ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΕΣ ΣΠΟΥΔΕΣ ΣΤΑ ΜΑΘΗΜΑΤΙΚΑ

Διπλωματική Εργασία

Αλγόριθμοι Μηχανικής Μάθησης για την εξόρυξη δεδομένων

Σταύρος Οικονόμου

Επιβλέπων καθηγητής: Δημήτριος Σωτηρόπουλος

Τρίκαλα, Ιούλιος 2020

Αλγόριθμοι μηχανικής μάθησης για την εξόρυξη δεδομένων

Σταύρος Οικονόμου

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:
Δημήτριος Σωτηρόπουλος
Επίκουρος Καθηγητής
Πανεπιστήμιο Πελοποννήσου

Συν-Επιβλέπων Καθηγητής:
Νικόλαος Τσίτσας
Αναπληρωτής Καθηγητής
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τρίκαλα, Ιούλιος 2020

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Δ. Σωτηρόπουλο για τη συνεργασία μας, την εμπιστοσύνη του στο πρόσωπό μου, και τις παρατηρήσεις του κατά τη διάρκεια εκπόνησης της παρούσας εργασίας. Επίσης, θα ήθελα να ευχαριστήσω και όλους τους υπόλοιπους διδάσκοντες καθηγητές που είχα τη χαρά να γνωρίσω και να συνεργαστώ κατά τη διάρκεια της φοίτησης μου στο Ε.Α.Π.

Πρωτίστως όμως, θα ήθελα να ευχαριστήσω την οικογένειά μου και ιδιαίτερα τους γονείς μου Βασίλη και Κατερίνα όπως και την αδερφή μου Λορέτα για την στήριξη και την δύναμη που μου παρείχαν καθ'όλη την διάρκεια της φοίτησης μου.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως θέμα τους αλγόριθμους μηχανικής μάθησης. Ο κύριος άξονάς της είναι μια συνολική παρουσίαση των δημοφιλέστερων αλγορίθμων επιβλεπόμενης και μη επιβλεπόμενης μάθησης. Αρχικά έχουμε την μαθηματική παρουσίαση των αλγορίθμων, ύστερα την εφαρμογή τους σε πραγματικά δεδομένα με χρήση της γλώσσας προγραμματισμού R και τέλος τον σχολιασμό και την σύγκριση των αλγορίθμων και της απόδοσής τους. Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στην μηχανική μάθηση και στην εξόρυξη των δεδομένων, το δεύτερο κεφάλαιο αναφέρεται στους βασικούς αλγορίθμους για προβλήματα παλινδρόμησης, στα επόμενα τρία κεφάλαια (3,4,5) γίνεται αναφορά στους κυριότερους αλγόριθμους ταξινόμησης ενώ στο κεφάλαιο έξι παρουσιάζουμε τους αλγόριθμους μη επιβλεπόμενης μάθησης. Τέλος, στο κεφάλαιο εφτά εφαρμόζουμε και συγκρίνουμε τους αλγόριθμους ταξινόμησης σε βάση δεδομένων με μετρήσεις σε όγκους για την διάγνωση του καρκίνου του μαστού. Οι βάσεις δεδομένων της εργασίας καθώς και ο κώδικας στην R υπάρχει ανεβασμένος και στο GitHub μου, στο https://github.com/stavoikono/Thesis .

Λέξεις-Κλειδιά

Μηχανική μάθηση, εξόρυξη δεδομένων, αλγόριθμοι ταξινόμησης, αλγόριθμοι παλινδρόμησης, μη επιβλεπόμενη μάθηση

Abstract

This dissertation investigates Machine Learning algorithms. The study focuses on an overall examination of the most frequently used supervised and unsupervised machine learning algorithms. Initially, we have the mathematical presentation of the algorithms, then their application to real data using the programming language R and, finally, the commentary and comparison of the algorithms and their performance. The first chapter is an introduction to machine learning and data mining, the second chapter refers to the basic algorithms for regression problems, in the next three chapters (3,4,5) reference is made to the main classification algorithms while in chapter six we present the unsupervised learning algorithms. Finally, in chapter seven we apply and compare the classification algorithms in a database with tumor measurements for breast cancer diagnosis. All the datasets used in this thesis as well as the R code are uploaded to my GitHub at https://github.com/stavoikono/Thesis .

Keywords

Machine learning, data mining, classification algorithms, regression algorithms, unsupervised learning

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

1 Εισαγωγή

1.1 Γενικά για την Μηχανική Μάθηση

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

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

κές φορές συγχέεται με την εξόρυξη δεδομένων, όπου η τελευταία επικεντρώνεται περισσότερο στην εξερευνητική ανάλυση των δεδομένων, γνωστή και ως μη επιτηρούμενη μάθηση.

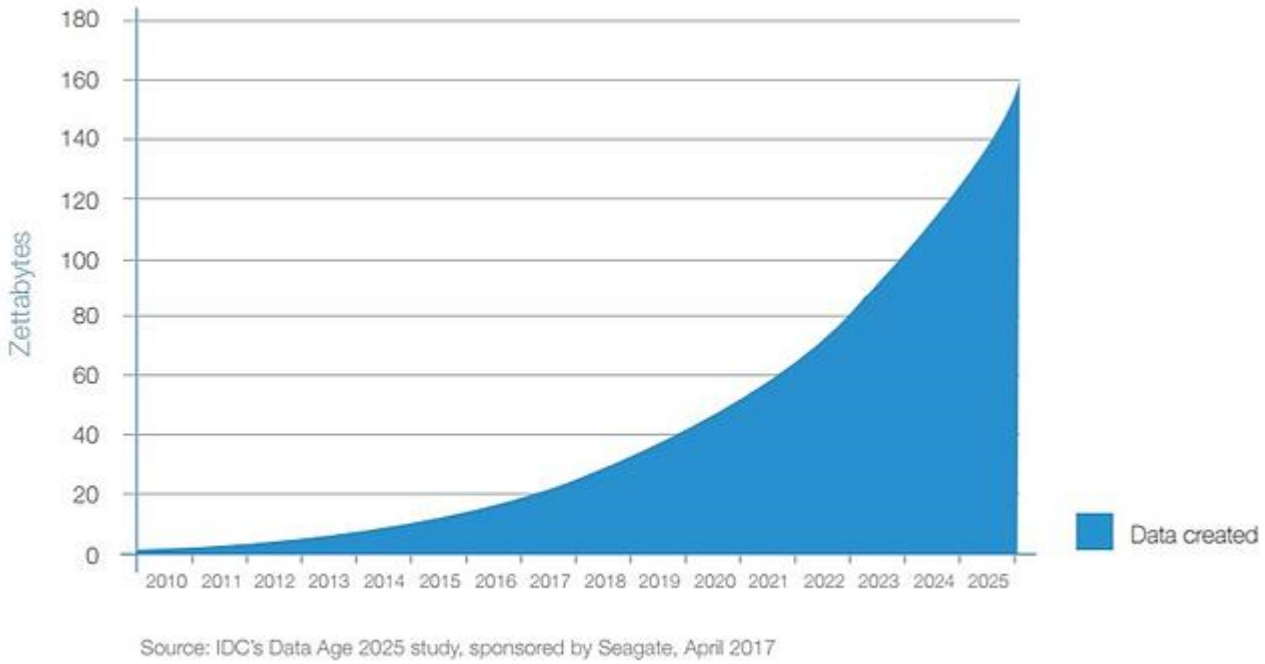
Στο πεδίο της ανάλυσης δεδομένων, η μηχανική μάθηση είναι μια μέθοδος που χρησιμοποιείται για την επινόηση πολύπλοκων μοντέλων και αλγορίθμων που οδηγούν στην πρόβλεψη. Τα αναλυτικά μοντέλα επιτρέπουν στους ερευνητές, τους επιστήμονες δεδομένων, τους μηχανικούς και τους αναλυτές να παράγουν αξιόπιστες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

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

1.2 Εφαρμογές Μηχανικής Μάθησης στην εξόρυξη δεδομένων

Εξόρυξη δεδομένων είναι η εξερεύνηση και ανάλυση μεγάλων δεδομένων για την ανεύρεση σημαντικών προτύπων και κανόνων.

Ζούμε στην εποχή των Μεγάλων Δεδομένων (Big Data). Για παράδειγμα από την απαρχή του χρόνου μέχρι το 2005 οι άνθρωποι δημιούργησαν δεδομένα 130 EB(Exabytes=10^{18} bytes), το 2010 θα γίνουν 1.200 EB, μέχρι το 2015 θα είναι 7.900 EB ενώ υπολογίζεται ότι μέχρι το 2025 ο αριθμός θα έχει φτάσει στα 160.000 EB(160 Zettabytes).

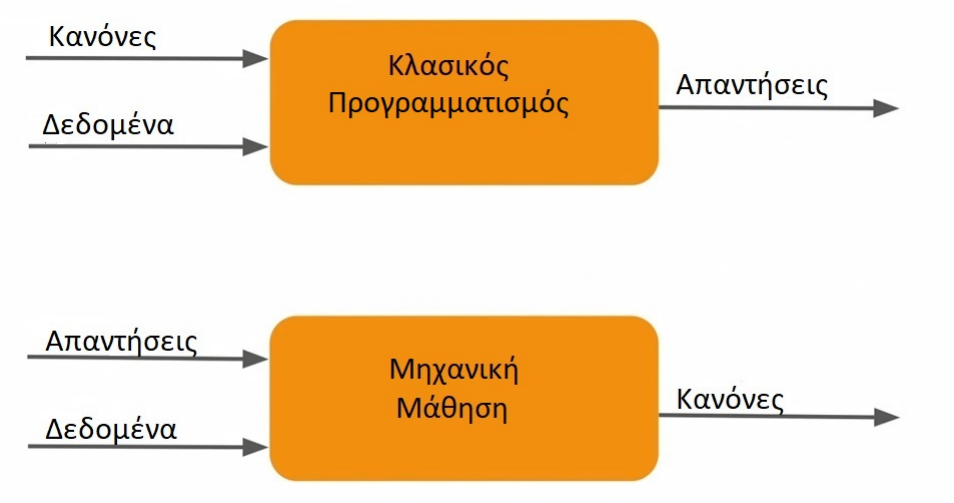


Σχήμα 1.1: Παραγώγη δεδομένων 2010-2025, Zettabytes =10^{21}bytes. [14]

Με την παραγωγή τέτοιας ποσότητας δεδομένων η διαδικασία της ανάλυσης και της εξαγωγής μοτίβων χειροκίνητα θα ήταν μια επίπονη και χρονο-

βόρα διαδικασία. Με την εξέλιξη της τεχνολογίας και της επιστήμης υπολογιστών η χειρωνακτική ανάλυση των δεδομένων έχει αντικατασταθεί από την αυτόματη επεξεργασία δεδομένων. Εκεί είναι που μπαίνει στην διαδικασία της εξόρυξης δεδομένων οι αλγόριθμοι μηχανικής μάθησης.

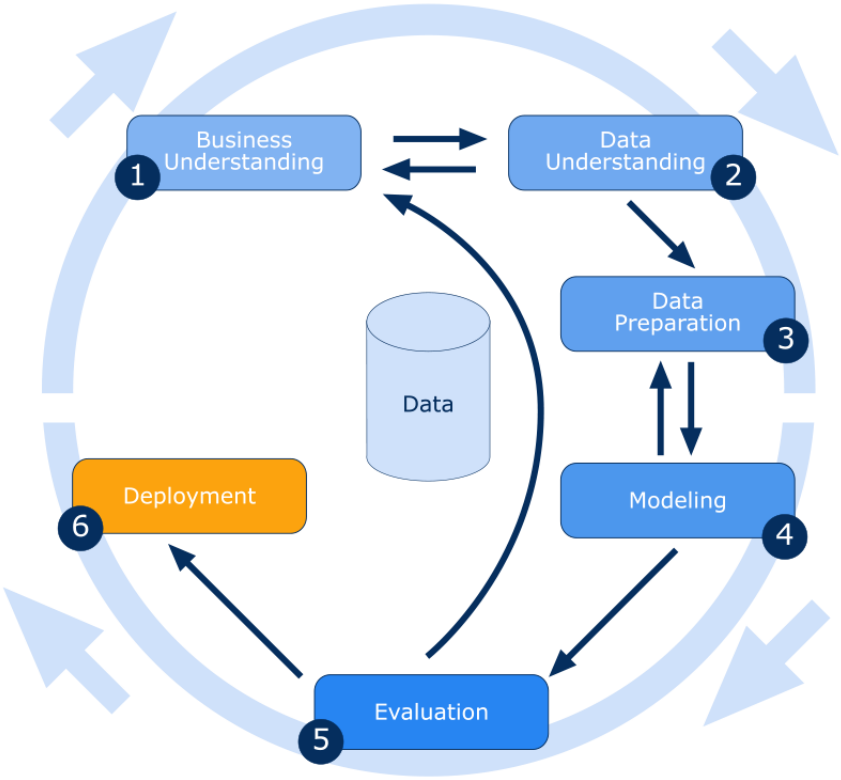
Τι είναι όμως αυτό που κάνει την μηχανική μάθηση τόσο ξεχωριστή σε σχέση με τον κλασικό προγραμματισμό όσο αφορά στην εξόρυξη δεδομένων; Στο κλασσικό προγραμματισμό ο χρήστης είναι αυτός που εισάγει τους κανόνες και τα δεδομένα τα οποία επεξεργάζονται σύμφωνα με τους κανόνες που ο ίδιος έχει ορίσει. Σε αντίθεση στην μηχανική μάθηση ο χρήστης εισάγει τα δεδομένα καθώς και τις απαντήσεις που αναμένονται από τα δεδομένα και παίρνει σαν αποτέλεσμα τους κανόνες οι οποίοι μπορούν να εφαρμοστούν σε νέα δεδομένα για την παραγωγή πρωτότυπων απαντήσεων.



Σχήμα 1.2: Τρόπος λειτουργίας κλασικού προγραμματισμού και μηχανικής μάθησης

1.3 Η Διαδικασία της εξόρυξης δεδομένων

Η διαδικασία της εξόρυξης δεδομένων είναι η ανακάλυψη μοτίβων, σχέσεων και πληροφοριών μέσα από τεράστια σύνολα δεδομένων τα οποία καθοδηγούν τις επιχειρήσεις να μετρήσουν και να διαχειριστούν την κατάσταση τους αλλά και να προβλέψουν που θα βρίσκονται στο μέλλον.



Σχήμα 1.3: Η διαδικασία της εξόρυξης δεδομένων. [**?**]

1. Επιχειρηματική κατανόηση : Διερεύνηση των επιχειρηματικών στόχων και απαιτήσεων, αποφασίζοντας εαν η εξόρυξη δεδομένων μπορεί να εφαρμοστεί για να ικανοποιήσει και να καθοριστεί τι είδους δεδομένα μπορούν να συλλεχθούν για να δημιουργηθεί ένα αναπτυσσόμενο μοντέλο.

2. Κατανόηση δεδομένων : Καθορίζεται ένα αρχικό σύνολο δεδομένων και μελετάται για να διαπιστωθεί αν είναι κατάλληλο για περαιτέρω επεξεργασία. Εαν η ποιότητα των δεδομένων είναι κακή μπορεί να χρειαστεί να συλλεχθούν νέα δεδομένα με χρήση αυστηρότερων κριτηρίων.

3. Προετοιμασία δεδομένων : Η προετοιμασία συνεπάγεται την προεπεξεργασία των αρχικών δεδομένων έτσι ώστε οι αλγόριθμοι μηχανικής μάθησης να μπορούν να παράγουν το ιδανικό μοντέλο. Αυτό το στάδιο είναι και το ποιό χρονοβόρο αφού εδώ παράγεται το τελικό σύνολο δεδομένων αφού πρώτα επιλεγούν καθαριστούν, κατασκευαστούν και μορφοποιηθούν στην επιθυμητή μορφή.

4. Μοντελοποίηση : Αρχικά επιλέγονται οι τεχνικές μοντελοποίησης που θα χρησιμοποιηθούν, μετά πρέπει να δημιουργηθεί ένα σενάριο δοκιμής για την επικύρωση της ποιότητας και εγκυρότητας του μοντέλου. Στην συνέχεια ένα ή περισσότερα μοντέλα δημιουργούνται στο προετοιμασμένο σύνολο δεδομένων ενώ στο τέλος τα μοντέλα αξιολογούνται έτσι ώστε να διασφαλιστεί ότι τα μοντέλα που δημιουργήθηκαν ικανοποιούν τις επιχειρηματικές πρωτοβουλίες.

