620     0.0500   -0.0003
  380        0.0238          0.5559     0.0500   -0.0002
  400        0.0226          0.5563     0.0500   -0.0001
  420        0.0218          0.5599     0.0500    0.0002
  440        0.0213          0.5666     0.0500   -0.0001
  460        0.0204          0.5609     0.0500   -0.0001
  480        0.0197          0.5603     0.0500    0.0000
  500        0.0192          0.5608     0.0500   -0.0001
  520        0.0187          0.5611     0.0500   -0.0001
  540        0.0186          0.5616     0.0500   -0.0002
  560        0.0182          0.5646     0.0500   -0.0001
  580        0.0178          0.5634     0.0500   -0.0001
  600        0.0177          0.5658     0.0500   -0.0001
  620        0.0176          0.5650     0.0500   -0.0001
  640        0.0174          0.5637     0.0500   -0.0000
  660        0.0171          0.5636     0.0500    0.0001
  680        0.0170          0.5657     0.0500    0.0001
  700        0.0167          0.5640     0.0500   -0.0001
  720        0.0166          0.5643     0.0500    0.0000
  740        0.0166          0.5653     0.0500   -0.0000
  760        0.0166          0.5646     0.0500    0.0000
  780        0.0167          0.5648     0.0500   -0.0000
  800        0.0166          0.5644     0.0500    0.0000
  820        0.0166          0.5648     0.0500   -0.0000
  840        0.0165          0.5634     0.0500   -0.0000
  860        0.0165          0.5617     0.0500   -0.0000
  880        0.0165          0.5602     0.0500   -0.0000
  900        0.0165          0.5609     0.0500   -0.0000
  920        0.0164          0.5607     0.0500    0.0000
  940        0.0165          0.5611     0.0500   -0.0000
  960        0.0165          0.5608     0.0500    0.0000
  980        0.0164          0.5600     0.0500   -0.0000
 1000        0.0164          0.5603     0.0500   -0.0000
```

Στην πρώτη στήλη δίνονται οι επαναλήψεις. Για κάθε επανάληψη δίνεται στη στήλη TrainDeviance η τιμή της συνάρτησης απώλειας πάνω στα δεδομένα εκπαίδευσης. Αντίστοιχα στη στήλη ValidDeviance εμφανίζεται η τιμή της συνάρτησης απώλειας πάνω στα test δεδομένα.

Η στήλη Improve δίνει τη μείωση της συνάρτησης απώλειας από επανάληψη σε επανάληψη και υπολογίζεται χρησιμοποιώντας τις λεγόμενες out-of-bag παρατηρήσεις, ενώ η στήλη StepSize είναι η παράμετρος σμίκρυνσης/ρυθμός εκμάθησης.

Έχοντας επιλέξει τη συνάρτηση απώλειας, το πλήθος των επαναλήψεων T, το βάθος κάθε δέντρου K ή διαφορετικά το μέγιστο βαθμό αλληλεπιδράσεων μεταξύ των μεταβλητών του μοντέλου, την παράμετρο σμίκρυνσης λ ή διαφορετικά το ρυθμό εκμάθησης και τέλος το ρυθμό υποδειγματοληψίας (όρισμα `bag.fraction`) p, ο αλγόριθμος εκτελείται από το πακέτο gbm ως εξής,

Αρχικά θέτουμε η $\widehat{f}(x)$ να είναι σταθερά, $\widehat{f}(x) = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, \rho)$.

Για $t = 1, \ldots, T$ εκτελέστε τα ακόλουθα βήματα,

1. Υπολογίστε το αρνητικό ανάδελτα

$$z_i = -\Big[\frac{\partial \Psi(y_i, f(x_i))}{\partial f(x_i)}\Big]_{f(x_i)=\widehat{f}(x_i)}$$

2. Επιλέξτε τυχαία $p * N$ παρατηρήσεις απ'το σύνολο των δεδομένων

3. Προσαρμόστε ένα δέντρο παλινδρόμησης με K τελικούς κόμβους, $g(x) = E(z|x)$. Το συγκεκριμένο δέντρο προσαρμόζεται χρησιμοποιώντας μόνο τις τυχαία επιλεγμένες παρατηρήσεις.

4. Υπολογίστε τις βέλτιστες προβλέψεις των τελικών κόμβων $\rho_1, \ldots, \rho_K$, ως

$$\rho_k = \arg\min_\rho \sum_{x_i \in S_k} \Psi(y_i, \widehat{f}(x_i) + \rho)$$

όπου S_k είναι το σύνολο των μεταβλητών που ορίζουν τον τελικό κόμβο k.

5. Αναβαθμίστε την $\widehat{f}(x)$ ως

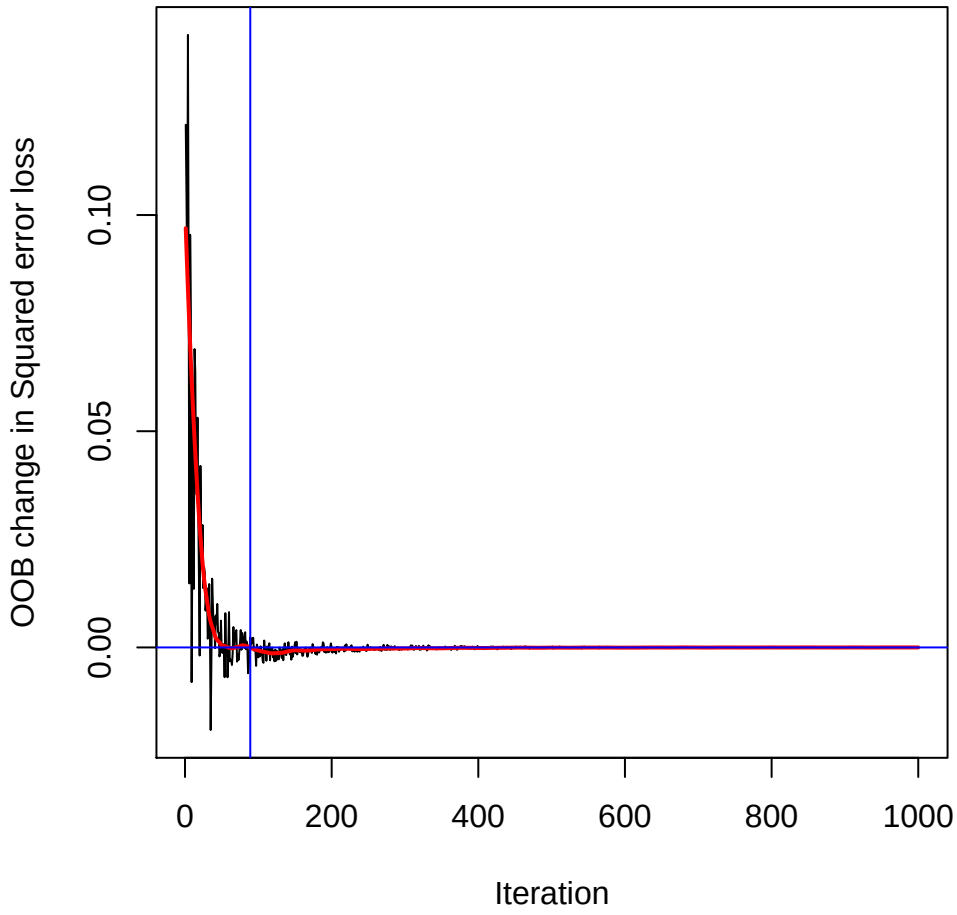
$$\widehat{f}(x) \leftarrow \widehat{f}(x) + \lambda \rho_{k(x)}$$

όπου το $k(x)$ είναι ο δείκτης του τελικού κόμβου στον οποίο θα περιλαμβάνονταν μια παρατήρηση με τις τιμές x. Οι παρατηρήσεις που χρησιμοποιούνται και σ'αυτό το βήμα είναι μόνο εκείνες που επιλέχθηκαν τυχαία.

Για την εκτίμηση του βέλτιστου αριθμού των boosting επαναλήψεων του αλγορίθμου πραγματοποιούνται οι ακόλουθοι έλεγχοι,