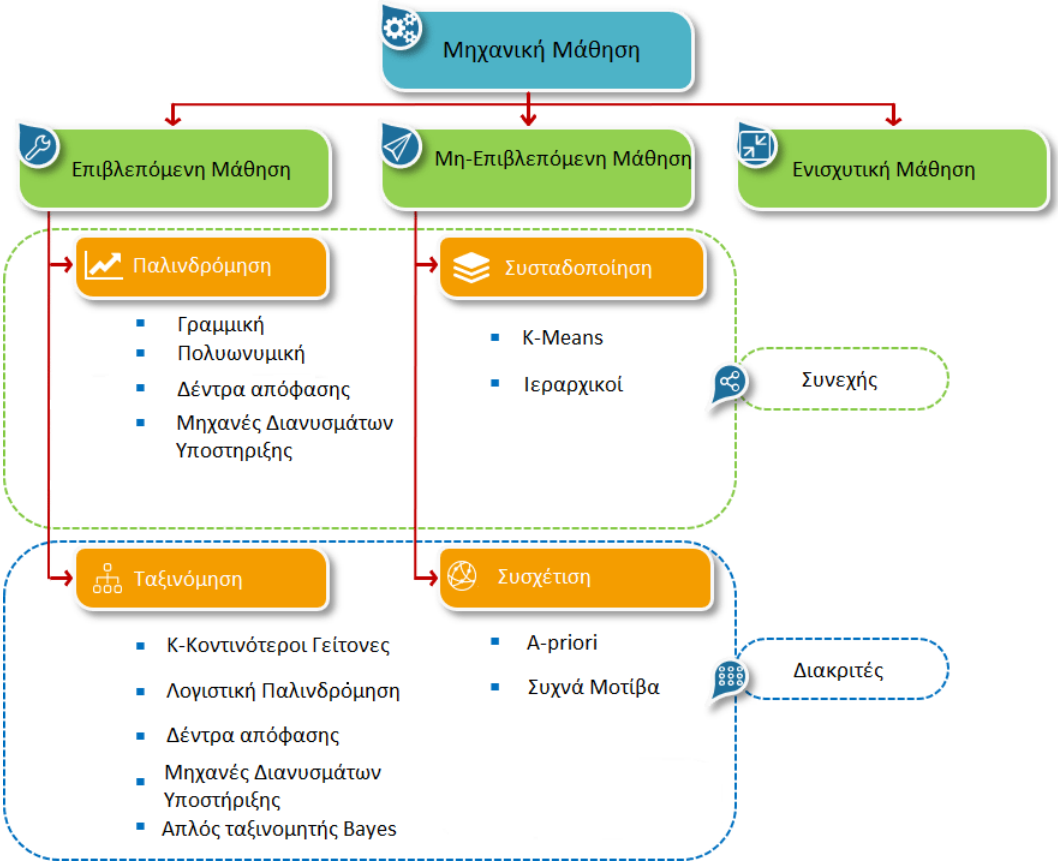
5. Εκτίμηση : Σε αυτή την φάση τα αποτελέσματα του μοντέλου αξιολογούνται στο πλαίσιο των επιχειρηματικών στόχων. Επίσης εδώ ενδέχεται να προκύψουν νέες επιχειρηματικές απαιτήσεις λόγω νέων προτύπων που ανακαλύφθηκαν από τα μοντέλα ή άλλους παράγοντες. Αν η φάση της εκτίμησης δείξει ότι το μοντέλο είναι φτωχό θα χρειαστεί να επανεξετάσει ολόκληρο το

πρότζεκτ ενώ αν η ακρίβεια του μόντελου είναι αρκετά ψηλή μπορεί να οδηγήσει στην κανονική του χρήση.

6. Ανάπτυξη : Οι γνώσεις ή οι πληροφορίες που αποκτώνται μέσω της διαδικασίας εξόρυξης δεδομένων πρέπει να παρουσιάστούν με τέτοιο τρόπο ώστε οι ενδιαφερόμενοι να μπορούν να τις χρησιμοποιήσουν όταν το θέλουν. Με βάση τις επιχειρηματικές απαιτήσεις, η φάση εγκατάστασης μπορεί να είναι τόσο απλή όσο η δημιουργία μιας παρουσίασης ή τόσο περίπλοκη όσο μια επαναλαμβανόμενη διαδικασία εξόρυξης δεδομένων σε ολόκληρο τον οργανισμό. Στη φάση της εγκατάστασης πρέπει να δημιουργηθούν τα σχέδια ανάπτυξης, συντήρησης και παρακολούθησης για την υλοποίηση και τις μελλοντικές υποστηρίξεις.

1.4 Είδη Μηχανικής Μάθησης

Τα κύρια είδη μηχανικής μάθησης είναι τρία, η επιβλεπόμενη μάθηση, η μη επιβλεπόμενη μάθηση και η ενισχυτική μάθηση.



Σχήμα 1.4: Είδη Μηχανικής Μάθησης

1.4.1 Επιβλεπόμενη μάθηση

Στην επιβλεπόμενη μάθηση (supervised learning), το σύνολο δεδομένων (dataset) είναι ένα σύνολο από επισημασμένα παραδείγματα $\{(x_i, y_i)\}_{i=1}^{N}$. Κάθε στοιχείο x_i απο τα N ονομάζεται χαρακτηριστικό διάνυσμα. Ενα χαρακτηριστικό διάνυσμα είναι ένα διάνυσμα στο οποίο κάθε διάσταση $j =$

$1, .., D$ περιέχει μια τιμή η οποία περιγράφει κάπως το παράδειγμα. Αυτή η τιμή ονομάζεται χαρακτηριστικό (feature) και συμβολίζεται σαν $x^{(j)}$. Για όλα τα παράδειγμα των δεδομένων μας το χαρακτηριστικό στην θέση j στο χαρακτηριστικό μας διάνυσμα πάντα περιέχει το ίδιο είδος πληροφορίας. Παραδείγματος χαρη το $x_k^{(j)}$ και το $x_l^{(j)}$περιέχουν το ίδιο είδος πληροφορίας. Ο στόχος ή τιμή εξόδου (label) y_i μπορεί να είναι είτε ένα στοιχείο που ανήκει σε ένα πεπερασμένο σύνολο από κλάσεις $\{1, 2, ..., C\}$, ένας πραγματικός αριθμός, είτε να έχει μια πιο πολύπλοκη δομή όπως διάνυσμα, πίνακας, δένδρο ή γράφημα.

Ο στόχος των αλγορίθμων επιβλεπόμενης μάθησης είναι να χρησιμοποιήσει τα δεδομένα για να παράξει ένα μοντέλο το οποίο παίρνει ένα χαρακτηριστικό διάνυσμα x σαν δεδομένα εισόδου και εξάγει πληροφορίες που επιτρέπουν να συμπεράνεις την τιμή εξόδου για το συγκεκριμένο χαρακτηριστικό διάνυσμα.

1.4.2 Μη Επιβλεπόμενη μάθηση

Στην μη επιβλεπόμενη μάθηση (Unsupervised Learning) το σύνολο δεδομένων είναι μια συλλογή απο μη επισημασμένα παραδείγματα $\{x_i\}_{\iota=1}^N$. Όπως και στην επιβλεπόμενη μάθηση το x είναι το χαρακτηριστικό διάνυσμα, και ο στόχος του αλγόριθμου μη επιβλεπόμενης μάθησης είναι να δημιουργήσει ένα μοντέλο το οποίο θα παίρνει ένα χαρακτηριστικό διάνυσμα x σαν δεδομένα εισόδου και είτε θα το μεταμορφώνει σε ένα άλλο διάνυσμα είτε σε

μια τιμή η οποία θα μπορεί να χρησιμοποιηθεί για να ληθεί κάποιο πρακτικό πρόβλημα. Για παράδειγμα, στην συσταδοποίηση (clustering) το μοντέλο επιστρέφει την ταυτότητα του συμπλέγματος για κάθε χαρακτηριστικό διάνυσμα στο σύνολο των δεδομένων. Στην διαστατική μείωση (dimensionality reduction) το προϊόν εξόδου του μοντέλου είναι ένα χαρακτηριστικό διάνυσμα το οποίο έχει λιγότερα χαρακτηριστικά από το αρχικό διάνυσμα x. Στην ανίχνευση (outlier detection), το προϊόν εξόδου είναι ένας πραγματικός αριθμός το οποίο υποδηλώνει πως το x είναι διαφορετικό από τα "τυπικα" παραδείγματα στο σύνολο δεδομένων.

1.4.3 Ενισχυτική μάθηση

Ενισχυτική μάθηση (Reinforcement Learning) είναι ένα είδος μηχανικής μάθησης όπου η μηχανή "ζει" μέσα σε ένα περιβάλλον και είναι ικανή να αντιλαμβάνεται την κατάσταση (state) του περιβάλλοντος σαν ενα διάνυσμα χαρακτηριστικών. Η μηχανή μπορεί να εκτελέσει δράσεις (actions) σε κάθε κατάσταση. Διαφορετικές δράσεις φέρνουν διαφορετικές ανταμοιβές (rewards) και μπορούν να μεταφέρουν την μηχανή σε άλλη κατάσταση περιβάλλοντος. Ο στόχος ενός αλγορίθμου ενισχυτικής μάθησης είναι να μάθει μια τακτική (policy).

Η τακτική είναι μια συνάρτηση η οποία παίρνει το χαρακτηριστικό διάνυσμα μιας κατάστασης σαν δεδομένο εισόδου και εξάγει την βέλτιστη δράση που μπορεί να εκτελεστεί στην υπάρχουσα κατάσταση. Η δράση είναι βέλ-

τιστη εάν μεγιστοποιεί την αναμενόμενη μέση ανταμοιβή (expected average reward).

Η ενισχυτική μάθηση λύνει ένα συγκεκριμένο είδος προβλήματος όπου η λήψη αποφάσεων είναι διαδοχική, και ο στόχος είναι μακροπρόθεσμος όπως σε ηλεκτρονικά παιχνίδια, τη ρομποτική, τη διαχείριση πόρων ή την εφοδιαστική.

1.4.4 Είδη Υβριδικής μάθησης

Οι γραμμές ανάμεσα στην επιβλεπόμενη και τη μη επιβλεπόμενη μάθηση δεν είναι τόσο αυστηρές και έτσι υπάρχουν αρκετές υβριδικές προσεγγίσεις που αντλούν διαδικασίες και από τους δύο τομείς.

Στην ημι-επιβλεπόμενη μάθηση το σύνολο δεδομένων είναι μια συλλογή από επισημασμένα και μη επισημασμένα παραδείγματα. Συνήθως η ποσότητα των μη επισημασμένων παραδειγμάτων είναι πολύ μεγαλύτερη από αυτή των επισημασμένων. Ο στόχος του αλγόριθμου ημί-επιβλεπόμενης μάθησης είναι ο ίδιος με αυτόν του αλγορίθμου επιβλεπόμενης μάθησης. Η ελπίδα εδώ είναι ότι χρησιμοποιόντας περισσότερα μη επισημασμένα παραδείγματα θα βοηθήσει τον αλγόριθμο να δημιουργήσει ένα καλύτερο μοντέλο.

Η Αυτοελεγχόμενη μάθηση (Self-supervised Learning) αναφέρεται σε ένα μη-επιβλεπόμενης μάθησης πρόβλημα το οποίο αντιμετωπίζεται σαν ένα πρόβλημα επιβλεπόμενης μάθησης στο οποίο το σύνολο δεδομένων δεν έχει έτοιμες ετικέτες αλλά τα δεδομένα τις δημιουργούν αυτόματα βρίσκοντας και χρησιμοποιόντας τις σχέσεις ανάμεσα στα διάφορα δεδομένα εισόδου.

Άλλα είδη μηχανικής μάθησης είναι τα Feature learning, Sparse dictionary learning, Anomaly detection, rule-based machine learning κτλπ..

1.5 Μαθηματικό υπόβαθρο

1.5.1 Γραμμική Άλγεβρα

Ο κλάδος των μαθηματικών που θα χρησιμοποιήσουμε περισσότερο σε αυτή την εργασία είναι η Γραμμική Άλγεβρα. Η γραμμική άλγεβρα είναι τα μαθηματικά των διανυσμάτων και των πινάκων. Συμβολίζουμε το διάνυσμα για παράδειγμα σε τρεις διαστάσεις ως

$$\overrightarrow{u} = (u_1, u_2, u_3) \ \in \ (\mathbb{R}, \mathbb{R}, \mathbb{R}) \equiv \mathbb{R}^3$$

Ένας πίνακας $A \in \mathbb{R}^{m \times n}$ είναι ένας ορθογώνιος πίνακας που αποτελειται από m γραμμές και n στήλες. Για παράδειγμα, ένας πίνακας 3×2 θα μοιάζει με

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \in \mathbb{R}^{3 \times 2}$$

με a_{ij} το στοιχείο που βρίσκεται στην i γραμμή και j στήλη.

Μερικές από τις πιό σημαντικές ιδιότητες τον διανυσμάτων είναι οι εξής

- $\overrightarrow{u} \pm \overrightarrow{v} = (u_1 \pm v_1, u_2 \pm v_2, u_3 \pm v_3)$
- $a\overrightarrow{u} = (au_1, au_2, au_3)$
- $\| \overrightarrow{u} \| = \sqrt{u_1^2 + u_2^2 + u_3^2}$

- $\overrightarrow{u} \cdot \overrightarrow{v} = u_1v_1 + u_2v_2 + u_3v_3$
- $\overrightarrow{u} \times \overrightarrow{v} = (u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1)$

Όσον αφορά τους πίνακες αρκετά χρήσιμοι θα είναι οι παρακάτω συμβολισμοί. Έστω ότι έχουμε ένα πίνακα X διάστασεων $n \times p$ και x_{ij} το στοιχείο της i γραμμής και j στήλης

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

μπορούμε να συμβολίσουμε την i στήλη σαν ένα διάνυσμα στήλης μήκους p ως

$$x_i = \begin{pmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{pi} \end{pmatrix}$$

και έτσι ο αρχικός πίνακας X να έχει πλέον την μορφή

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_p \end{pmatrix}$$

ή με την ίδια λογική χρησιμοποιόντας τις γραμμές σαν διανύσματα ο πίνακας να έχει την μορφή

$$X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}$$

όπου ο $^{\mathrm{T}}$ συμβολισμός δηλώνει την αναστροφή ενός διανύσματος ή ενός πίνακα όπου στον πίνακα οι γραμμες γίνονται στήλες και οι στήλες γίνονται γραμμές με

$$X^T = \begin{pmatrix} x_{11} & x_{21} & \dots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{pn} \end{pmatrix}$$

Οι μαθηματικές πράξεις που ορίζονται για πίνακες είναι οι ακόλουθες:

- $C = A \pm B \iff c_{ij} = a_{ij} \pm b_{ij}$
- $C = mA \iff c_{ij} = m * a_{ij}$
- Ο πολλαπλασιασμός των πινάκων $A \in \mathbb{R}^{m \times n}$ και $B \in \mathbb{R}^{n \times l}$ μας δίνει έναν πίνακα $C \in \mathbb{R}^{m \times l}$ ο οποίος δίνεται από τον τύπο

$$C = AB \iff c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} \end{bmatrix}$$

ενώ αρκετά σημαντικό είναι να αναφέρουμε ότι $AB \neq BA$

Επίσης υπάρχει και ο αντίστροφος πίνακας A^{-1} ενός πίνακα A για τον οποίον ισχύουν οι ιδιότητες

- $AA^{-1} = A^{-1}A = I$
- $(AB)^{-1} = B^{-1}A^{-1}$

με I τον μοναδιαίο τετράγωνο πίνακα που έχει μονάδα στην διαγώνιο του και μηδέν οπουδήποτε αλλού

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Τέλος κάποιες χρήσιμες ιδιότητες είναι οι εξής

- $(AB)^T = B^T A^T$
- $(A^{-1})^T = (A^T)^{-1}$
- $tr(A) = \sum_{i=1}^{n} a_{ii}$

1.5.2 Υπολογισμός Αποστάσεων

Σε αυτή την εργασία σε αρκετούς αλγορίθμους θα χρειαστούμε να υπολογίσουμε αποστάσεις ανάμεσα σε δύο σημεία. Οι σύνηθες φόρμουλες για τον υπολογισμό των αποστάσεων είναι οι εξής:

- Ευκλείδια απόσταση : $d(x_i, y_i) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

- Απόσταση Manhattan : $d(x_i, y_i) = \sum_{i=1}^{n} |x_i - y_i|$
- Απόσταση Hamming : $D_H = \sum_{i=1}^{n} |x_i - y_i|$ όπου όταν $x = y \Rightarrow D = 0$ και όταν $x \neq y \Rightarrow D = 1$
- Απόσταση Minkowski : $d(x_i, y_i) = (\sum_{i=1}^{n} (|x_i - y_i|)^q)^{1/q}$ όπου για $q = 1$ θα έχουμε την απόσταση Manhattan και για $q = 2$ την ευκλείδια απόσταση.

1.5.3 Διαφορικός Λογισμός

Αρκετές φορές θα χρειαστούμε να βρούμε το κατάλληλο διάνυσμα για το οποίο η συνάρτηση μας θα παίρνει την ελάχιστη τιμή. Έτσι καλό θα ήταν να ορίσουμε την παράγωγο μιας συνάρτησης ως προς ένα διάνυσμα. Έστω η συνάρτηση $f(w)$ και ένα διάνυσμα $w = (w_1, w_2, ..., w_n)$. Θα έχουμε

$$\frac{\partial f}{\partial w} = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

ενώ χρήσιμες θα ήταν οι παρακάτω ιδιότητες

- $f(w) = w^T x \qquad \frac{\partial f}{\partial w} = x$
- $f(w) = x w^T \qquad \frac{\partial f}{\partial w} = x$
- $f(w) = w^T w \qquad \frac{\partial f}{\partial w} = 2w$
- $f(w) = w^T C w \qquad \frac{\partial f}{\partial w} = 2Cw$

2 Επιβλεπόμενη μάθηση

2.1 Πως λειτουργεί η Επιβλεπόμενη μάθηση

Σε ένα δοσμένο σύνολο δεδομένων N παρατηρήσεων $\{x_n\}$, όπου $n = 1, .., N$ μαζί με τις αντίστοιχες τιμές εξόδου $\{y_n\}$, ο στόχος είναι είναι να προβλέψουμε την τιμή του y για μια νέα τιμή του x. Στην απλούστερη προσέγγιση, αυτό μπορεί να γίνει με την άμεση κατασκευή μιας κατάλληλης συνάρτησης $f(x)$. Γενικότερα από μια πιο πιθανοκρατική προοπτική στοχεύουμε να μοντελοποιήσουμε την προβλεπόμενη κατανομή $P(y|x)$ διότι αυτό εκφράζει την αβεβαιότητα μας για την τιμή του y για κάθε τιμή του x. Από αυτή την υπό όρους κατανομή μπορούμε να κάνουμε προβλέψεις του y για κάθε νέα τιμή του x, με τέτοιο τρόπο έτσι ώστε να μπορέσουμε να ελαχιστοποιήσουμε την αναμενόμενη τιμή μιας κατάλληλα επιλεγμένης συνάρτησης απώλειας (Loss function).