```
best.iter1<-gbm.perf(model1,
   plot.it = TRUE,
           oobag.curve = TRUE,
           overlay = TRUE,
           method="OOB")
print(best.iter1)
[1] 89
```

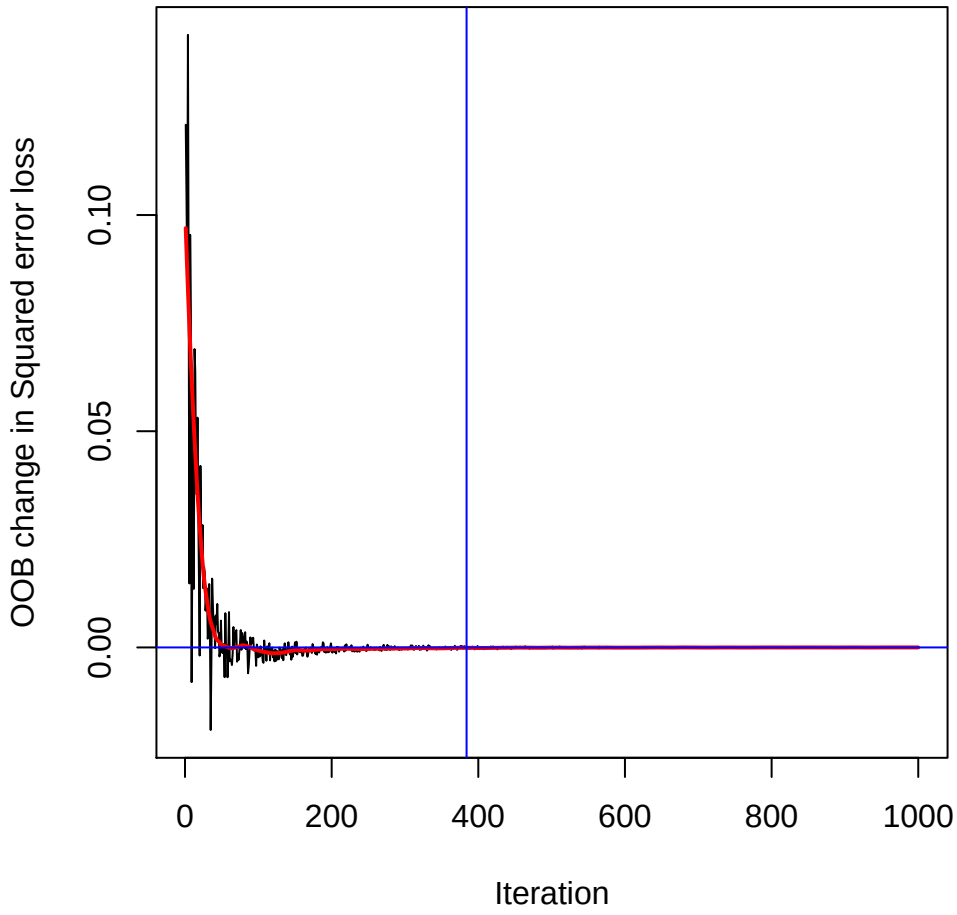
Άρα βάση της μεθόδου **Out-of-bag** το 89 είναι ο εκτιμώμενος βέλτιστος αριθμός επαναλήψεων, αν και πρέπει να σημειωθεί ότι ο συγκεκριμένος εκτιμητής τον υποεκτιμά. Ο OOB θεωρείται πως είναι ένας αμερόληπτος εκτιμητής του πραγματικού σφάλματος του μοντέλου. Η λογική του σχετίζεται με το ότι στη δημιουργία οποιουδήποτε δέντρου εκ του συνόλου (ensemble) χρησιμοποιούμε ένα δείγμα των δεδομένων εκπαίδευσης που καλείται bootstrap δείγμα [6]. Οι παρατηρήσεις που δεν περιέχονται στο συγκεκριμένο δείγμα και επομένως δε χρησιμεύουν στην κατασκευή του μοντέλου καλούνται out-of-bag. Οι εν λόγω παρατηρήσεις χρησιμεύουν αυτή τη φορά ως test δεδομένα. Κάποιες παρατηρήσεις θα είναι out-of-bag για ένα εύλογο αριθμό φορών, επομένως υπολογίζοντας την αναλογία των φορών σωστής ταξινόμησης μιας τέτοιας παρατήρησης, έχουμε και τις εκτιμήσεις της απόδοσης ολόκληρου του μοντέλου.



Σχήμα 7.4: Είναι προφανές ότι μέχρι και την εκατοστή επανάληψη η μείωση του τετραγωνικού σφάλματος είναι σημαντική.

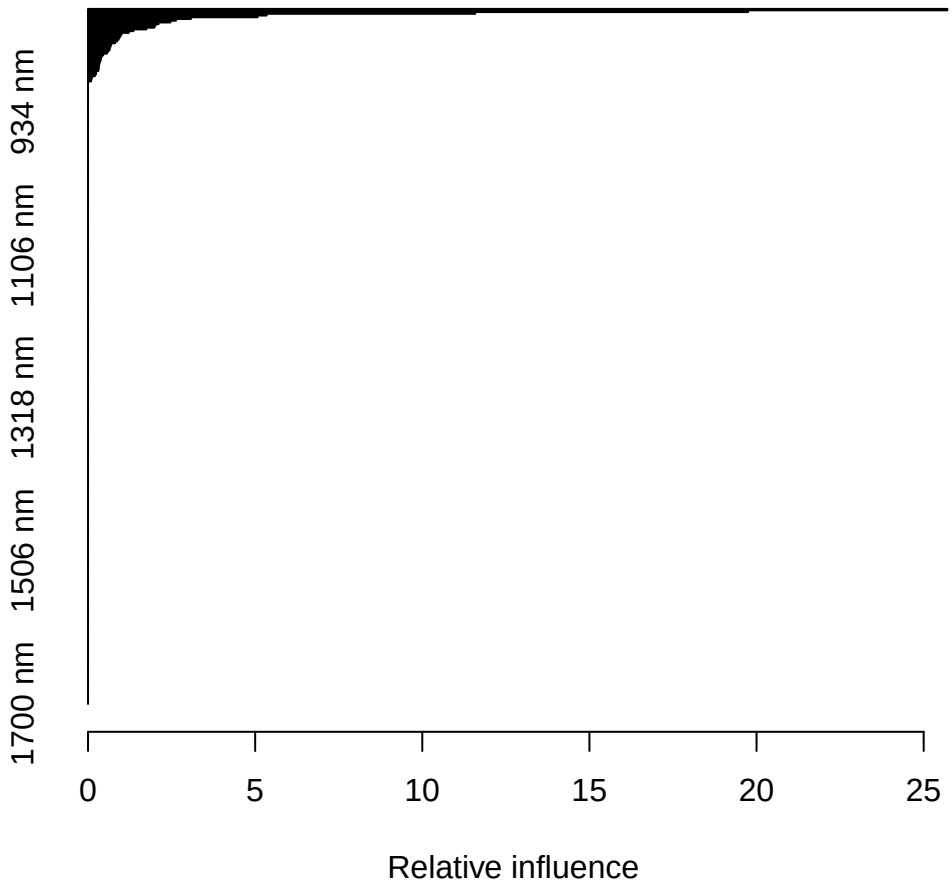
Στον έλεγχο της αποδοτικότητας με την **test** μέθοδο, γίνεται η χρήση των δεδομένων επικύρωσης προκειμένου να υπολογιστεί ένας εκτιμητής εκτός δείγματος (out-of-sample)