Ταξινόμηση VS Παλινδρόμηση

Η Ταξινόμηση (Classification) είναι ένα πρόβλημα αυτόματης ανάθεσης μιας ετικέτας σε ένα παράδειγμα που δεν έχει ετικέτα. Η ανίχνευση των spam e-mails είναι ένα διάσημο παράδειγμα ταξινόμησης. Σε ένα πρόβλημα ταξινόμησης μια ετικέτα είναι μέρος ενός πεπερασμένου συνόλου τάξεων. Εαν το μέγεθος του συνόλου των τάξεων είναι δύο τότε μιλάμε για μια διωνυμική ταξινόμηση (Binomial Classification). Για προβλήματα με περισσότερες από δύο κατηγορίες θα μιλάμε για προβλήματα πολυωνυμικής ταξινόμησης (Multinomial classification).

Η Παλινδρόμηση (Regression) είναι ένα πρόβλημα πρόβλεψης μιας πραγματικής τιμής ετικέτας για δοσμένο παράδειγμα που δεν έχει ετικέτα. Η εκτί-

μηση της τιμής μια κατοικίας με βάση τα χαρακτηριστικά του σπιτιού, όπως η περιοχή, ο αριθμός των υπνοδωματίων και λοιπά είναι ένα διάσημο παράδειγμα παλινδρόμησης.

2.2 Γραμμική Παλινδρόμηση

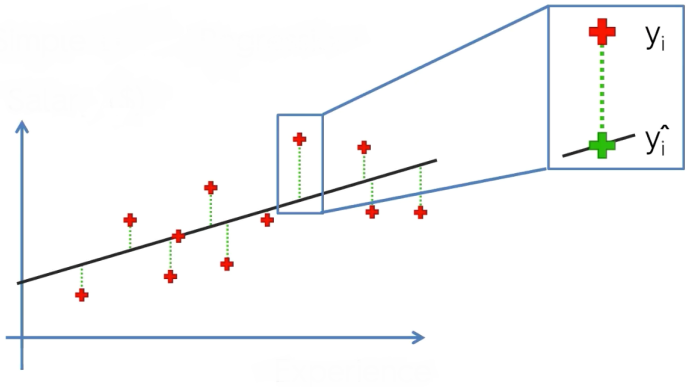
Το απλούστερο γραμμικό μοντέλο για την παλινδρόμηση είναι αυτό που περιλαμβάνει έναν γραμμικό συνδυασμό των μεταβλητών εισόδου.

Έστω ότι έχουμε μια συλλογή από επισημασμένα παραδείγματα $\{(x_i, y_i)\}_{i=1}^{n}$ όπου n είναι το μέγεθος της συλλογής. Θέλουμε να κατασκευάσουμε ένα μοντέλο Y ως γραμμικό συνδιασμό των x. Το μοντέλο όταν έχουμε εξάρτηση από μία μόνο μεταβλητή θα είναι της μορφής:

$$\hat{Y} = \beta_1 X + \beta_0$$

με $X = (x_1, x_2, .., x_{\mathrm{n}})^T$ το διάνυσμα των μεταβλητών εισόδου και $\hat{Y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)^T$ το διάνυσμα των προβλεπόμενων τιμων εξόδου. Θα χρησιμοποιήσουμε το μοντέλο για να προβλέψουμε ένα άγνωστο y όταν μας δίνεται ένα νέο x. Στόχος μας είναι χρησιμοποιόντας τα επισημασμένα παραδείγματα να παράξουμε εκτιμήσεις για τα β_1και β_0τα οποία θα μας οδηγήσουν σε όσο το δυνατό καλύτερη πρόβλεψη του y. Για να επιτύχουμε την βέλτιστη εκτίμηση στόχος μας είναι να ελαχιστοποιήσουμε το σφάλμα μεταξύ εκτιμόμενης και πραγματικής τιμής. Υπάρχουν πολλοί μεθόδοι εκτίμησης σφάλματος αλλά ο πιό διαδεδομένος είναι η μέθοδος των ελαχίστων τετραγώνων. Έστω $\hat{y}_i = \beta_1 x_i + \beta_0$ η πρόβλεψη του y_i για την τιμη του x_i, τότε η συνάρτηση

απώλειας (Loss function) θα είναι η $e_i = y_i - \hat{y}_i$ που μας δίνει το σφάλμα του i δείγματος.



Σχήμα 2.1: Σφάλμα

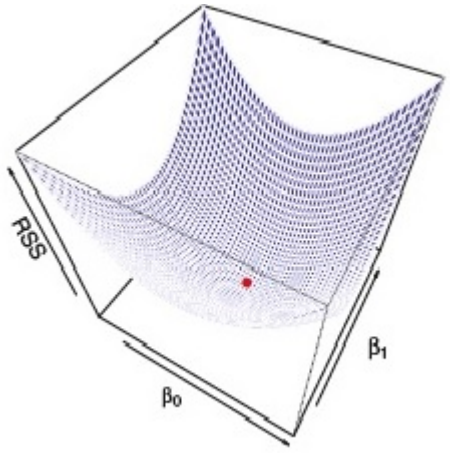
Άρα το άθροισμα των τετραγώνων των σφαλμάτων (RSS-residual sum of squares) θα ειναι:

$$RSS = e_1^2 + e_2^2 + ...e_n^2 = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2$$

Χρησιμοποιόντας απειροστικό λογισμό για να βρούμε το ελάχιστο του RSS καταλήγουμε στα εξης αποτελέσματα:

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$



Σχήμα 2.2: Ελάχιστη τιμή του RSS [22]

με $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ και $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ οι μέσες τιμές των δειγμάτων.

Τέλος για σύναρτηση κόστους (Cost function) θα χρησιμοποιήσουμε το μέσο τετραγωνικό σφάλμα (mean square error ή MSE) θα είναι

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}RSS$$

Με την ίδια λογική κατασκευάζουμε ενα μοντέλο γραμμικής παλινδρόμησης αν έχουμε παραπάνω από μία μεταβλητές:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

όπου $x_i = (x_{i1}, x_{i2}, ..., x_{ip})^T$.

Η παραπάνω εξίσωση γίνεται πιό απλή με την χρήση πινάκων

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

ή $\hat{Y} = X\beta$

με το άθροισμα των τετραγώνων των σφαλμάτων:

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{p}\beta_j x_{ij}))^2 = (Y - X\beta)^2$$

Ενώ με την μορφή πινάκων και για $Y = (y_1, y_2, ..., y_n)^T$το RSS θα πάρει την μορφή

$$RSS = (Y - X\beta)^T(Y - X\beta)$$

παίρνοντας την πρώτη και δεύτερη μερική παράγωγο του RSS ως προς β μετά από εφαρμογή των τύπων του κεφαλαίου 1.5.3 θα πάρουμε

$$\frac{\partial RSS}{\partial \beta} = -2X^T(y - X\beta)$$

$$\frac{\partial^2 RSS}{\partial \beta} = 2X^T X$$

Υποθέτοντας ότι X θα έχει πλήρη τάξη στηλών θα έχουμε $X^T X$ θετικά ορισμένο, άρα μπορούμε να θέσουμε την πρώτη παράγωγο ίση με το μηδέν για να βρούμε το ελάχιστο και θα έχουμε

$$-2X^T(y - X\beta) = 0$$

$$\beta = (X^T X)^{-1}X^T$$

2.3 Πολυωνυμική Παλινδρόμηση

Η μέθοδος που ακολουθήσαμε στην απλή γραμμική παλινδρόμηση μπορεί να λειτουργήσει και για μεγαλύτερου βαθμου πολυώνυμα. Έστω ότι έχουμε N σημεία $\{(x_i, y_i)\}_{i=1}^{N}$ το μοντέλο για πολυώνυμο m-βαθμού θα έχει την μορφή:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_p x_i^m$$

με το άθροισμα των τετραγώνων των σφαλμάτων να δίνεται από τον τύπο:

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}[y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i^2 - ... + \beta_p x_i^m)]^2$$

ή μπορεί να γραφτεί με την μορφή πινάκων:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix}$$

για $y = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$, $\beta = (\beta_0, \beta_1, .., \beta_m)$ και X τον παραπάνω πίνακα, θα γραφτεί στην μορφή

$$Y = X\beta$$

πολλαπλασιάζοντας και τα δύο μέλη με τον X^Tθα έχουμε

$$X^T y = X^T X \beta$$

$$\beta = (X^T X)^{-1} X^T Y$$

2.4 Αξιολόγηση συνεισφοράς χαρακτηριστικών και ακρίβειας μοντέλου

Μετά την παρουσίαση του πρώτου μας μοντέλου, χρήσιμο θα ήταν να έχουμε και έναν μηχανισμο για να αξιολογήσουμε πόσο ακριβές είναι το μοντέλο μας, αλλά και πόσο σημαντική είναι η συνεισφορά του κάθε χαρακτηριστικού στην σωστή πρόβλεψη του μοντέλου μας. Υπάρχουν διάφοροι τρόποι να αξιολογήσουμε την ακρίβεια και την συνεισφορα των χαρακτηριστικών του μοντέλου μας. Το πρώτο που θα δούμε είναι το Τυπικό σφάλμα (Standard Error).

Σε γενικές γραμμές, το τυπικό σφάλμα μας δίνει τον μέσο όρο της διαφοράς ανάμεσα στις δύο μέσες τιμές $\hat{\mu}$ και μ, όπου μ και $\hat{\mu}$ η πραγματική και η προβλεπόμενη μέση τιμή της τυχαίας μεταβλητής Υ αντίστοιχα. Ο τύπος που μας δίνει το τυπικό σφάλμα στην γραμμική παλινδρόμηση με μία μεταβλητή είναι οι εξής

$$SE(\hat{\beta}_0)^2 = \sigma^2[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}]$$

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

όπου $\sigma^2 = Var(\epsilon)$. Για να είναι αυστηρά σωστοί αυτοί οι τύποι, πρέπει να υποθέσουμε ότι τα σφάλματα ϵ_i για κάθε παρατήρηση δεν συσχετίζονται και έχουν κοινή διακύμανση σ^2.

Γενικά το σ^2δεν είναι γνωστό, αλλά μπορεί να εκτιμηθεί από τα δεδομένα. Η εκτίμηση του σ είναι γνωστή ως Υπόλοιπο τυπικό σφάλμα (Residual Standard Error-RSE) και δίνεται από τον τύπο

$$RSE = \sqrt{\frac{RSS}{n-2}}$$

Αν χρησιμοποιήσουμε αυτό το σ^2 για να υπολογίσουμε το τυπικό σφάλμα συνήθως χρησιμοποιούμε το σύμβολο του καπέλου, δηλαδή $\hat{SE}(\hat{\beta}_1)$.

Έχοντας υπολογίσει το Τυπικό σφάλμα είναι εύκολο να βρούμε και τα διαστήματα εμπιστοσύνης. Για την γραμμική παλινδρόμηση το 95% διάστημα εμπιστοσύνης για το $\hat{\beta}_1$δίνεται από τον τύπο $\hat{\beta}_1 \pm 2 \cdot SE(\hat{\beta}_1)$ και έτσι το διάστημα εμπιστοσύνης θα είναι το

$$[\hat{\beta}_1 - 2 \cdot SE(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot SE(\hat{\beta}_1)]$$

και ομοίως για το β_0

$$[\hat{\beta}_0 - 2 \cdot SE(\hat{\beta}_0), \hat{\beta}_0 + 2 \cdot SE(\hat{\beta}_0)]$$

Ένα αρκετά χρήσιμο στατιστικό είναι το t-στατιστικο (t-statistic) το οποίο μετράει τον αριθμό των τυπικών σφαλμάτων του β_1από το 0 και δίνεται από τον τύπο

$$t = \frac{\beta_1 - 0}{SE(\hat{\beta}_1)}$$

Εαν δεν υπάρχει πραγματική σχέση ανάμεσα στα X και Y τότε περιμένουμε η παραπάνω σχέση να έχει t-κατανομή με $n-2$ βαθμούς ελευθερίας. Η κατανομή έχει σχήμα καμπάνας και για τιμές του n μεγαλύτερες από 30 είναι αρκετά παρόμοια με την κανονική κατανομή.

Μια από τις σημαντικότερες μετρήσεις όσο αφορά το μοντέλο μας είναι η p-τιμή (p-value). Ουσιαστικά η τιμή αυτή είναι η πιθανότητα του πόσο το συγκεκριμένο χαρακτηριστικό να μην βοηθάει στην πρόβλεψη του μοντέλου μας. Άρα όσο χαμηλότερη είναι η τιμή τόσο χρησιμότερο είναι το συγκεκριμένο χαρακτηριστικό στην πρόβλεψη του μοντέλου μας.

Για να υπολογίσουμε την p-τιμή αρκεί να υπολογίσουμε την πιθανότητα του κάθε χαρακτηριστικού να ισούτε με το $|t|$ ή να είναι μεγαλύτερη κατά απόλυτη τιμή αν υποθέσουμε ότι το $\beta_1 = 0$.

Όλα τα παραπάνω που αναφέραμε έχουν να κάνουν κυρίως με την συνεισφορά του εκάστοτε χαρακτηριστικού στην πρόβλεψη του μοντέλου μας, αλλά είναι φυσικό να θέλουμε να ποσοτικοποιήσουμε το βαθμό στον οποίο το μοντέλο μας ταιρίαζει στα δεδομένα.

Εκτός του τυπικού σφάλματος που αναφέραμε παραπάνω μια εναλλακτική λύση στο μέτρο προσαρμογής των δεδομένων μας είναι η R^2 στατιστική (R^2 statistic). Ο τύπος για τον υπολογισμό της R^2 στατιστικής είναι

$$R^2 = \frac{TSS-RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

με $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$

Ουσιαστικά το R^2 μας δίνει το ποσοστό μεταβλητότητας στο Y που μπορεί να εξηγηθεί χρησιμοποιόντας το X.

Επίσης καλό είναι να αναφέρουμε και τον τύπο της συσχέτισης

$$Cor(X,Y) = \frac{\sum_{\iota=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{\iota=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{\iota=1}^{n}(y_i-\bar{y})^2}}$$

Έτσι μπορούμε να χρησιμοποιήσουμε το $r = Cor(X, Y)$ αντί για το R^2 ώστε να εκτιμήστε την εφαρμογή του γραμμικού μοντέλου. Μπορούμε όμως εύκολα να δείξουμε ότι για το απλό γραμμικό μοντέλο $r^2 = R^2$. Αυτό όμως δεν θα ισχυεί για τα γραμμικά μοντέλα με πολλά χαρακτηριστικά. Εκεί είναι που χρησιμοποιούμε το R^2.

Τέλος, θα δούμε και την F-στατιστική, η οποία εξετάζει την μηδενική υπόθεση H_0 στην οποία $\beta_1 = \beta_2 = .. = \beta_p = 0$. Όταν η F-Στατιστική έχει τιμή πολύ μεγαλύτερη από ένα τότε είναι ασφαλές να πούμε ότι τουλάχιστον ένα χαρακτηριστικό συνεισφέρει στην σωστή πρόβλεψη του μοντέλου μας. Ο τύπος θα είναι

$$F = \frac{(TSS-RSS)/p}{RSS/(n-p-1)}$$

όπου p ο αριθμός των χαρακτηριστικών.

2.5 Υλοποίηση Γραμμικής/Πολυωνυμικής Παλινδρόμησης στην R

2.5.1 Γραμμική Παλινδρόμηση μιας μεταβλητής

Θα χρησιμοποιήσουμε την βάση δεδομένων Salary_Data. Η βάση έχει 30 καταχωρίσεις όπου στην κάθε μια αναφέρονται τα χρόνια εμπειρίας και ο μισθός. Σκοπός του μοντέλου μας είναι να προβλέψουμε τον κατάλληλο μισθό ενός υπαλλήλου με βάση τα χρόνια προυπηρεσίας.

Φορτώνουμε την βάση δεδομένων μας και την βιβλιοθήκη που θα χρειαστούμε για να κάνουμε τα γραφήματα μας.

```r
library(ggplot2) # Loading the plot library
# Loading the dataset
df <- read.csv("C:/Users/tit0v/R/PTUXIAKH/Salary_Data.csv")
```

Χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 2/1. Θέτουμε συγκεκριμένο αριθμό στο set.seed για να παράγουμε πάντα τα ίδια αποτελέσματα.

```r
set.seed(1234)
#Create indices (1 and 2) with 2/1 proportion to split the data
#ind==1 with probability 2/3 for train set and ind==2 with probability
#1/3 is for test set, so we filter the data using this two indices.
ind <- sample(2,nrow(df),replace=T,prob=c(2/3,1/3))
train<- df[ind==1,] #filtering the dataset to create the train set
test <- df[ind==2,] #filtering the dataset to create the test set
```

Δημιουργούμε το μοντέλο μας το οποίο θα εκπαιδευτεί στα δεδομένα εκπαίδευσης και θα προβλέπει την μισθό χρησιμοποιόντας τα χρόνια προυπηρεσίας.

```r
# Training the model to predict Salary using YearsExperience
lm_model <- lm(Salary~YearsExperience, data = train)
```

Οι λεπτομέρειες για το μοντέλο μας θα είναι:

```r
summary(lm_model) # Model details

##
```

```
## Call:
## lm(formula = Salary ~ YearsExperience, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -9167  -3847   -272   4493  10144
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      26462.1     2654.8   9.968 2.05e-09 ***
## YearsExperience   9555.7      467.9  20.422 2.47e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5944 on 21 degrees of freedom
## Multiple R-squared:  0.9521,Adjusted R-squared:  0.9498
## F-statistic: 417.1 on 1 and 21 DF,  p-value: 2.47e-15
```

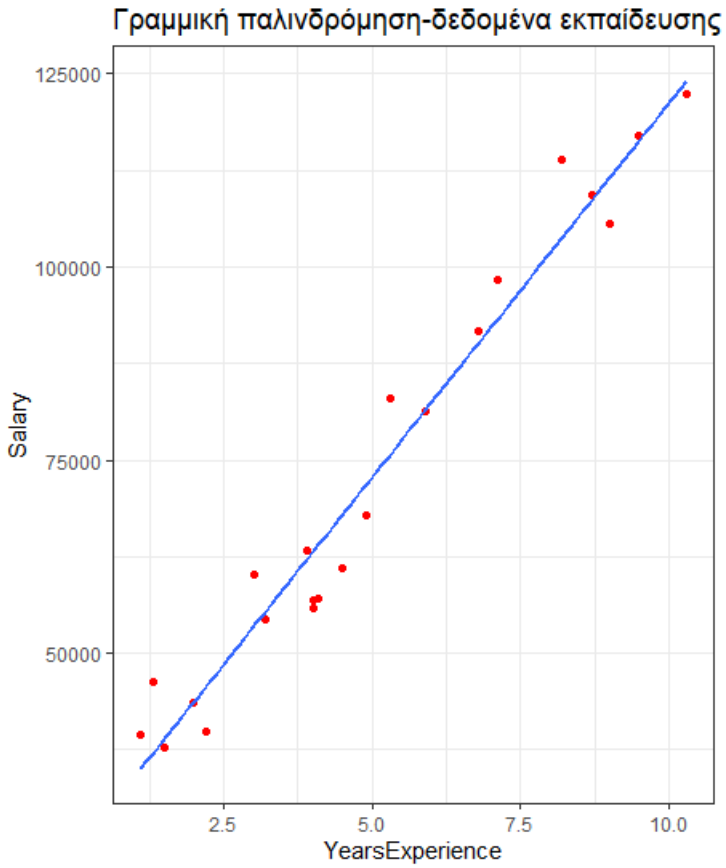
Σε αυτόν τον πίνακα μπορούμε να δούμε όλα τα στατιστικά τα οποία αναφέρθηκαν στο 2.4. Η p-τιμή είναι $2.47e-15$ η οποία είναι υπερβολίκα μικρή άρα μπορούμε να πούμε με βεβαιότητα ότι τα χρόνια εμπειρίας επηρεάζουν σημαντικά τον μισθό. Επίσης η F-Στατιστικη είναι 417, πολύ μεγαλύτερη από 1 άρα μπορούμε να απορρίψουμε την μηδενική υπόθεση H_0 και σίγουρα το $\beta_1 \neq 0$.