```
best.iter2<-gbm.perf(model1,
   plot.it = TRUE,
           oobag.curve = TRUE,
           overlay = TRUE,
           method="test")
print(best.iter2)
[1] 384
```



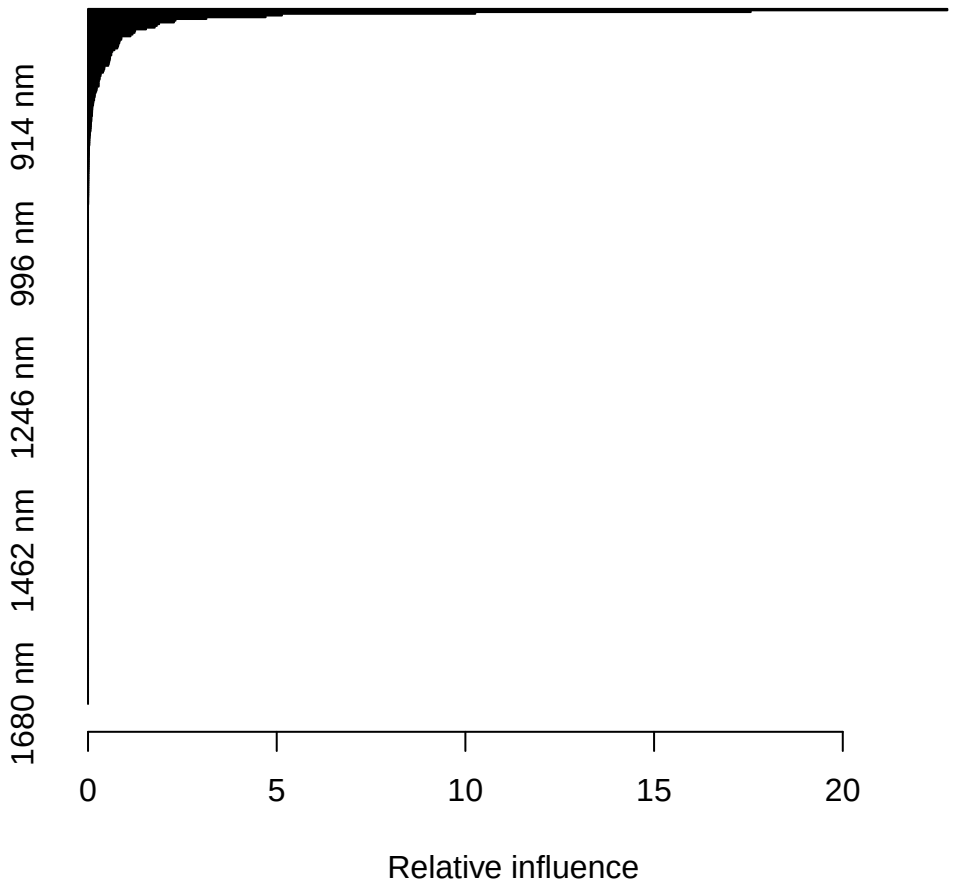
Σχήμα 7.5: Σύμφωνα με την test μέθοδο ο βέλτιστος αριθμός επαναλήψεων είναι το 384.