Χρησιμοποιόντας το μοντέλο που παράξαμε θα προβλέψουμε το μισθό για τα δεδομένα δοκιμής.

```
# Prediction on testset using the model we create
lm_pred <- predict(lm_model, newdata = test)
```

Ας παραστήσουμε γραφικά το μοντέλο μας σε σχέση με τα δεδομένα εκπαίδευσης

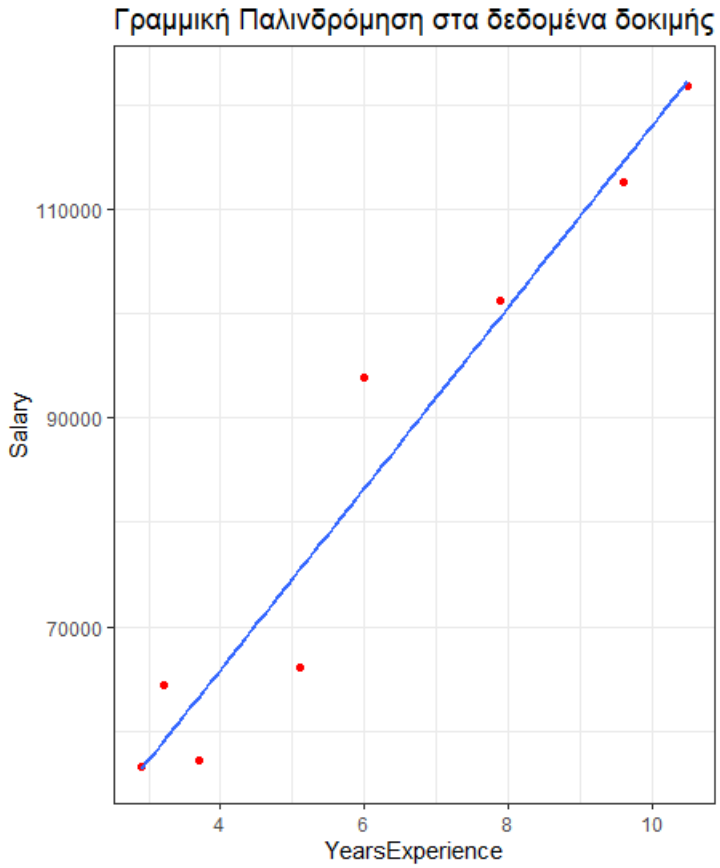
```
# Plotting the Linear regression model and the trainset
ggplot(train,aes(x=YearsExperience,y=Salary))+
geom_point(colour=I("Red")) +
geom_smooth(method="lm", se=F)+   theme_bw() +
ggtitle("Γ            -            ")
```



Βλέπουμε από τα δεδομένα μας ότι υπάρχει μια γραμμική συσχέτιση ανάμεσα στα χρόνια προυπηρεσίας και στον μισθό, έτσι η γραμμική παλινδρόμηση εφαρμόζει αρκετά καλά στα δεδομένα μας.

Τέλος για τα δεδομένα δοκιμής η γραφική παράσταση θα είναι

```
# Plotting the model we create to see how it fits
# on your test set
ggplot(test,aes(x=YearsExperience,y=Salary))+
geom_point(colour=I("Red")) +
geom_smooth(method="lm", se=F)+   theme_bw() +
ggtitle("Γ     Π                    ")
```



Όπως και στα δεδομένα εκπαίδευσης, έτσι και στα δεδομένα δοκιμών η γραμμική παλινδρόμηση εφαρμόζει αρκετά καλά και μας δίνει προβλέψεις κοντά στην πραγματική τιμή.

2.5.2 Γραμμική Παλινδρόμηση πολλών μεταβλητών

Θα χρησιμοποιήσουμε την βάση δεδομένων 50_Startups. Η βάση έχει 50 καταχωρίσεις από 50 startup εταιρείες όπου στην κάθε μια αναφέρονται τα έξοδα για την έρευνα, την διοίκηση, το μάρκετινγκ και το σε ποιά πολιτεία βρίσκεται η εταιρεία. Σκοπός του μοντέλου μας είναι να προβλέψουμε το κέρδος που θα έχει η εταιρεία.

Φορτώνουμε την βάση δεδομένων, και χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 80/20. Θέτουμε συγκεκριμένο αριθμό στο set.seed για να παράγουμε πάντα τα ίδια αποτελέσματα.

```
# Loading the dataset
dataf <- read.csv("C:/Users/tit0v/R/PTUXIAKH/50_Startups.csv")
set.seed(1234)
# creating the train and test set
ind <- sample(2,nrow(dataf),replace=T,prob=c(0.8,0.2))
train<- dataf[ind==1,]
test <- dataf[ind==2,]
```

Παράγουμε το μοντέλο γραμμικής παλινδρόμησης όπου θα χρησιμοποιήσουμε 4 μεταβλητές για να προβλέψουμε το κέρδος της εταιρείας. Το μοντέλο θα εκπαιδευτεί στα δεδομένα εκπαίδευσης. Επίσης εκτυπώνουμε και τις λε-

πτομέρειες του μοντέλου μας.

```
# Building the model to predict Profit using R.D.Spend, Administration
# Marketing.Spend and State as features.
mlm_model <- lm(Profit~R.D.Spend+Administration+Marketing.Spend+State,
data = train)
summary(mlm_model) # model details

##
## Call:
## lm(formula = Profit ~ R.D.Spend + Administration + Marketing.Spend +
##     State, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31832  -5261   -621   6254  17604
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.851e+04  7.798e+03   6.221 3.52e-07 ***
## R.D.Spend        8.169e-01  5.040e-02  16.209  < 2e-16 ***
## Administration  -2.837e-02  6.048e-02  -0.469    0.642
## Marketing.Spend  2.919e-02  1.846e-02   1.582    0.122
## StateFlorida     9.108e+02  3.702e+03   0.246    0.807
## StateNew York    1.458e+03  3.637e+03   0.401    0.691
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 9533 on 36 degrees of freedom
## Multiple R-squared:  0.9557,Adjusted R-squared:  0.9495
## F-statistic: 155.2 on 5 and 36 DF,  p-value: < 2.2e-16
```

Η τιμή p-value για το χαρακτηριστικό R.D.Spend είναι $2e-16$ και αυτό μας δείχνει ότι παίζει σημαντικό ρόλο στην πρόβλεψη του κέρδους σε αντίθεση με την πολιτεία στην οποία βρίσκεται η εταιρία που φαίνεται να μην επηρεάζει την πρόβλεψη του μοντέλου. Επίσης η F-Στατιστικη είναι 155, πολύ μεγαλύτερη από 1 άρα μπορούμε να απορρίψουμε την μηδενική υπόθεση H_0 και σίγουρα το $\beta_i \neq 0$.

Χρησιμοποιόντας το μοντέλο μας προβλέπουμε την τιμή του κέρδους για τις εταιρείες στα δεδομένα δοκιμής. Τέλος δημιουργούμε ένα πίνακα με τις προβλεπόμενες τιμές, τις πραγματικές τιμές και την διαφορά της προβλεπόμενης τιμής απο την πραγματική.

```
mlm_pred <- predict(mlm_model,newdata = test) # prediction
# Table with predicted real and difference values
tbl<-data.frame(prediction=mlm_pred,real=test$Profit,
difference=mlm_pred-test$Profit)
knitr::kable(tbl)
```

	prediction	real	difference
5	173613.92	166187.94	7425.976
14	127197.40	134307.35	-7109.948
16	147693.43	129917.04	17776.387
26	101408.62	107404.34	-5995.720
28	115562.01	105008.31	10553.701
29	101651.29	103282.38	-1631.091
39	70035.41	81229.06	-11193.653
40	82767.86	81005.76	1762.098

2.5.3 Πολυωνυμική Παλινδρόμηση

Θα χρησιμοποιήσουμε ως βάση δεδομένων το Position_Salaries.csv. Η βάση περιέχει τους μισθούς μιας εταιρίας ανάλογα με τον επίπεδο και τον τίτλο της θέσης. Σκοπός μας είναι να δημιουργήσουμε ένα μοντέλο που θα προβλέπει τον μισθό ενός υπαλλήλου με βάση την θέση του.

Φορτώνουμε την βάση δεδομένων και τις βιβλιοθήκες που θα χρειαστούμε. Αφαιρούμε την πρώτη στήλη μιας και έχουμε την πληροφορία από το αριθμό του επιπέδου.

```
# Loading the plot library and dataset
library(ggplot2)
datas <- read.csv("C:/Users/tit0v/R/PTUXIAKH/Position_Salaries.csv")
dataset <- datas[,2:3] # filtering the features we gonna use
```

Δημιουργούμε τα μοντέλα. Τα μοντέλα θα προβλέπουν τον μισθό με βάση το επίπεδο της δουλειάς. Το πρώτο μοντέλο είναι το γραμμικό, το δεύτερο

είναι το μοντέλο δευτέρου βαθμού και το τρίτο θα είναι το μοντέλου τρίτου βαθμού και το τελευταίο το μοντέλο τετάρτου βαθμού.

```
# Linear model
poly1_model <- lm(Salary~Level,data=dataset)
# Quadratic model
poly2_model <- lm(Salary~poly(Level, 2, raw = TRUE), data=dataset)
# Cubic model
poly3_model <- lm(Salary ~ poly(Level, 3, raw = TRUE), data = dataset)
# fourth degree model
poly4_model <- lm(Salary ~ poly(Level, 4, raw = TRUE), data = dataset)
```

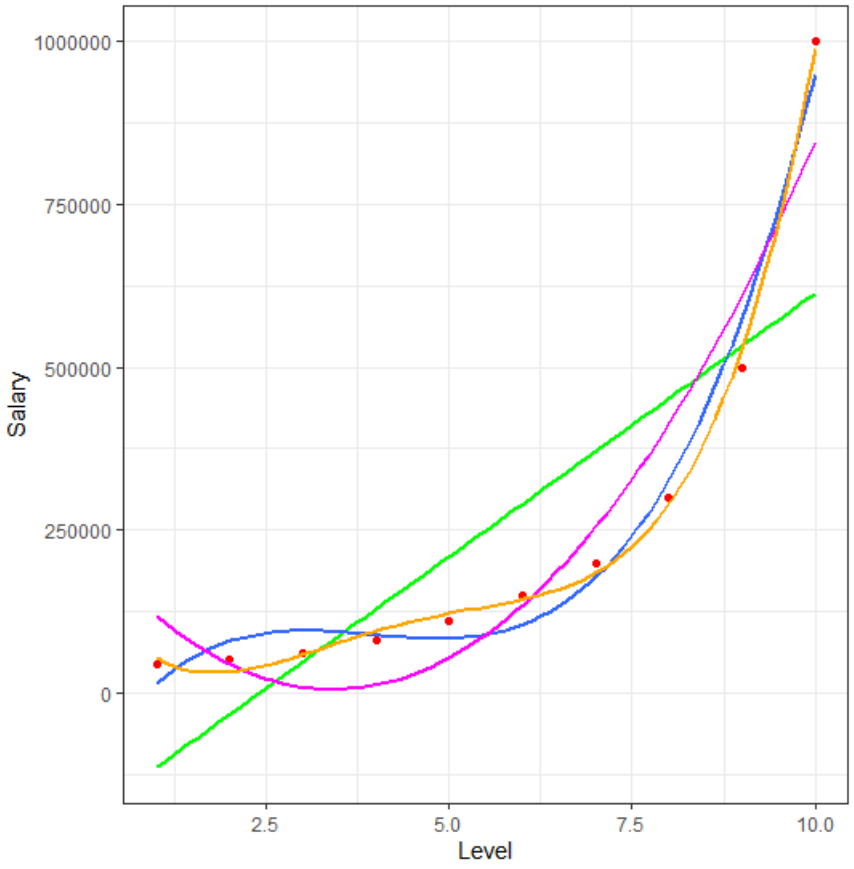
Χρησιμοποιόντας τα τρια μοντέλα θα προβλέψουμε τον μισθό για μια θέση επιπέδου 6.5 και θα τους παρουσιάσουμε συνολικά σε έναν πίνακα.

```
# Predictions for every model
poly1_pred <- predict(poly1_model, data.frame(Level=6.5))
poly2_pred <- predict(poly2_model, data.frame(Level=6.5))
poly3_pred <- predict(poly3_model, data.frame(Level=6.5))
poly4_pred <- predict(poly4_model, data.frame(Level=6.5))
# Creating the comparison table
tbl123 <- data.frame(Model=c("Linear","Quadratic","Cubic",
"Fourth degree"),Prediction=c(poly1_pred,poly2_pred,
poly3_pred,poly4_pred))
knitr::kable(tbl123)
```

Model	Prediction
Linear	330378.8
Quadratic	189498.1
Cubic	133259.5
Fourth degree	158862.5

Τέλος θα παραστήσουμε γραφικά την βάση δεδομένων μας και τα τρια μοντέλα. Με πράσινο θα είναι το γραμμικό, με ροζ το μοντέλο δευτέρου βαθμού,με μπλε το μοντέλο τρίτου βαθμού και με πορτοκαλί το μοντέλο τετάρτου βαθμού. Βλέπουμε πως για μεγαλύτερο βαθμό έχουμε και καλύτερες προβλέψεις.

```
# Plotting the three models and the dataset
ggplot(dataset,aes(x=Level,y=Salary))+geom_point(colour=I("Red")) +
geom_smooth(method="lm", se=F, colour="Green")+   theme_bw() +
geom_smooth(method="lm", formula = y ~ poly(x, 3), se = F) +
geom_smooth(method="lm",formula = y ~ poly(x, 2),se=F,colour="Magenta") +
geom_smooth(method="lm",formula = y ~ poly(x, 4),se=F,colour="orange")
```



2.6 Αλγόριθμος απότομης καθόδου

Θα έχετε παρατηρήσει από τα προηγούμενα κεφάλαια ότι κάθε αλγόριθμος αποτελείται από τρία μέρη:

1. μία συνάρτηση απώλειας
2. ένα κριτήριο βελτιστοποίησης που βασίζεται στη συνάρτηση απώλειας (για παράδειγμα μια συνάρτηση κόστους)
3. μία ρουτίνα βελτιστοποίησης αξιοποιώντας τα δεδομένα εκπαίδευσης

για να βρεθεί μια λύση στο κριτήριο βελτιστοποίησης

Σε μερικούς αλγόριθμους η εύρεση της βέλτιστης λύσης είναι εύκολη υπόθεση. Όπως είδαμε στην γραμμική παλινδρόμηση ήταν αρκετό το να βρούμε τις λύσεις της πρώτης παραγώγου για να ελαχιστοποιήσουμε την συνάρτηση κόστους. Αυτό όμως ήταν εφικτό γιατί η συνάρτηση κόστους ήταν κυρτή με ολικό ελάχιστο. Υπάρχουν όμως αρκετοί αλγόριθμοι που δεν έχουν κυρτή συνάρτηση κόστους και τα ελάχιστα είναι περισσότερα από ένα, αυτό κάνει την εύρεση του ολικού ελάχιστου αρκετά δυσκολότερο εγχείρημα.

Οι πίο διαδεδομένοι αλγόριθμοι βελτιστοποίησης είναι ο αλγόριθμος απότομης καθόδου(Gradient descent ή steepest descent) και ο αλγόριθμος στοχαστικής απότομης καθόδου(stohastic gradient descent).

Στον αλγόριθμο απότομης καθόδου έστω ότι ξεκινάμε από ένα τυχαίο σημείο της 'επιφάνειας' της πολυδιάστατης συνάρτησης κόστους. Ακολουθώντας την διεύθυνσης της ταχύτερης καθόδου βρίσκουμε το επόμενο σημείο. Επαναλαμβάνοντας την παραπάνω διαδικασία όσες φορές χρειαστεί θα φτάσουμε σε ένα τοπικό ελάχιστο. Αν η συνάρτηση κόστους είναι κυρτή τότε το ελάχιστο που θα βρούμε θα είναι το τοπικό ελάχιστο και επομένως η βέλτιστη λύση μας.

Ας δούμε πως λειτουργεί ο αλγόριθμος απότομης καθόδου ως αλγόριθμος βελτιστοποίησης στην γραμμική παλινδρόμηση. Η συνάρτηση κόστους θα είναι η εξής:

$$MSE = l = \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2$$

Πρώτα πρέπει να βρούμε τις μερικές πρώτες παραγώγους του l ως προς w και b.