```
summary(model1,n.trees=best.iter1)
summary(model1,n.trees=best.iter2)
```



Σχήμα 7.6: Η απεικόνιση των μεταβλητών με τη μεγαλύτερη σχετική επιρροή στο μοντέλο, βασισμένη στη βέλτιστη Out-of-Bag εκτίμηση για τον αριθμό των δέντρων/επαναλήψεων.

Το πακέτο εκτιμά την σχετική επιρροή των μεταβλητών με τη βοήθεια της συνάρτησης `relative.influence` η οποία αναπτύχθηκε από τον Friedman. Τα μέτρα βασίζονται στις φορές που μια μεταβλητή επιλέγεται για διαχωρισμούς, με τα βάρη τους να εξαρτώνται από το τετράγωνο της βελτίωσης που επέρχεται στο μοντέλο εξαιτίας κάθε διαχωρισμού, και για τα οποία λαμβάνεται η μέση τιμή απ'όλα τα δέντρα που προκύπτουν.



Σχήμα 7.7: Η απεικόνιση των μεταβλητών με τη μεγαλύτερη σχετική επιρροή στο μοντέλο, βασισμένη στη βέλτιστη test εκτίμηση για τον αριθμό των δέντρων/επαναλήψεων. Από το σύνολο των 401 μεταβλητών μόνο οι 113 είχαν μη μηδενική επιρροή.

Τρέχοντας το μοντέλο ακόμα δύο φορές, με τον αριθμό των επαναλήψεων να ισούται αρχικά με **89** (model2) και κατόπιν με **384** (model3), λαμβάνουμε τα εξής αποτελέσματα,