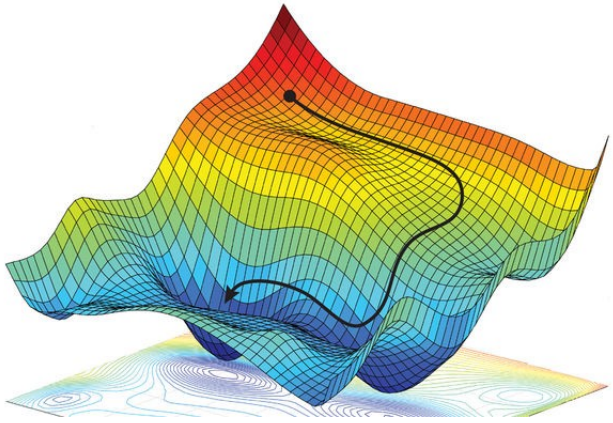
$$\frac{\partial l}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (wx_i + b))$$

$$\frac{\partial l}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - (wx_i + b))$$

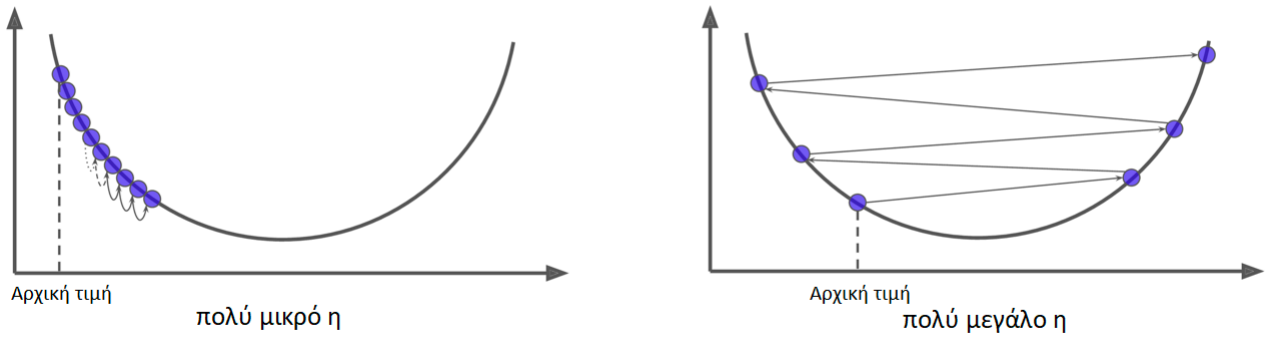
Όπως αναφέραμε και παραπάνω ο αλγόριθμος απότομης καθόδου στηρίζεται στις επαναλήψεις. Μπορούμε να ξεκινήσουμε με $w \leftarrow 0$ και $b \leftarrow 0$ και σε κάθε επανάληψη να υπολογίζουμε τα νεά w και b.

$$w \leftarrow w - \eta\frac{\partial l}{\partial w}$$

$$b \leftarrow b - \eta\frac{\partial l}{\partial b}$$



Σχήμα 2.3: Γραφική παράσταση αλγορίθμου απότομης καθόδου [17]



Σχήμα 2.4: Σύγκριση ρυθμού μάθησης

Με η συμβολίζουμε τον ρυθμό μάθησης (learning rate), ο οποίος ρυθμίζει το μέγεθος του βήματος άρα και το μέγεθος της αναβάθμισης του w και του b σε κάθε επανάληψη. Ο αλγόριθμος απότομης καθόδου είναι ευαίσθητος στην επιλογή του ρυθμού μάθησης. Για μικρό η θα χρειαστούν πολλές επαναλήψεις για να βρεθούμε στο ελάχιστο ενώ υπάρχει η περίπτωση να μην φτάσουμε ποτέ, ενώ για μεγάλο η πάλι υπάρχει η περίπτωση να μην βρούμε ποτέ το ελάχιστο.

Σύνοψη κεφαλαίου

Σε αυτό το πρώτο κεφάλαιο ορίσαμε τις διαφορές ανάμεσα στους αλγόριθμους παλινδρόμησης και ταξινόμησης σε προβλήματα επιβλεπόμενης μάθησης, επίσης παρουσιάσαμε το απλούστερο μοντέλο παλινδρόμησης, την γραμμική παλινδρόμηση καθώς επίσης και την πολυωνυμική παλινδρόμηση.

Αναφερθήκαμε ακόμα στα στατιστικά εργαλεία τα οποία χρησιμοποιούμε για να αξιολογήσουμε την συνεισφορά των εκάστοτε χαρακτηριστικών αλλά και την ακρίβεια του μοντέλου μας, στοιχεία τα οποία μας βοηθούν επίσης

στην βελτίωση της απόδοσης του μοντέλου μας.

Έγινε μια παρουσίαση του αλγόριθμου απότομης καθόδου, ενός αλγορίθμου αρκετά χρήσιμου για την βελτιστοποίηση των μοντέλων μας και την μείωση του σφάλματος ειδικά όταν έχουμε να κάνουμε με συναρτήσεις που είναι δύσκολο να βρεις την ελάχιστη τιμή, πράγμα σημαντικό στην λειτουργία πολλών μοντέλων μηχανίκης μάθησης

Τέλος εφαρμόσαμε τους αλγόριθμους του κεφαλαίου σε μία βάση δεδομένων με την χρήση του προγράμματος R. Αυτό που παρατηρήσαμε από το γράφημα ήταν πόσο καλύτερες προβλέψεις κάνουν τα μοντέλα παλινδρόμησης μεγαλύτερης τάξης σε σχέση με το γραμμικό, αν και όσο αυξάνουμε την τάξη τόσο ποιό πιθανό είναι να έχουμε προβλήματα overfitting τα οποία ναι μεν θα έχουν πολύ καλή ακρίβεια πάνω στα δεδομένα εκπαίδευσης αλλά όχι τόσο καλή ακρίβεια πάνω στα δεδομένα δοκιμών.

Στον παρακάτω πίνακα παρατηρούμε πόσο μειώνεται το RMSE όσο ανεβάζουμε τον βαθμό της πολυωνυμικής παλινδρόμησης.

Βαθμός Παλινδρόμησης	RMSE
Γραμμικός	163388.7
Δευτέρου βαθμού	82212.1
Τρίτου βαθμού	38931.5
Τετάρτου βαθμού	14503.2

Πίνακας 1: RMSE (Root Mean Square error) για τους δίαφορους βαθμούς γραμμικής παλινδρόμησης

Χρησιμοποιόντας τα μοντέλα θα προβλέψουμε τον μισθό ενός υπαλλήλου επιπέδου 7. Όπως ήταν αναμενόμενο μετά και από τον παραπάνω πίνακα όσο

μεγαλύτερος είναι ο βαθμός τόσο καλύτερη πρόβλεψη παίρνουμε.

```
dataset[dataset$Level==7,"Salary"] # real value

## [1] 200000

predict(poly1_model, data.frame(Level=7)) #linear prediction

##        1
## 370818.2

predict(poly2_model, data.frame(Level=7)) #quadratic prediction

##        1
## 254227.3

predict(poly3_model, data.frame(Level=7)) #cubic prediction

##        1
## 177594.4

predict(poly4_model, data.frame(Level=7))

##        1
## 184003.5
```

Τρόπος Υπολογισμού	Μισθός
Πραγματική τιμή	200000
Γραμμική Παλινδρόμηση	370818
Πολυωνυμική Παλινδρόμηση 2ου βαθμού	254227
Πολυωνυμική Παλινδρόμηση 3ου βαθμού	177594
Πολυωνυμική Παλινδρόμηση 4ου βαθμού	184003

Πίνακας 2: Σύγκριση μισθών για έναν υπάλληλου επιπέδου 7

3 Αλγόριθμοι ταξινόμησης

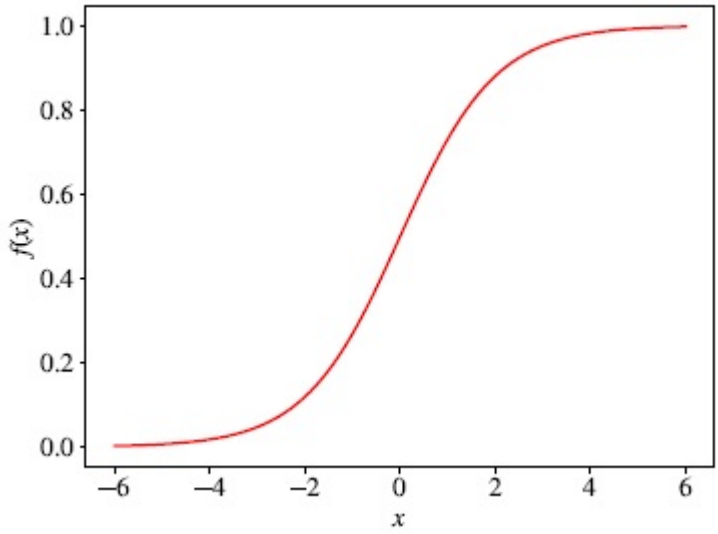
3.1 Λογιστική παλινδρόμηση

3.1.1 Περιγραφή Αλγορίθμου Λογιστικής Παλινδρόμησης

Το πρώτο πράγμα που πρέπει να ξεκαθαρίσουμε είναι ότι η λογιστική παλινδρόμηση (Logistic Regression) δεν είναι ένας αλγόριθμος παλινδρόμησης αλλά ένας αλγόριθμος ταξινόμησης. Το όνομα προέρχεται από την στατιστική και οφείλεται στο γεγονός ότι η μαθηματική διατύπωση της λογιστικής παλινδρόμησης είναι παρόμοια με εκείνη της γραμμικής παλινδρόμησης.

Στην λογιστική παλινδρόμηση εξακολουθούμε να θέλουμε να μοντελοποιήσουμε το y_i ως γραμμική συνάρτηση του x_i ωστόσο όταν το y_i είναι δυαδικό αυτό δεν είναι τόσο απλό. Ο γραμμικός συνδιασμός των χαρακτηριστικών όπως το $ax_i + b$ είναι μια συνάρτηση που εκτείνεται από το μείον άπειρο μέχρι το συν άπειρο ενώ το y_i έχει μόνο δύο πιθανές τιμές. Για να αποφύγουμε αυτό το πρόβλημα πρέπει να μοντελοποιήσουμε το y_i χρησιμοποιόντας μια συνάρτηση που παίρνει τιμές μεταξύ 0 και 1 για όλες τις τιμές του x_i όπου 0 θα είναι η μια ετικέτα του y και 1 η άλλη. Πολλές συναρτήσεις πληρούν αυτή την περιγραφή. Στην λογιστική παλινδρόμηση χρησιμοποιούμε τη λογιστική συνάρτηση (γνωστή και ως σιγμοειδή συνάρτηση)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Σχήμα 3.1: Σιγμοειδής συνάρτηση $\sigma(x) = \frac{1}{1+e^{-x}}$

3.1.2 Μαθηματική παρουσίαση της Λογιστικής Παλινδρόμησης

Κοιτόντας την γραφική παράσταση της λογιστικής συνάρτησης μπορούμε να δούμε πόσο καλά ταιρίαζει στην πρόθεση μας να μοντελοποιήσουμε την ταξινόμηση εάν βελτιστοποιήσουμε κατάλληλα τους συντελεστές του γραμμικου συνδιασμου α και β. Αρκεί να δουμε την λογιστική συνάρτηση ως μια συνάρτηση που μας δίνει την πιθανότητα για καποιό συγκεκριμένο x_i το y_i να είναι 1, δηλαδή $f(x) = Pr(Y = 1|X)$. Μπορούμε επίσης να χρησιμοποιήσουμε ένα όριο για παράδειγμα 0.5 για το οποίο αν για κάποιο x θα έχουμε $f(x) \geq 0.5$ μπορούμε να το ταξινομήσουμε σαν 1 ενώ αντίθετα αν $f(x) < 0.5$ μπορούμε να το ταξινομήσουμε σαν 0. Άρα το λογιστικό μοντέλο θα έχει την μορφή

$$f(x) = \frac{1}{1+e^{-(ax+b)}}$$

$$\frac{f(x)}{1-f(x)} = e^{ax+b}$$

$$log(\frac{f(x)}{1-f(x)}) = ax + b$$

Το ερώτημα που γεννάται είναι πως θα βρεθούν οι βέλτιστοι συντελεστές a και b. Στην γραμμική παλινδρόμηση προσπαθούσαμε να ελαχιστοποιήσουμε το μέσο τετραγωνικό σφάλμα.Από την άλλη στην λογιστική παλινδρόμηση προσπαθούμε να μεγιστοποιήσουμε την πιθανοφάνεια του συνόλου εκπαίδευσης σύμφωνα με το μοντέλο. Στην στατιστική η συνάρτηση πιθανοφάνειας καθορίζει πόσο πιθανή είναι η παρατήρηση(ένα παράδειγμα) σύμφωνα με το μοντέλο μας.

Για παράδειγμα, αν έχουμε επισημασμένα παραδείγματα (x_i, y_i) και έχουμε βρει κάποια a, b αν εφαρμόσουμε το λογιστικό μοντέλο για τα x_i θα πάρουμε μια τιμή $0 < p < 1$. Αν το y_i έχει θετική κλάση ($y_i = 1$) τότε η πιθανοφάνεια το y_i να έχει θετική κλάση θα δίνεται από το μοντέλο και θα είναι p. Ομοίως άμα το y_i έχει αρνητική κλάση ($y_i = 0$) τότε η πιθανοφάνεια το y_i να έχει αρνητική κλάση θα δίνεται από το μοντέλο και θα είναι $1 - p$.

Το κριτήριο βελτιστοποίησης στην λογιστική παλινδρόμηση ονομάζεται μέγιστη πιθανοφάνεια (maximum likelihood). Αντί να ελαχιστοποιήσουμε την μέγιστη απώλεια, όπως κάναμε στη γραμμική παλινδρόμηση, μεγιστοποιούμε την πιθανοφάνεια για τα δεδομένα εκπαίδευσης.

$$L_{a,b} = \prod_{i=1}^{N} f(x)^{y_i}(1 - f(x))^{(1-y_i)}$$

Επειδή έχουμε εκθετική συνάρτηση στο μοντέλο μας, είναι πιο βολικό να μεγιστοποιήσουμε την λογαριθμική πιθανοφάνεια

$$LogL_{a,b} = ln(L_{a,b}) = \sum_{i=1}^{N} y_i ln(f(x)) + (1 - y_i) ln(1 - f(x))$$

Μιας και η λογαριθμική είναι γνησίως αύξουσα συνάρτηση, μεγιστοποιώντας την συνάρτηση είναι το ίδιο σαν να μεγιστοποιούμε το ζητούμενο, και η λύση σε αυτό το πρόβλημα βελτιστοποίησης είναι η ίδια με την λύση του αρχικού προβλήματος.

Ένας άλλος τρόπος να γράψουμε τον παραπάνω τύπο είναι ο εξης. Υποθέτουμε ότι $y_i \in \{-1, 1\}$ αντί για $y_i \in \{0, 1\}$. Έχουμε $P(y = 1) = \frac{1}{1+e^{-(ax+b)}}$ και $P(y = -1) = \frac{1}{1+e^{(ax+b)}}$ τότε θα έχουμε:

$$LogL_{a,b} = -\sum_{i=1}^{N} log(1 + exp(-y_i(ax + b)))$$

Η συνάρτηση Cross-entropy δίνεται απο τον τύπο $H_{a,b} = -LogL_{a,b}$. Άρα μπορούμε αντί να ψάχνουμε το μέγιστο της συνάρτησης $LogL_{a,b}$ να βρούμε το ελάχιστο της συνάρτησης $H_{a,b}$.

Σε αντίθεση με την γραμμική παλινδρόμηση ο υπολογισμός της βέλτιστης λύσης δεν είναι τόσο εύκολος. Εδώ θα χρειαστεί μια τυπική διαδικασία αριθμητικής βελτιστοποίησης όπως ο αλγόριθμος απότομης καθόδου (gradient descent).

3.1.3 Υλοποίηση Λογιστικής Παλινδρόμησης στην R

Θα χρησιμοποιήσουμε την βάση δεδομένων Binary. Αυτή η βάση περιέχει 400 καταχωρίσεις όπου στην κάθε μία έχουμε τον μέσο όρο του μαθητή(GPA),

την βαθμολογία του στο GRE και την τάξη του ενώ η στήλη admit έχει να κάνει με το γεγονός αν έγινε δεκτος στο πανεπιστήμιο ή όχι.