```
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        2.1894          2.0606     0.0500    0.1523
     2        2.0191          1.9007     0.0500    0.1574
     3        1.8838          1.7564     0.0500    0.1005
     4        1.7856          1.6479     0.0500    0.0728
     5        1.7017          1.5531     0.0500    0.0817
     6        1.5906          1.4460     0.0500    0.0679
     7        1.5276          1.4850     0.0500    0.0372
     8        1.4361          1.4184     0.0500    0.0673
     9        1.3513          1.3284     0.0500    0.0792
    10        1.2732          1.2396     0.0500    0.0784
```

```
    20        0.7542          0.9723     0.0500    0.0289
    40        0.3841          0.6929     0.0500    0.0018
    60        0.2477          0.6200     0.0500   -0.0060
    80        0.1866          0.5892     0.0500   -0.0013
    89        0.1707          0.5730     0.0500   -0.0056

print(pretty.gbm.tree(model2,1))
SplitVar SplitCodePred LeftNode RightNode MissingNode
0      151    0.35839150        1         2       3
1       -1    0.04428758       -1        -1      -1
2       -1   -0.12251389       -1        -1      -1
3       -1    0.01251587       -1        -1      -1



ErrorReduction Weight  Prediction
  36.03706     21     0.01251587
   0.00000     17     0.04428758
   0.00000      4    -0.12251389
   0.00000     21     0.01251587
```

Ο πίνακας περιέχει όλη την πληροφορία που έχει εξαχθεί από το πρώτο δέντρο ή διαφορετικά την πρώτη επανάληψη για το μοντέλο με τις 89 επαναλήψεις. Η στήλη SplitVar δείχνει τη μεταβλητή που χρησιμοποιήθηκε για το διαχωρισμό ενώ το -1 υποδηλώνει τελικό κόμβο. Η μεταβλητή X_{151} αντιστοιχεί στα 1200 νανόμετρα. Στη στήλη SplitCodePred δίνεται η τιμή του σημείου διαχωρισμού και στην περίπτωση τελικού κόμβου δίνεται η πρόβλεψη. Με LeftNode και RightNode εμφανίζεται ο δείκτης της γραμμής που αντιστοιχεί στον αριστερό και δεξιό κόμβο. Στο ErrorReduction δίνεται η μείωση στη συνάρτηση απώλειας που επέρχεται λόγω του διαχωρισμού του κόμβου αναφοράς. Τέλος δίνονται τα βάρη, δηλαδή το πλήθος των παρατηρήσεων ανά κόμβο, αφού στο μοντέλο όλες οι μεταβλητές είχαν βάρος ίσο με τη μονάδα.

```
print(pretty.gbm.tree(model2,model2$n.trees))
SplitVar SplitCodePred LeftNode RightNode MissingNode
0       12  -0.044878500        1         2       3
1       -1  -0.003136992       -1        -1      -1
2       -1  -0.043729449       -1        -1      -1
3       -1  -0.007002941       -1        -1      -1


ErrorReduction Weight   Prediction
     1.192655     21   -0.007002941
     0.000000     19   -0.003136992
     0.000000      2   -0.043729449
     0.000000     21   -0.007002941
```

Αντίθετα για το τελευταίο δέντρο του μοντέλου έχουμε ότι η μεταβλητή που χρησιμοποιήθηκε για το διαχωρισμό είναι η X_{12} δηλαδή τα 922 νανόμετρα, με τους τελικούς κόμβους να είναι και εδώ τρεις. Απ'τις 401 μεταβλητές μόνο οι 39 είχαν μη μηδενική επιρροή.

```
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        2.2710          2.1074     0.0500    0.0170
     2        2.1117          1.9629     0.0500    0.1407
```

```
     3        1.9756          1.8379     0.0500    0.1133
     4        1.8582          1.7258     0.0500    0.1063
     5        1.7352          1.6024     0.0500    0.0978
     6        1.6477          1.5045     0.0500    0.0768
     7        1.5296          1.3988     0.0500    0.1018
     8        1.4621          1.3271     0.0500    0.0130
     9        1.4028          1.2649     0.0500   -0.0001
    10        1.3380          1.2968     0.0500    0.0269
    20        0.8038          0.9475     0.0500    0.0376
    40        0.4176          0.7103     0.0500    0.0070
    60        0.2630          0.6511     0.0500    0.0025
    80        0.1870          0.6429     0.0500    0.0039
   100        0.1453          0.6049     0.0500   -0.0004
   120        0.1206          0.5780     0.0500    0.0004
   140        0.0947          0.5631     0.0500   -0.0006
   160        0.0810          0.5470     0.0500    0.0010
   180        0.0643          0.5256     0.0500   -0.0003
   200        0.0561          0.5187     0.0500    0.0007
   220        0.0469          0.5097     0.0500   -0.0005
   240        0.0429          0.5009     0.0500   -0.0005
   260        0.0358          0.4949     0.0500   -0.0004
   280        0.0317          0.4928     0.0500    0.0003
   300        0.0269          0.4839     0.0500   -0.0003
   320        0.0243          0.4865     0.0500    0.0003
   340        0.0216          0.4869     0.0500   -0.0002
   360        0.0196          0.4830     0.0500   -0.0002
   380        0.0181          0.4810     0.0500    0.0002
   384        0.0178          0.4748     0.0500   -0.0002
```