Φορτώνουμε την βάση δεδομένων μας και μετατρέπουμε τα χαρακτηριστικά admit και rank απο αριθμητικά σε κατηγορικά. Στην στήλη admit με 0 θα έχουμε τους μαθητές που δεν γίναν δεκτοί στο πανεπιστήμιο και με 1 αυτούς που γίναν δεκτοί.

```
# Loading the dataset
binary <- read.csv("C:/Users/tit0v/R/PTUXIAKH/binary.csv")
# change the object type of admit and rank
binary$admit <- as.factor(binary$admit)
binary$rank <- as.factor(binary$rank)
```

Χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 80/20. Θέτουμε το set.seed με έναν τυχαίο αριθμό για να παίρνουμε πάντα τα ίδια αποτελέσματα και να παράξει ο αναγνώστης ακριβώς τα ίδια αποτελέσματα με αυτά που θα παρουσιάσουμε.

```
# Create indices (1 and 2) with 80/20 proportion to split the data
# ind==1 with probability 80% for train set and ind==2 with probability
# 20% is for test set, so we filter the data using this two indices.
set.seed(1234)
ind <- sample(2,nrow(binary),replace=T,prob=c(0.8,0.2))
train<- binary[ind==1,] #filtering the dataset to create the train set
test <- binary[ind==2,] #filtering the dataset to create the test set
```

Δημιουργούμε το μοντέλο μας εκπαιδεύοντας το πάνω στα δεδομένα εκπαίδευσης. Η φόρμουλα admit~. σημαίνει ότι θα προβλέψουμε την τιμή του admit χρησιμοποιόντας όλα τα υπόλοιπα χαρακτηριστικά (εξού και η τελεία)

```
# Creating the logistic regression model on train set
glm_model <- glm(admit~.,data = train, family='binomial')
```

Οι λεπτομέρειες του μοντέλου μας δίνονται παρακάτω

```
summary(glm_model) # Printing the details of our model

##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5873  -0.8679  -0.6181   1.1301   2.1178
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.009514   1.316514  -3.805 0.000142 ***
## gre          0.001631   0.001217   1.340 0.180180
## gpa          1.166408   0.388899   2.999 0.002706 **
## rank2       -0.570976   0.358273  -1.594 0.111005
## rank3       -1.125341   0.383372  -2.935 0.003331 **
## rank4       -1.532942   0.477377  -3.211 0.001322 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 404.39  on 324  degrees of freedom
## Residual deviance: 369.99  on 319  degrees of freedom
## AIC: 381.99
##
## Number of Fisher Scoring iterations: 4
```

Παρατηρούμε ότι το GPA όπως και τα rank2 και rank3 έχουν μεγαλύτερη βαρύτητα στην πρόβλεψη του μοντέλου μας πράγμα που φαίνεται από την ένδειξη των δύο αστερίσκων (**)

Χρησιμοποιούμε το μοντέλο που δημιουργήσαμε για να βρούμε την τιμή του admit στα δεδομένα δοκιμής.

```
# Calculating the probability
glm_pred <- predict(glm_model, newdata = test, type = 'response')
```

Επειδή όμως ο αλγόριθμος μας επιστρέφει πιθανότητα θα θεωρήσουμε ότι αν είναι πάνω από 0.5 θα γίνεται δεκτός.

```
# Setting the prediction with 1 if the probability is higher than 0.5
# or with 0 if is lower than 0.5
glm_prediction <- ifelse(glm_pred > 0.5,1,0)
```

Ο πίνακας σύγκρισης των πραγματικών δεδομένων και των δεδομένων πρόβλεψης θα είναι ο εξής

```
# Confusion matrix
cm_glm <- table(glm_prediction, test$admit)
cm_glm

##
## glm_prediction  0  1
##              0 48 21
##              1  2  4
```

Τέλος η ακρίβεια του μοντέλου μας θα είναι

```
accuracy_glm <- sum(diag(cm_glm)) / sum(cm_glm)
accuracy_glm # Model accuracy

## [1] 0.6933333
```

Ο αλγόριθμος λογιστικής παλινδρόμησης μας δίνει ακρίβεια 69%. Από τις 75 καταχωρίσεις ταξινόμησε σωστά τις 52 και λάθος τις 23. Από αυτές τις 23 οι 21 είναι για υποψηφίους που το μοντέλο πρόβλεψε ότι θα γίνουν δεχτοί στο πανεπιστήμιο, ενώ στην πραγματικότητα δεν γίναν. Τέλος μόνο 2 μαθητές εγίναν δεχτοί στο πανεπιστήμιο έχοντας προβλέψει το μοντέλο μας ότι δεν θα γίνουν.

3.2 Απλός ταξινομητής Bayes

3.2.1 Μαθηματική περιγραφή του ταξινομητή Bayes

Ο απλός ταξινομητής Bayes (Naive bayes Classifier) είναι ένας ταξινομητής βασισμένος στο θεώρημα Bayes με την απλή παραδοχή ότι τα χαρακτη-

ριστικά είναι ανεξάρτητα το ένα από το άλλο.

Θεώρημα Bayes

'Εστω ένα χαρακτηριστικό διάνυσμα $X = (x_{1,}x_2, ..., x_n)$ και μια κατηγορική μεταβλητή C_k , το θεώρημα Bayes μας λέει ότι:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, \quad \gamma\iota\alpha\, k = 1, 2, ..., K$$

με $P(C_k|X)$ να η εκ των υστέρων πιθανότητα, $P(X|C_k)$ η πιθανοφάνεια, $P(C_k)$ η εκ των προτέρων πιθανότητα της κλάσης ενώ $P(X)$ η εκ των προτέρων πιθανότητα της πρόβλεψης.

Χρησιμοποιόντας τον κανόνα αλυσίδας, η πιθανοφάνεια $P(X|C_k)$ μπορει να πάρει την μορφή

$$P(X|C_k) = P(x_1, ..., x_n|C_k) =$$

$$P(x_1|x_2, ..., x_n, C_k)P(x_2|x_3, ..., x_n, C_k)...P(x_{n-1}|x_n, C_k)P(x_n|C_k)$$

Σε αυτό το σημείο η "αφελής-naive" υποθετική προϋπόθεση ανεξαρτησίας αναλαμβάνει ρόλο. Συγκεκριμένα τα απλά μοντέλα Bayes υποθέτουν ότι το χαρακτηριστικό x_i είναι ανεξάρτητο από το χαρακτηριστικό x_j για $i \neq$j για δοσμένη κλάση.

$$P(x_i|x_{i+1}, ..., x_n|C_k) = P(x_i|C_k)$$

θα έχουμε

$$P(x_1, ..., x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

έτσι η πιθανότητα θα γίνει

$$P(C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(X)}$$

και μιας και το $P(X)$ είναι σταθερο δεδομένου των τιμών εισόδου θα έχουμε

$$P(C_k|X) \propto P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

όπου το σύμβολο $\propto$ σημαίνει ότι είναι θετικά αναλόγο προς το δεύτερο μέλος

Έτσι ένα πρόβλημα απλού ταξινομητής Bayes γίνεται ως εξής, για τις διάφορες κλάσεις του C_k βρες το μέγιστο του $P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$. Αυτό μπορεί να γραφτεί και ως:

$$\hat{C} = argmax_{C_k} P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

όπου $\hat{C}$ είναι η εκτιμώμενη κλάση για το δοσμένο διάνυσμα $\vec{x}$.

Πρακτικά μιλόντας, η πιθανοφάνεια $P(x_i|C_k)$ συνήθως διαμορφώνεται χρησιμοποιόντας την ίδια κατηγορία κατανομής πιθανοτήτων. Οι διάφοροι απλοί ταξινομητές Bayes διαφέρουν κυρίως ως προς τις υποθέσεις που κάνουμε σχετικά με την κατανομή του $P(x_i|C_k)$.

3.2.2 Ταξινομητές Bayes

Gaussian naive Bayes

Όταν πρόκειται για συνεχή δεδομένα, μια τυπική παραδοχή είναι ότι οι συνεχείς τιμές που σχετίζονται με τη κάθε κατηγορια κατανέμονται σύμφωνα με την κανονική(ή Γκαουσιαννη) κατανομή. Η συνάρτηση Gauss θα είναι:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

όπου μ είναι η μέση τιμή των παρατηρήσεων και σ η διακύμανση.

Έτσι για συγκεκριμένη κλάση C_k η κατανομή πιθανότητας του u, με μ_k, σ_kτην μέση την τιμη και την διακύμανση των τιμών που σχετίζονται με την κλάση C_k , θα είναι:

$$P(x = u|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(u-\mu_k)^2}{2\sigma_k^2}}$$

Multinomial naive Bayes

Με ένα πολυωνυμικό μοντέλο, τα δείγματα αντιπροσωπεύουν τις συχνότητες με τις οποίες έχουν δημιουργηθεί από ένα πολύ-πολυώνυμο $(p_1, .., p_n)$ όπου p_i είναι η πιθανότητα το i συμβάν να πραγματοποιηθεί. Ένα χαρακτηριστικό διάνυσμα $x = (x_1, .., x_n)$ τότε είναι ένα ιστόγραμμα, με το x_i να υπολογίζει τον αριθμό των συμβάντων i που παρατηρήθηκαν σε μια συγκεκριμένη περίπτωση. Η πιθανοφάνεια του να παρατηρηθεί ένα ιστόγραμμα x δίνεται από τον τύπο:

$$P(x|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

Ο απλός πολυωνυμικός ταξινομητής Bayes γίνεται γραμμικός ταξινομητής εκφρασμένος στον λογαριθμικό χώρο.

$$logP(C_k|x) \propto log(P(C_k \prod_{i=1}^{n} p_{ki}^{x_i})) = logP(C_k) + \sum_{i=1}^{n} x_i logP_{ki} = b + w_k^T x$$

με $b = logP(C_k)$ και $w_{ki} = logP_{ki}$

Bernoulli naive Bayes

Στο μοντέλο πολλαπλών μεταβλητών Bernoulli ,τα χαρακτηριστικά είναι ανεξάρτητες δυαδικές μεταβλητές που περιγράφουν τις τιμές εισόδου. Το μοντέλο αυτό είναι αρκετά δημοφιλές για εργασίες ταξινόμηση εγγράφων. Εάν το x_i είναι δυαδική έκφαση της απουσίας ή εμφάνισης του i όρου από το λεξικό, τότε η πιθανοφάνεια ενός εγγράφου δοσμένης κλάσης C_kθα δίνεται από τον τύπο:

$$P(X|C_k) = \prod_{i=1}^{n} P_{ki}^{x_i}(1 - P_{ki})^{(1-x_i)}$$

όπου P_{ki} είναι η πιθανότητα της κλάσης C_k να δημιουργήσει τον όρο x_i.

3.2.3 Υλοποίηση Naive Bayes στην R

Θα χρησιμοποιήσουμε την βάση δεδομένων binary που χρησιμοποιήσαμε και στην λογιστικη παλινδρόμηση.

Φορτώνουμε την βάση δεδομένων και την βιβλιοθήκη που έχει έτοιμο τον αλγόριθμο Naive Bayes που θα χρησιμοποιήσουμε. Πάλι χωρίζουμε τα δεδομένα σε αναλογία 80/20 και θέτουμε το set.seed για να μπορούμε να παράγουμε πάντα τα ίδια ακριβώς αποτελέσματα.

```
# Loading the library and the dataset
library(naivebayes)
binary <- read.csv("C:/Users/tit0v/R/PTUXIAKH/binary.csv")
binary$admit <- as.factor(binary$admit)
binary$rank <- as.factor(binary$rank)
set.seed(1234)
# Creating the train and test set from dataset
ind <- sample(2,nrow(binary),replace=T,prob=c(0.8,0.2))
```

```r
train<- binary[ind==1,] #filtering the dataset to create the train set
test <- binary[ind==2,] #filtering the dataset to create the test set
```

Δημιουργούμε το μοντέλο μας το οποίο εκπαιδεύεται στα δεδομένα εκπαίδευσης και χρησιμοποιεί όλα τα χαρακτηριστικά για να προβλέψει το αν θα γίνει δεκτός ο μαθητής ή οχι.

```r
# Model creation
nb_model <- naive_bayes(admit~.,data=train)
summary(nb_model) # Model's details
```

```
##
## ================================== Naive Bayes ==================================
##
## - Call: naive_bayes.formula(formula = admit ~ ., data = train)
## - Laplace: 0
## - Classes: 2
## - Samples: 325
## - Features: 3
## - Conditional distributions:
##     - Categorical: 1
##     - Gaussian: 2
## - Prior probabilities:
##     - 0: 0.6862
##     - 1: 0.3138
##
## ---------------------------------------------------------------------------
```

Χρησιμοποιούμε το μοντέλο που δημιουργήσαμε για να βρούμε την τιμή του admit στα δεδομένα δοκιμής και παράγουμε τον πίνακα σύγκρισης

```
# Predicting the class using the Naive Bayes model
nb_pred <- predict(nb_model, newdata = test, type='class')
cm_nb <- table(nb_pred,test$admit)
cm_nb # Confusion Matrix

##
## nb_pred  0  1
##       0 47 21
##       1  3  4
```

Το μοντέλο μας θα έχει ακρίβεια

```
accuracy_nb <- sum(diag(cm_nb)) / sum(cm_nb)
accuracy_nb # Printing the accuracy

## [1] 0.68
```

Η ακρίβεια του μοντέλου μας είναι 68%. Οι λάθος ταξινομήσεις είναι 24 από τις οποίες 21 είναι για μαθητές που προβλέφθηκαν να γίνουν δεκτοί και δεν έγινε και 3 για μαθητές που προβλέφθηκαν να μην γίνουν δεκτοί και τελικά μπήκαν στο πανεπιστήμιο. Παρατηρούμε ότι η απόδοση του Naive Bayes και της λογιστικής παλινδρόμησης είναι παρόμοια.

3.3 Μέθοδος Κ-Κοντινότερων Γειτόνων - kNN

3.3.1 Παρουσίαση του kNN αλγορίθμου

Η μέθοδος Κ-Κοντινότερων Γειτόνων (K-Nearest Neightbors or kNN) είναι ένας μη παραμετρικός αλγόριθμος μάθησης. Αντίθετα με άλλους αλγόριθμους μάθησης που επιτρέπουν την απόρριψη των δεδομένων εκπαίδευσης μετά την κατασκευή του μοντέλου, ο αλγόριθμος kNN κρατά τα παραδείγματα εκπαίδευσης στην μνήμη. Μόλις ένα νέο παραδειγμα x , εισέλθει στον αλγόριθμο, ο kNN αλγόριθμος ψάχνει τα k παραδείγματα εκπαίδευσης κοντινότερα στο x επιστρέφει την ετικέτα με τις περισσότερες εμφανίσεις σε περίπτωση προβλήματος ταξινόμησης, αλλίως την μέση τιμή σε περίπτωση προβλήματος παλινδρόμησης.

Ο Αλγόριθμος Κ-κοντινότερων γειτόνων είναι ένας τύπος μάθησης κατα περίπτωση (Instance-based Learning), όπου η συνάρτηση προσεγγίζεται μόνο τοπικά και όλοι οι υπολογισμοί αναβάλλονται μέχρι την ταξινόμηση.

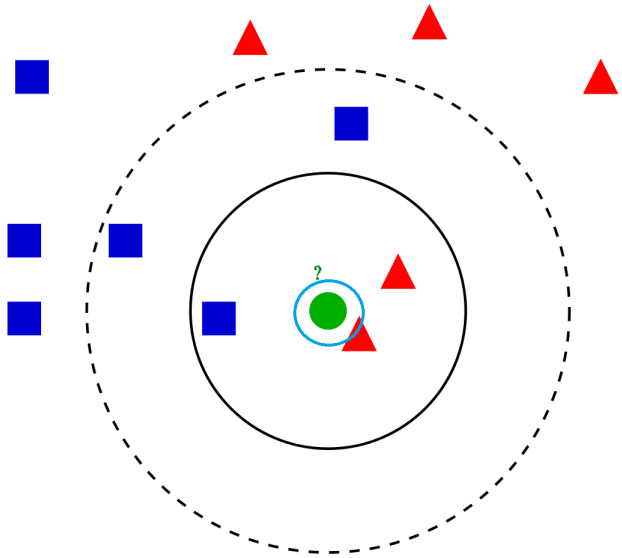
Τόσο για την ταξινόμηση όσο και για την παλινδρόμηση, μια χρήσιμη τεχνική μπορεί να είναι η χρήση συντελεστών βαρύτητας, έτσι ώστε όσο κοντινότεροι είναι οι γείτονες τόσο να συμβάλλουν περισσότερο στον μέσο όσο σε σχέση με του πιό μακρινούς. Για παράδειγμα, μια κοινή διαδικασία για τον υπολογισμό των συντελεστών βαρύτητας είναι να δώσουμε στον κάθε γείτονα τιμή βαρύτητας $1/d$ όπου d είναι η απόσταση από τον γείτονα.

Ο αλγόριθμος Κ-κοντινότερων γειτόνων κάνει την παραδοχή ότι τα διάφορα παραδείγματα μπορούν να αναπαρασταθούν ως σημεία σε κάποιον n-διάστατο χώρο $\mathbb{R}^n$ όπου n ο αριθμός των χαρακτηριστικών (ανεξάρτητων μεταβλητών). Κάθε νέα περίπτωση τοποθετείται στο χώρο αυτό σαν νέα τιμή που προσδιο-

ρίζεται με βάση το χαρακτηρισμό των k γειτονικών σημείων. Αν $k = 1$ τότε στην νέα μεταβλητή ανατίθεται η τάξη του πλησιέστερου γείτονα.

Για δοσμένο K και για μια νέα μεταβλητή x_0, ο kNN ταξινομητής βρίσκει τα K κοντινότερα σημεία στο $x_0 \in$ A, όπου A το σύνολο που περιέχει τις κοντινότερες K παρατηρησεις στο x_0. Έτσι η πιθανότητα το x_0 να ανήκει στην κλάση j θα δίνεται από τον παρακάτω τύπο.

$$P(Y = j|x = x_0) = \frac{1}{K} \sum_{i \in A} I(y_i = j)$$



Σχήμα 3.2: Παράδειγμα ταξινόμησης με k-NN [22]

με $I(x)$ η συνάρτηση η οποία μετράει πόσες φορές έχει εμφανιστεί η κλάση j στο σύνολο A.

Στο παράδειγμα της παραπάνω εικόνας (σχήμα 3.2) παρατηρούμε ότι η νέα μεταβλητή (Πράσινη βούλα) για $k = 1$ (μπλε κύκλος) θα ταξινομηθεί ως

κόκκινο τρίγωνο, για $k = 3$ (μαύρος κύκλος) θα ταξινομηθεί πάλι ως κόκκινο τρίγωνο ενώ για $k = 5$ (διακεκομμένος κύκλος) θα ταξινομηθεί ως μπλε τετράγωνο. Από αυτό το παράδειγμα είναι φανερό πως η επιλογή του k είναι σημαντική και μπορεί να επηρεάσει κατά μεγάλο βαθμό την ποιότητα των προβλέψεων. Αν το k είναι μικρό μπορεί να οδηγήσει σε πολύ μεγάλη διακύμανση ενώ αν το k είναι πολύ μεγαλό μπορεί να οδηγήσει σε μεγάλη μεροληψία. Η επιλογή του k όταν δεν μπορεί να γίνει εμπειρικά, γίνεται συνήθως με τον αλγόριθμο του Cross-Validation.