```
print(pretty.gbm.tree(model3,1))
  SplitVar SplitCodePred LeftNode RightNode MissingNode
0      131    0.14247050        1         2           3
1       -1    0.04443464       -1        -1          -1
2       -1   -0.11063889       -1        -1          -1
3       -1    0.01489683       -1        -1          -1

  ErrorReduction Weight  Prediction
       31.14763     21  0.01489683
        0.00000     17  0.04443464
        0.00000      4 -0.11063889
        0.00000     21  0.01489683
```

Το πρώτο δέντρο του μοντέλου των 384 επαναλήψεων έχει ως μεταβλητή διαχωρισμού την X_{131} δηλαδή τα 1160 νανόμετρα.

```
print(pretty.gbm.tree(model3,model3$n.trees))
  SplitVar SplitCodePred LeftNode RightNode MissingNode
0      398  1.2150455000        1         2           3
1       -1 -0.0084141434       -1        -1          -1
2       -1  0.0005936403       -1        -1          -1
3       -1 -0.0006931860       -1        -1          -1


 ErrorReduction Weight    Prediction
     0.08345846     21 -0.0006931860
     0.00000000      3 -0.0084141434
```

```
0.00000000     18  0.0005936403
0.00000000     21 -0.0006931860
```

Ενώ το τελευταίο δέντρο έχει ως μεταβλητή διαχωρισμού την X_{398} δηλαδή τα 1694 νανόμετρα. Απ'τις 401 μεταβλητές μόνο οι 111 είχαν μη μηδενική επιρροή.

Απομένει να υπολογιστεί η ποσότητα **R**oot **M**ean **S**quared **E**rror of **P**rediction, δηλαδή η ρίζα του μέσου τετραγωνικού σφάλματος πρόβλεψης, ούτως ώστε να γίνει εφικτή η σύγκριση της μεθόδου SGB με τις μεθόδους PLS και Bagging.

Για τον υπολογισμό του RMSEP προσαρμόζεται το gbm μοντέλο με το πλήθος των επαναλήψεων να είναι ίσο με 384 στα test δεδομένα. Εφόσον τα training δεδομένα ήταν οι πρώτες 45 παρατηρήσεις του αρχικού πλαισίου δεδομένων x και του αρχικού διανύσματος y, αρκεί να δημιουργηθεί ένα νέο πλαίσιο δεδομένων xtest και ένα νέο διάνυσμα ytest εκ των αρχικών x και y που θα περιέχουν τις υπόλοιπες, τελευταίες 15 παρατηρήσεις.

```
f.predict <- predict(model3,xtest,384)
f.predict
[1] 87.74983 87.71901 87.87658 87.77716 87.87690 87.95459
    88.12052 88.05549 84.91562 85.37994 84.72385 87.89393
    87.72652 88.13443 87.66963
```

Η ποσότητα f.predict είναι το διάνυσμα των προβλέψεων για τα test δεδομένα σύμφωνα με το SGB μοντέλο των 384 επαναλήψεων και ακολουθεί ο υπολογισμός του σφάλματος με τη μέθοδο των ελαχίστων τετραγώνων.

```
squared_error <- print(sum((ytest-f.predict)^2))
squared_error
[1] 7.12161
```

Επομένως έχουμε ότι

$$RMSEP = \sqrt{\frac{SquaredError}{15}} = \sqrt{\frac{7.12161}{15}} = 0.689$$

7.3 Bagging στην παλινδρόμηση

Όπως έχει ήδη αναφερθεί στην ενότητα 6.3.1 του παρόντος το bagging, δηλαδή το **B**ootstrap **Agg**regat**ing** είναι μια μέθοδος η οποία συνδυάζει πολλαπλούς μηχανισμούς πρόβλεψης. Έχοντας στη διάθεσή μας ένα σύνολο δεδομένων εκπαίδευσης $D = \{(y_i, x_i) : i = 1, \ldots, n\}$, θέλουμε να προβλέψουμε την απόκριση μιας καινούργιας παρατήρησης. Λαμβάνουμε με δειγματοληψία με επανάθεση B τυχαία δείγματα των n παρατηρήσεων από το αρχικό σύνολο D. Τα δείγματα $\{D_1, D_2, \ldots, D_B\}$ είναι τα B training σύνολα δεδομένων που έχουν αναπαραχθεί.

Κατόπιν εκπαιδεύουμε ένα μοντέλο σε κάθε σύνολο D_b, $b = 1, \ldots, B$ και προκύπτει μια ακολουθία B προβλέψεων. Ο τελικός αθροιστικός ταξινομητής λαμβάνεται ως μέσος όρος στην παλινδρόμηση, και ως ψήφος πλειοψηφίας στην ταξινόμηση.

Αναλυτικά ο αλγόριθμος έχει την ακόλουθη μορφή,

1. Αναπαράγετε εκ του συνόλου D, B bootstrap δείγματα $D_1, \ldots, D_B$
2. Εφαρμόστε τον αλγόριθμο εκμάθησης σε κάθε σύνολο $D_b, b = 1, \ldots, B$ για να ληφθεί η υπόθεση ή πρόβλεψη h_b
3. Έστω $T_b = \frac{D}{D_b}$ οι τιμές των δεδομένων που δεν εμφανίζονται στο δείγμα D_b (σημεία out-of-bag)
4. Υπολογίστε την πρόβλεψη $h_b(x)$ για κάθε $x \in T_b$
5. Για κάθε τιμή x είναι πλεον διαθέσιμες η αντίστοιχη παρατηρημένη τιμή y και οι προβλέψεις $\hat{y_1}(x), \hat{y_2}(x), \ldots, \hat{y_K}(x)$. Υπολογίστε τη μέση πρόβλεψη $\hat{h}(x)$.
6. Εκτιμήστε τη μεροληψία
$$y - \hat{h}(x)$$
και τη διασπορά
$$\sum_{k=1}^{K} \left\{ \hat{y_k}(x) - \hat{h}(x) \right\}^2$$

Εαν οι ταξινομητές είναι ασταθείς, δηλαδή παρουσιάζουν υψηλή διασπορά, ο αθροιστικός ταξινομητής έχει μικρότερη διασπορά συγκρινόμενος με τον καθένα ξεχωριστά. Ο αθροιστικός ταξινομητής μπορεί να θεωρηθεί ως μια προσέγγιση του πραγματικού μέσου f που προκύπτει αντικαθιστώντας την κατανομή πιθανότητας p με την bootstrap προσέγγιση της, συγκεντρώνοντας μάζα $1/n$ σε κάθε σημείο (x_i, y_i).

Η μέθοδος του bagging δίνει καλά αποτελέσματα όταν οι αλγόριθμοι εκμάθησης είναι ιδιαιτέρως ασταθείς, όταν δηλαδή για μικρές μεταβολές στο σύνολο εκπαίδευσης δίνουν μεγάλες μεταβολές στις προβλέψεις. Παραδείγματα ασταθών αλγορίθμων είναι τα Νευρωνικά δίκτυα, τα δέντρα απόφασης και παλινδρόμησης καθώς και η επιλογή υποσυνόλων στη λογιστική παλινδρόμηση.

Ωστόσο το bagging μπορεί να υποβαθμίσει ελαφρά την αποδοτικότητα ευσταθών αλγορίθμων εκμάθησης όπως είναι η μέθοδος των K πλησιέστερων γειτόνων. Σε γενικές γραμμές μπορούμε να πούμε ότι είναι αποτελεσματικό όταν ο αλγόριθμος είναι αποδοτικός κατά μέσο όρο αλλά ασταθής σε σχέση με τα δεδομένα εκπαίδευσης. Η εφαρμογή του bagging σε ευσταθή αλγόριθμο χειροτερεύει τα πράγματα.

Το γενικό πλαίσιο της μεθοδολογίας του bagging σε προβλήματα ταξινόμησης και παλινδρόμησης, είναι το πακέτο ipred (Improved Predictors).

```
model1<- ipredbagg(y, X=x, nbagg=25, control=rpart.control(xval=0),
comb=NULL, coob=TRUE, ns=length(y), keepX = TRUE)
model1

Bagging regression trees with 25 bootstrap replications
Out-of-bag estimate of root mean squared error:  0.7842


model1<- ipredbagg(y, X=x, nbagg=50, control=rpart.control(xval=0),
comb=NULL, coob=TRUE, ns=length(y), keepX = TRUE)
model1
```