Μια σημαντική παράμετρος του αλγόριθμου είναι ο τρόπος με τον οποίο υπολογίζουμε την απόσταση ανάμεσα στην νέα παρατήρηση και στους κοντινότερους γείτονες. Οι διάφορες φόρμουλες υπολογισμού αποστάσεων βρίσκονται στην υποενότητα 1.5.2.

3.3.2 Υλοποίηση K-Κοντινότερων Γειτόνων στην R

Θα χρησιμοποιήσουμε την βάση δεδομένων binary που χρησιμοποιήσαμε και στους δύο προηγούμενους αλγορίθμους.

Φορτώνουμε την βάση δεδομένων και την βιβλιοθήκη που έχει έτοιμο τον αλγόριθμο KNN που θα χρησιμοποιήσουμε. Πάλι χωρίζουμε τα δεδομένα σε αναλογία 80/20 και θέτουμε το set.seed για να μπορούμε να παράγουμε πάντα τα ίδια ακριβώς αποτελέσματα.

```
# Loading the library and dataset
library(class)
binary <- read.csv("C:/Users/tit0v/R/PTUXIAKH/binary.csv")
binary$admit <- as.factor(binary$admit)
```

```
set.seed(1234)
# Creating the train and test set from dataset
ind <- sample(2,nrow(binary),replace=T,prob=c(0.8,0.2))
train<- binary[ind==1,] #filtering the dataset to create the train set
test <- binary[ind==2,] #filtering the dataset to create the test set
```

Αυτό που θα κάνουμε στο συγκεκριμένο αλγόριθμο και θα μας βοηθήσει πολύ στο να βελτιώσουμε την ακρίβεια είναι να κανονικοποιήσουμε τα χαρακτηριστικά μας. Η κανονικοποίηση γίνεται με τον τύπο $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$. Επειδή ο αλγόριθμος μας στηρίζεται στις αποστάσεις και στο παράδειγμα μας συγκεκριμένα στην ευκλείδια απόσταση για να μην υπερισχύει το ένα χαρακτηριστικό από τα άλλα μετατρέπουμε όλες τις τιμές στο διάστημα από 0 έως 1. Στο τέλος κατηγοροποιούμε την στήλη rank μιας και παρόλο που η τιμή του είναι αριθμητική τα νούμερα που παίρνει είναι συγκεκριμένα.

```
# Data normalization
train[,-1] <-apply(train[,-1], 2, function(x) (x-min(x))/(max(x)-min(x)))
test[,-1] <-apply(test[,-1], 2, function(x) (x-min(x))/(max(x)-min(x)))
# Change the object type to factor
train$rank<- as.factor(train$rank)
test$rank <- as.factor(test$rank)
```

Επειδή ο αλγόριθμος κ-Κοντινότερων γειτόνων είναι ένας Instance-based Learning αλγόριθμος μπορούμε με την μια να προβλέψουμε την κλάση χωρίς να χρειαστεί να δημιουργήσουμε μοντέλο.

```
# Training the K-NN model choosing k=5
knn_pred <- knn(train[,-1],test[,-1], cl=train[,1], k=5)
```

Ο πίνακας σύγκρισης ανάμεσα στις πραγματικές τιμές και τις τιμές που προβλέψαμε θα είναι:

```
knn_cm <- table(knn_pred,test$admit)
knn_cm # Confusion matrix

##
## knn_pred  0  1
##        0 45 15
##        1  5 10
```

Τέλος η ακρίβεια του μοντέλου μας θα είναι:

```
accuracy_knn <- sum(diag(knn_cm)) / sum(knn_cm)
accuracy_knn # kNN model accuracy

## [1] 0.7333333
```

Το μοντέλο των κ-Κοντινότερων γειτόνων για $k = 5$ μας δίνει ακρίβεια 73% και 20 λάθος ταξινομήσεις. Από αυτές 15 λάθος προβλέψεις για είσοδο στο πανεπιστήμιο και 5 λάθος προβλέψεις για απόρριψη από το πανεπιστήμιο.

3.4 K-Fold Cross-Validation

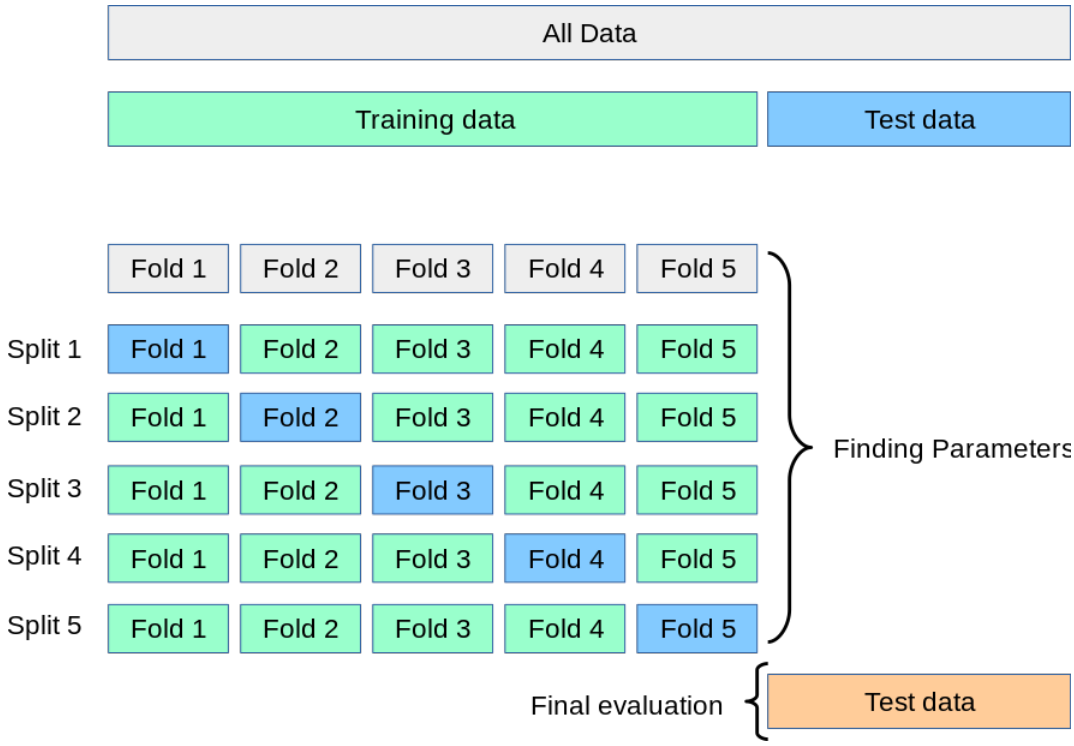
Στην ιδανική περίπτωση, αν είχαμε αρκετά δεδομένα, θα αφήναμε στην άκρη ένα σύνολο επικύρωσης (validation set) και θα το χρησιμοποιούσαμε για να αξιολογήσουμε την απόδοση του μοντέλου μας. Δεδομένου ότι τα δεδομένα είναι συχνά σπάνια και δύσκολα μπορείς να τα αποκτήσεις, αυτό συνήθως δεν είναι δυνατό. Για να αντιμετωπίσουμε αυτό το πρόβλημα η τεχνική K-fold cross validation χρησιμοποιεί μέρος των διαθέσιμων δεδομένων για να εκπαιδεύσει το μοντέλο, και ένα διαφορετικό μέρος για να το δοκιμάσει. Ουσιαστικά διαχωρίζουμε τα δεδομένα μας σε K ίσα μέρη και από αυτά χρησιμοποιούμε το ένα σαν δεδομένα δοκιμών και τα υπόλοιπα K-1 σαν δεδομένα εκπαίδευσης. Αυτή την διαδικασία την επαναλαμβάνουμε, παίρνοντας για δεδομένα δοκιμής σε κάθε επανάληψη άλλο κ-οστό μέρος μέχρι να τα έχουμε χρησιμοποιήσει όλα. Αν το πρόβλημα μας είναι πρόβλημα ταξινόμησης και θέλουμε να μετρήσουμε το Μέσο τετραγωνικό σφάλμα δεν έχουμε παρά να βρούμε την μέση τιμή των σφαλμάτων για κάθε fold. Ακριβώς το ίδιο κάνουμε αν ψάχνουμε την ακρίβεια σε ένα πρόβλημα ταξινόμησης.

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

$$Accuracy = \frac{1}{k} \sum_{i=1}^{k} ACC_i$$

Συνήθως εφαρμόζουμε την τεχνική k-fold CV στα δεδομένα εκπαίδευσης έτσι ώστε να μπορέσουμε να κάνουμε ένα είδος επικύρωσης της απόδοσης

του μοντέλου μας πριν το εφαρμόσουμε στα δεδομένα δοκιμών. Αρκετές φορές η τεχνική χρησιμοποιείται για να μπορέσουμε να ρυθμίσουμε (tuning) τις υπερπαραμέτρους μας αφού μπορούμε σε κάθε ένα διαφορετικό μέρος (fold) να δοκιμάζουμε διαφορετικές τιμές για τις υπερπαραμέτρους, μέχρι να βρούμε αυτές που μας δίνουν τα καλύτερα δυνατά αποτελέσματα.



Σχήμα 3.3: k-fold Cross Validation γράφημα για k=5 [29]

3.5 Σύνοψη Κεφαλαίου - Συγκριτικό Αλγορίθμων

Σε αυτό το κεφάλαιο είδαμε τους αλγόριθμους ταξινόμησης και συγκεκριμένα τους αλγόριθμους της λογιστικής παλινδρόμησης, τον απλό ταξινομητή

Bayes και την μέθοδο K-Κοντινότερων γειτόνων. Και οι τρεις αλγόριθμοι εφαρμόστηκαν στην ίδια βάση δεδομένων. Τέλος εξετάσαμε την τεχνική του k-fold Cross Validation, ένα εργαλείο που όχι μονο μας βοηθάει σημαντικά να αξιολογήσουμε το μοντέλο μας αλλά και να το βελτιώσουμε .Στον παρακάτω πίνακα θα δούμε τις διαφορές των μοντέλων σε σχέση με την ακρίβεια, την ευαισθησία (Sensitivity) την ειδικότητα (Specificity) και το 95% διάστημα εμπιστοσύνης, όπου ευαισθησία και ειδικότητα δίνονται από τους παρακάτω τύπους

	Diseased (TD)	**Healthy (TH)**	
Test Positive	True positive (T_P)	False Positive (F_P)	$\text{PPV} = \frac{(TP)}{(TP)+(FP)}$
Test Negative	False Negative (F_N)	True negative (T_N)	$\text{NPV} = \frac{(TN)}{(FN)+(TN)}$
	$\text{Sensitivity} = \frac{(TP)}{(TP)+(FN)}$	$\text{Specificity} = \frac{(TN)}{(FP)+(TN)}$	

Σχήμα 3.4: Πίνακας υπολογισμού ευαισθησίας, ειδικότητας [28]

Μοντέλο	Ακρίβεια	Ευαισθησία	Ειδικότητα	95% ΔΕ
Λογιστική Παλινδρόμηση	0.69	0.96	0.16	(0.5762, 0.7947)
Απλός Ταξινομητής Bayes	0.68	0.94	0.16	(0.5622, 0.7831)
K-Κοντινότερων γειτόνων	0.73	0.90	0.40	(0.6186, 0.8289)

Πίνακας 3: Σύγκριση μοντέλων ταξινόμησης

Παρατηρούμε για την συγκεκριμένη βάση δεδομένων ότι ο αλγόριθμος kNN μας δίνει καλύτερα αποτελέσματα. Έχει μεγαλύτερη ακρίβεια σε σχέση

με τα άλλα δύο μοντέλα και παρόλο που η ευαισθησία του είναι ελαφρώς χαμηλότερη έχει πολύ καλύτερη ειδικότητα, πράγμα αρκετά σημαντικό όσο αφορά την απόδοση ενός μοντέλου.

Τα αποτελέσματα του παραπάνω πίνακα όμως περιέχουν τα στατιστικά των μοντέλων μετά από ένα πείραμα τα οποία τα πήραμε χρησιμοποιώντας συγκεκριμένο διαχωρισμό της βάσης δεδομένων για τα δεδομένα εκπαίδευσης και δεδομένα δοκιμών. Αυτός ο διαχωρισμός έγινε τυχαία άρα αυτή η τυχαιότητα επηρεάζει και τα αποτελέσματα των μετρήσεων μας.

Για να βγάλουμε ποιό ασφαλή συμπεράσματα για την απόδοση των μοντέλων, τα μοντέλα θα πρέπει να εκπαιδευτούν σε διαφορετικούς διαχωρισμούς της βάσης δεδομένων. Έτσι παρακάτω θα τρέξουμε τα μοντέλα μας 100 φορές όπου κάθε φορά θα αλλάζουν τα δεδομένα εκπαίδευσης και δοκιμών για να μπορέσουμε να αξιολογήσουμε καλύτερα το μοντέλο μας.

```
# Loading the library
library(caret)
set.seed(1928)
# Creating the 5-fold Cross Validation and repeat it 20 times
cvfolds1 <- createMultiFolds(binary$admit,k=5,times=20)
tcontrol1 <- trainControl(method = "repeatedcv", number = 5,
repeats = 20, index=cvfolds1)
# Logistic Regression model
glm2 <- train(admit ~ ., data = binary, method = "glm",
family = binomial,trControl = tcontrol1)
# Naive Bayes model
```

```
nb2 <- train(admit ~ ., data = binary, method = "nb",
trControl = tcontrol1,tuneGrid = data.frame(fL=0,
usekernel=T, adjust = 1))
# k-Nearest Neighbor
knn2 <- train(admit ~ ., data = binary, method = "knn",
preProcess = c("center", "scale"),trControl = tcontrol1)
```

```
t.test(glm2$resample$Accuracy)$estimate # GLM mean accuracy

## mean of x
## 0.7014751

t.test(glm2$resample$Accuracy)$conf.int # GLM 95% CI

## [1] 0.6941499 0.7088004
## attr(,"conf.level")
## [1] 0.95

t.test(nb2$resample$Accuracy)$estimate # Naive Bayes mean accuracy

## mean of x
## 0.7075005

t.test(nb2$resample$Accuracy)$conf.int # Naive Bayes 95% CI

## [1] 0.7015523 0.7134487
## attr(,"conf.level")
## [1] 0.95

t.test(knn2$resample$Accuracy)$estimate # kNN mean accuracy
```

```
## mean of x
## 0.6897635

t.test(knn2$resample$Accuracy)$conf.int # kNN 95% CI

## [1] 0.6807115 0.6988156
## attr(,"conf.level")
## [1] 0.95
```

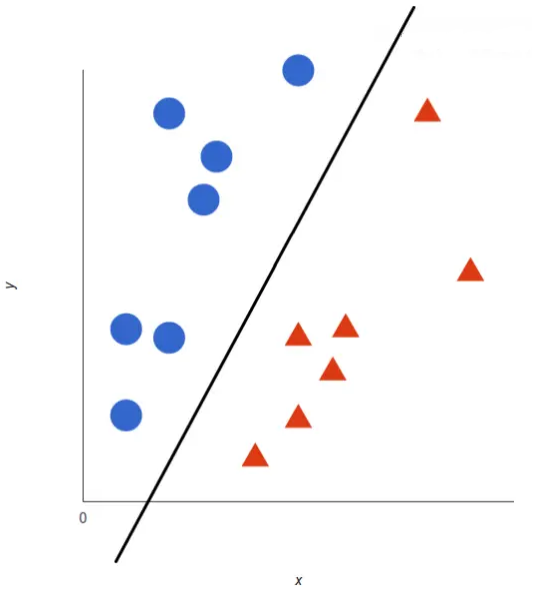
Συγκεντρώνοντας όλα τα στοιχεία στο παρακάτω πίνακα και κάνοντας σύγκριση με τον πίνακα 2 βλέπουμε πόσο διαφέρουν τα αποτελέσματα. Ο αλγόριθμος kNN από εκεί που ήταν ο αποτελεσματικότερος έγινε ο λιγότερο αποτελεσματικός. Αυτό το λάθος έγινε γιατί ”έτυχε” στο συγκεκριμένο πείραμα να έχει μεγαλύτερη ακρίβεια ενώ στα 100 πειράματα έχει κατά μέσο όρο 5% χαμηλότερη ακρίβεια, επίσης η πρόβλεψη είναι τόσο λάθος που η ακρίβεια του αρχικού πειράματος δεν ανήκει καν στο 95% διάστημα εμπιστοσύνης των 100 πειραμάτων. Από την άλλη τα άλλα δύο μοντέλα έχουν καλύτερη απόδοση κατά μέσο όρο σε σχέση με το αρχικό πείραμα με τον απλό ταξινομητή Bayes να μας δίνει ελάχιστα καλύτερα αποτελέσματα.