```
Bagging regression trees with 50 bootstrap replications
Out-of-bag estimate of root mean squared error:  0.7455
```

Επιλέγοντας ως αριθμό bootstrap δειγμάτων το 25 και το 50 βλέπουμε ότι οι ποσότητες RMSEP είναι ίσες με 0.7842 και 0.7455 αντίστοιχα (όρισμα `nbagg`). Στο όρισμα `ns` δίνεται το πλήθος των παρατηρήσεων σε κάθε δείγμα που αναπαράγεται και στην εφαρμογή μας ισούται με 60. Για καθένα από τα `nbagg` bootstrap δείγματα που αναπαράγονται δημιουργείται ένα δέντρο παλινδρόμησης με τη βοήθεια της γνωστής συνάρτησης rpart.

Συμπεράσματα

Από την εφαρμογή με τα δεδομένα NIR έχουμε για την ποσότητα **RMSEP**, δηλαδή για τη ρίζα του μέσου τετραγωνικού σφάλματος πρόβλεψης, τις εξής τιμές: 0.2817 για την παλινδρόμηση PLS, 0.689 για την Stochastic Gradient Boosting και 0.7455 για το Bagging. Άρα στη συγκεκριμένη περίπτωση η PLS φαίνεται να είναι η καλύτερη επιλογή με δεύτερο καλύτερο το SGB. Η PLS γενικά είναι μια πολύ καλή προσέγγιση όταν σε κάποιο πρόβλημα, ο αριθμός των επεξηγηματικών μεταβλητών είναι πολύ μεγαλύτερος από τον αριθμό των παρατηρήσεων και υπάρχει επιπλέον πολυσυγγραμικότητα στις τιμές του X.

Αξίζει να σημειωθεί ότι στην περίπτωση που τα δεδομένα παρουσιάζουν θόρυβο, είναι πιθανό το boosting να μην έχει καλή απόδοση αφού φαίνεται να είναι ιδιαιτέρως επιρρεπές σ'αυτόν. Ωστόσο σε άλλες περιπτώσεις το boosting μπορεί να βελτιώσει την ακρίβεια οποιουδήποτε αλγορίθμου και να βελτιώσει την απόδοση ενός αδύναμου μηχανισμού εκμάθησης. Ακόμη ένα βασικό πλεονέκτημα του boosting είναι ότι μια μη γραμμική σχέση, τόσο στην ταξινόμηση όσο και στην παλινδρόμηση, μπορεί να μοντελοποιηθεί ως ένας τρόπος προσαρμογής μιας αθροιστικής επέκτασης σε μια συλλογή βασικών μηχανισμών εκμάθησης.

Ιστότοποι

1. Andrea Peters, Torsten Hothorn. Improved Predictors. R Package Version 0.8-13, 2012. http://cran.r-project.org/web/packages/ipred/index.html.

2. Bjorn-Helge Mevik, Ron Wehrens, Kristian Hovde Liland. Partial Least Squares and Principal Component regression. R package version 2.3-0, 2011. http://cran.rproject.org/web/packages/pls/index.html.

3. Gaston Sanchez. Tools of the Trade for Discriminant Analysis. R package version 0.1-22, 2012.http://cran.r-project.org/web/packages/DiscriMiner/index.html.

4. Greg Ridgeway. Generalized Boosted Regression Models. R package version 1.6-3.2, 2012. http://cran.r-project.org/web/packages/gbm/index.html.

5. Mark Culp, Kjell Johnson, George Michailidis.ada: an R package for Stochastic Boosting. R package version 2.0-3, 2007. http://cran.rproject. org/web/packages/ada/index.html.

6. Brian Ripley. tree: Classification and regression trees. R package version 1.0-28,2010. http://CRAN.R-project.org/package=tree.

7. Terry M. Therneau, Beth Atkinson. R port by Brian Ripley. rpart: Recursive Partitioning.R package version 3.1-46, 2010. http://CRAN.R-project.org/package=rpart.

Bibliography

[1] Shalizi C. Lecture notes: Classification and Regression Trees. 1999.

[2] Chung D. and Keles S. Sparse Partial Least Squares Classification for High Dimensional Data.

[3] Mease D., Wyner A.J., and Buja A. Boosted Classification Trees and Class Probability/Quantile Estimation. *Journal of Machine Learning Research*, 8(2007):409–439, 2007.

[4] Hand D.J. Data Mining: Statistics and More? *The American Statistician*, 52(2):112–118, May 1998.

[5] Cao D.S., Xu Q.S., Liang Y.Z., Zhang L.X., and Li H.D. The boosting: A new idea of building models. *Chemometrics and Intelligent Laboratory Systems Elsevier*, 100(2010):1–11, 2009.

[6] Williams G. *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*. Use R! Springer, 1st edition, 2011.

[7] Witten I. and Frank E. *Data Mining, Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier, 2nd edition, 2005.

[8] SPSS Inc. *SPSS Clementine 12.0 Algorithms Guide*, 2007.

[9] Friedman J.H. Stochastic Gradient Boosting. 1999.

[10] Yu L. and Wu T. Boosting the partial least square algorithm for regression modelling. *Journal of Control Theory and Applications*, 3:257–260, 2006.

[11] Culp M., Johnson K., and Michailidis G. ada: an R Package for Stochastic Boosting. *Journal of Statistical Software*.

[12] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

[13] Tan S.M., Luo R.M., Zhou Y.P., Xu H., Song D.D., Ze T., Yang T.M., and Nie Y. Boosting partial least-squares discriminant analysis with application to near infrared spectroscopic tea variety discrimination. *Journal of Chemometrics John Wiley and Sons*, 26:34–39, 2012.

[14] Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning*. Data Mining, Inference and Prediction. Springer, 2nd edition, 2008.

[15] Therneau T.M. and Atkinson E.J. *An Introduction to Recursive Partitioning / Using the RPART Routines*, 2011.

[16] Venables W.N. and Ripley B.D. *Modern Applied Statistics with S*. Statistics and Computing. Springer, 4th edition, 2002.