Μοντέλο	Mean	min	max	sd	95% CI
Λογιστική Παλινδρόμηση	0.7014	0.5926	0.7722	0.033	0.6941-0.7088
Απλός Ταξινομητής Bayes	0.7075	0.5875	0.7468	0.0348	0.7015-0.7134
K-Κοντινότερων γειτόνων	0.6897	0.6	0.775	0.034	0.6807-0.6988

Πίνακας 4: Πίνακας στατιστικών ακρίβειας μετά απο 100 επαναλήψεις

4 Μηχανές Διανυσμάτων Υποστήριξης

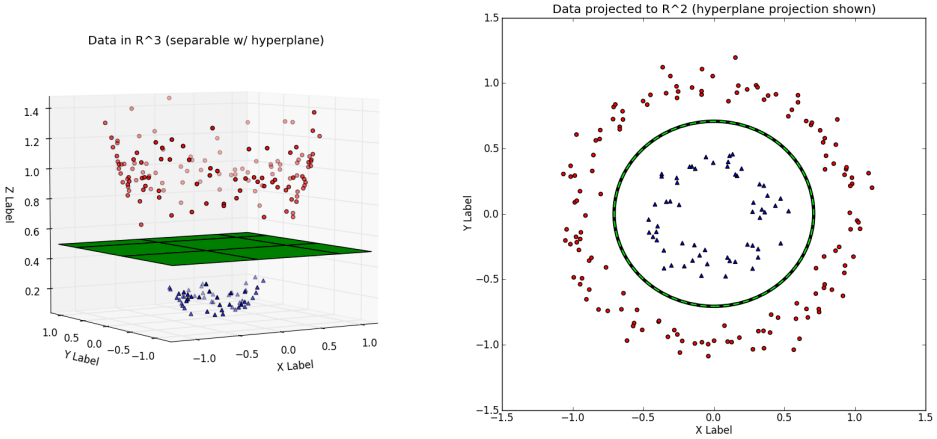
Στην μηχανική μάθηση, οι μηχανές διανυσμάτων υποστήριξης (Support Vector Machines - SVM), είναι μοντέλα επιβλεπόμενης μάθησης τα οποία μπορούν να χρησιμοποιηθούν σε προβλήματα παλινδρόμησης και ταξινόμησης. Ωστόσο χρησιμοποιούνται κυρίως σε προβλήματα ταξινόμησης. Σε αυτόν τον αλγόριθμο σχεδιάζουμε κάθε τιμή εισόδου ως ένα σημείο στον n-διάστατο χώρο (με n ο αριθμός των χαρακτηριστικών που έχουμε) με την τιμή κάθε χαρακτηριστικού να είναι η τιμή μιας συγκεκριμένης συντεταγμένης. Στη συνέχεια, πραγματοποιούμε ταξινόμηση βρίσκοντας το υπέρεπίπεδο (hyperplane) διάστασης $n - 1$ που διαφοροποιεί τις δύο κατηγορίες.



Σχήμα 4.1: Γραμμικά διαχωρίσιμο πρόβλημα

Η πίο απλή περίπτωση εμφανίζεται όταν έχουμε ένα γραμμικά διαχωρίσιμο πρόβλημα όπου τα δείγματα των δύο κλάσεων μπορούν να διαχωριστούν από μία ευθεία στις δύο διαστάσεις ή ένα υπερεπίπεδο σε περισσότερες διαστά-

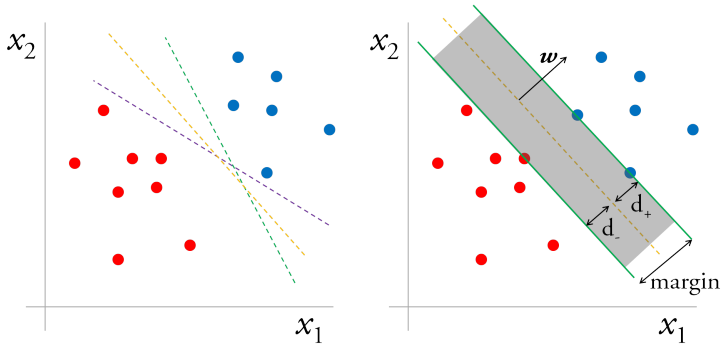
σεις χωρίς κανένα δείγμα να ταξινομηθεί λάθος



Σχήμα 4.2: Μη Γραμμικά διαχωρίσιμο πρόβλημα και αύξηση διάστασης [24]

Τι γίνεται όμως στην περίπτωση που τα δείγματα των δύο κλάσεων μπορούν εύκολα να διαχωριστούν αλλά όχι γραμμικά; Αυτό που χρειάζεται να κάνουμε είναι να προσθέσουμε μια ακόμα διάσταση και να βρούμε ένα υπερεπίπεδο διάστασης αυξημένης κατά ενα.

4.1 Ταξινομητής Διανυσμάτων Υποστήριξης



Σχήμα 4.3: (α)Παράδειγμα γραμμικού διαχωρίσιμου προβλήματος (β)Παράδειγμα ευθείας μέγιστου περιθωρίου [25]

Για να δούμε καλύτερα πως λειτουργεί το μοντέλο της μηχανής διανυσμάτων υποστήριξης θα εξετάσουμε τον τρόπο λειτουργίας των γραμμικά διαχωρίσιμων προβλημάτων. Έστω ότι έχουμε ένα πρόβλημα ταξινόμησης δύο κλάσεων όπου τα δείγματα της μίας κλάσης είναι πλήρως διαχωρίσιμα από τα δείγματα της άλλης κλάσης. Στο παρακάτω σχήμα παρατηρούμε ότι υπάρχει ένα άπειρο σύνολο ευθειών που διαχωρίζουν τα δείγματα των δύο κλάσεων χώρις κανένα δείγμα να ταξινομείται λάθος.

Σκόπος του αλγόριθμου είναι να βρούμε την καλύτερη δυνατή ευθεία η οποία θα μας δώσει τον μεγαλύτερο διαχωρισμό ή περιθώριο (margin) μεταξύ των δύο τάξεων. Έτσι επιλέγουμε την ευθεία για την οποία μεγιστοποιείται η απόσταση από το πλησιέστερο σημείο δεδομένων για κάθε κλάση. Εάν αυτή η ευθεία υπάρχει, είναι γνωστή ως υπερεπίπεδο μέγιστου περιθωρίου.

Έστω ένα πρόβλημα ταξινόμησης δύο κλάσεων C_1και C_2, οι οποίες είναι γραμμικά διαχωρίσιμες. Έστω επίσης ό,τι δίνεται ένα σύνολο δειγμάτων εκπαίδευσης x_i, όπου $x_i \in \mathbb{R}^d$, και η ετικέτα(label) κάθε δείγματος y_i για την οποία ισχύει $y_i \in \{-1, 1\}$. Η εξίσωση του υπερεπιπέδου που διαχωρίζει τα δείγματα των δύο κλάσεων είναι

$$w \cdot x + b = 0$$

Για παράδειγμα, αν τα δείγματα βρίσκονται στις δύο διαστάσεις, δηλαδή $x_i \in \mathbb{R}^2$, τότε η εξίσωση του υπερεπιπέδου (ευθεία) είναι

$$w_1 x_1 + w_2 x_2 + b = 0$$

ενώ αν τα δείγματα βρίσκονται στις n διαστάσεις, τότε η εξίσωση που περιγράφει το υπερεπίπεδο είναι η

$$w_1x_1 + w_2x_2 + ... + w_nx_n + b = 0$$

Η προβλεπόμενη ετικέτα για κάποιο διάνυσμα εισόδου x θα δίνεται έτσι

$$f(x) = sign(w \cdot x + b)$$

όπου το $sign$ είναι μια μαθηματική συνάρτηση που παίρνει οποιαδήποτε τιμή και μας επιστρέφει $+1$ αν το αποτέλεσμα είναι θετικός αριθμός και -1 άμα είναι αρνητικός.

Επομένως οι δύο ευθείες που χωρίζουν τα δείγματα είναι οι:

$$\varepsilon_1 : \ w \cdot x + b = -1$$

και

$$\varepsilon_2 : \ w \cdot x + b = +1$$

όπου θα ισχύει

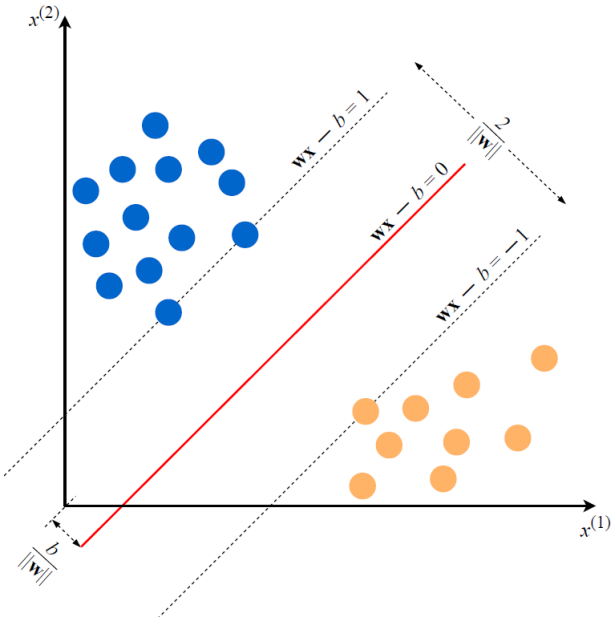
$$w \cdot x_i + b \leq -1 \quad \alpha\nu \quad x_i \in C_1$$

ή

$$w \cdot x_i + b \geq +1 \quad \alpha\nu \quad x_i \in C_2$$

Αν πολλαπλασιάσουμε τις παραπάνω ανισότητες με $y_i = 1$ για $x_i \in C_2$ ή με $y_i = -1$ για $x_i \in C_1$ θα πάρουμε την κοινή ανισότητα

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall x_i$$



Σχήμα 4.4: Παράδειγμα μοντέλου SVM για χαρακτηριστικά δυσδιάστατα διανύσμα [20]

Στόχος του αλγόριθμου είναι να μεγιστοποιήσουμε το περιθώριο $M = \frac{2}{\|w\|}$, όπου M η απόσταση των ευθειών ε_1και ε_2 και $\| w \|$ η ευκλείδια νόρμα. Ετσί αντί να μεγιστοποιήσουμε το περιθώριο μπορούμε να ελαχιστοποιήσουμε την νόρμα $\| w \|$. Οπότε καταλήγουμε στο εξής πρόβλημα βελτιστοποίησης:

$$\min_{w,b} \frac{1}{2} \| w \|^2$$

υπό τους περιορισμούς

$$y_i(w \cdot x_i + b) \geq 1 \quad i = 1, 2, ..., n$$

Μπορούμε να πάρουμε το παραπάνω πρόβλημα στην δυϊκή του μορφή και να επιλύσουμε το ίδιο πρόβλημα αναζητώντας τις βέλτιστες τιμές για τους πολλαπλασιαστές Lagrange

$$L_P(w, b, a) = \frac{1}{2} \| w \|^2 - \sum_{i=1} a_i[y_i(w^T x_i + b) - 1]$$

όπου $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)^T$. Το αρνητικό πρόσημο μπροστά απο τον πολλαπλασιαστή Lagrange το έχουμε επειδή προσπαθούμε να ελαχιστοποιήσουμε το L ως προς τα w και b, και να μεγιστοποιήσουμε σε σχέση με τα a_i. Έτσι θα έχουμε:

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow w^* = \sum_{i=1}^{n} a_i y_i x_i$$

όπου w^* η βέλτιστη τιμή του w και

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow 0 = \sum_{i=1}^{n} a_i y_i$$

Αντικαθιστόντας στην λανγκρασιανη θα πάρουμε

$$L_D = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} a_i a_j y_i y_j x_i^T x_j$$

με $\alpha_i \geq 0$ για κάθε i, και $\sum_{i=1}^{n} a_i y_i = 0$

Ακόμα θέλουμε να βρούμε και την βέλτιστη τιμή του b . Γνωρίζουμε ότι για ένα διάνυσμα υποστήριξης ισχύει $y_i(w^T x_i + b) = 1$, όπου αν για w χρησιμοποιήσουμε το w^* και λύσουμε ως προς b θα έχουμε $b^* = \frac{1}{y_i} - w^{*T} x_i$. Για λόγους αριθμητικής ευστάθειας καλό είναι να υπολογίσουμε τον μέσο όρο για όλα τα διανύσματα υποστήριξης, δηλαδή

$$b^* = \frac{1}{N_{sv}} \sum_{i \in N_{sv}} (y_i - w^{*T} x_i) = \frac{1}{N_{sv}} \sum_{i \in N_{sv}} (y_i - a_j y_j x_j^T x_i)$$

Τέλος άμα θέλουμε να κάνουμε πρόβλεψη για κάποιο νέο σημείο z, αρκεί να υπολογίσουμε το $sign$ του $y(z)$ με

$$y(z) = w^{*T} z + b^* = (\sum_{i=1}^{n} a_i y_i x_i)^T + b^*$$

4.2 Γραμμικά διαχωρίσιμα προβλήματα με ποσοστό λάθους

Μέχρι τώρα, έχουμε υποθέσει ό,τι τα σημεία των δεδομένων εκπαίδευσης είναι γραμμικά διαχωρίσιμα. Η προκύπτουσα μηχανή διανυσμάτων υποστήριξης θα μας δώσει τον ακριβή χωρισμό των δεδομένων εκπαίδευσης, ακόμα και αν αυτά δεν ειναι γραμμικά διαχωρίσιμα. Στην πράξη, ώστοσο, στη κατανομή ανά κλάση μπορεί να υπάρχει επικάλυψη των σημείων, οπότε ο ακριβής διαχωρισμός των δεδομένων μπορεί να οδηγήσει σε γενικεύσεις.

Χρειαζόμαστε λοιπόν να τροποποιήσουμε την μηχανή διανυσμάτων υποστήριξης έτσι ώστε να επιτρέπει ορισμένα σημεία των δεδομένων εκπαίδευσης να ταξινομούνται εσφαλμένα. Έτσι θα εισάγουμε την μεταβλητή $\xi_i \geq 0$ με $i = 1, 2, ..., n$, μία μεταβλητή χαλαρότητας για κάθε σημείο. Θεωρούμε ότι $\xi_i = 0$ για τα σημεία τα οποία βρίσκονται στη σωστή μερία του περιθωρίου και $\xi_i = |y_i - y(x_i)|$ για τα υπόλοιπα σημεία. Για να υπάρχει λάθος ταξινόμηση θα πρέπει $\xi_i > 1$. Έτσι μπορούμε να πούμε ότι η ποσότητα $\sum_{i=1}^{n} \xi_i$ αποτελεί το άνω όριο των δειγμάτων εκπαίδευσης που μπορούν να ταξινομηθούν εσφαλμένα. Έτσι θα έχουμε $\sum_{i=1}^{n} \xi_i \leq C$ όπου το C θα είναι μια υπερπαράμετρος που θα πρέπει να οριστεί από τον χρήστη αναλόγως πόση χαλαρότητα επιτρέπουμε στο εκάστοτε πρόβλημα. Για $C = 0$ αγνοούμε εντελώς τις μεταβλητές χαλαρότητας, ενώ όσο μεγαλώνουμε το C τόσο μεγαλύτερη βαρύτητα δίνουμε στην σωστή ταξινόμηση των δειγμάτων εκπαίδευσης.

Άρα θα έχουμε

$$w^T x_i + b \geq 1 - \xi_\iota, \quad \gamma\iota\alpha\ y_i = +1$$

$$w^T x_i + b \leq -1 - \xi_\iota, \quad \gamma\iota\alpha\ y_i = -1$$

τα οποία γράφονται σε μία σχέση

$$y_i(w^T x_i + b) \geq 1 - \xi_\iota, \quad \kappa\alpha\iota\ \xi_i \geq 0 \quad \gamma\iota\alpha\ i = 1, 2, ..., n$$

ενώ πλέον το πρόβλημα βελτιστοποίησης θα είναι το εξης

$$\underset{w,b,\xi}{min} \tfrac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{n} \xi_i$$

με περιορισμούς $y_i(w^T x_i + b) \geq 1 - \xi_\iota$, $\xi_i \geq 0$, $0 \leq \sum_{i=1}^{n} \xi_i \leq C$ για $i = 1, 2, ..., n$

Η αντίστοιχη συνάρτηση Lagrange (primal) θα είναι η

$$L_P(w, b, a) = \tfrac{1}{2} \parallel w \parallel^2$$
$$+C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} a_i[y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i$$

την οποία ελαχιστοποιούμε ως προς τα w, b, ξ_i μηδενίζοντας τις αντίστοιχες παραγώγους και θα πάρουμε

$$w = \sum_{i=1}^{n} a_i y_i x_i$$

$$0 = \sum_{i=1}^{n} a_i y_i$$

$$a_i = C - \mu_\iota$$

με a_i, μ_i,$\xi_i \geq 0 \quad \forall i$ όπου μ_i σχετίζεται με τους περιορισμούς του προβλήματος βελτιστοποίησης.

Η δυική συνάρτηση Lagrange (dual) που θα προκύψει βάζοντας στην L_p τις παραπάνω λύσεις θα μας δώσει

$$L_D(a) = \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} a_i a_j y_i y_j x_i^T x_j$$

Θα μεγιστοποιήσουμε την L_D ενώ θα πρέπει να ισχύει $0 \leq a_i \leq C$ και $\sum_{i=1}^{n} a_i y_i = 0$. Επιπλέον οι Karush-Kuhn-Tucker συνθήκες περιλαμβάνουν τους περιορισμούς

$$a_i[y_i(w^T x_i + b) - 1 + \xi_i] = 0,$$

$$\mu_i \xi_i = 0$$

$$y_i(w^T x_i + b) - 1 + \xi_i \geq 0$$

4.3 Μηχανές Διανυσμάτων Υποστήριξης

Μέχρι τώρα έχουμε δει ότι ο ταξινομητής διανυσμάτων υποστήριξης ανταποκρίνεται καλά σε προβλήματα που είναι γραμμικά διαχωρίσιμα ή γραμμικά διαχωρίσιμα με κάποιο ποσοστό λάθους. Το ερώτημα που προκύπτει είναι πως θα μπορούσαμε να διαχωρίσουμε μια βάση δεδομένων η οποία δεν είναι γραμμικά διαχωρίσιμη. Η ιδέα για να ξεπεράσουμε αυτό το πρόβλημα είναι να μεταφέρουμε τα δείγματα μέσω ενός μετασχηματισμού σε ένα χώρο υψηλότερων διαστάσεων και εκεί εφόσον πλέον τα δείγματα είναι γραμμικά διαχωρίσιμα βρίσκουμε το βέλτιστο γραμμικό υπερεπίπεδο.

Έστω $x_i \in \mathbb{R}^p$ τα δεδομένα εκπαίδευσης και Φ ο μετασχηματισμός με $\Phi : \mathbb{R}^p \mapsto H$, με Η να είναι χώρος Hilbert που σημαίνει πως είναι ένας διανυσματικός χώρος στον οποίο ορίζεται η πράξη του εσωτερικού γινομένου διανυσμάτων. Έτσι τα νέα στοχεία του χώρου είναι της μορφής $\Phi(x) = (\Phi_1(x), \Phi_2(x), ..., \Phi_n(x))$. Αφού πραγματοποιηθεί ο μετασχηματισμός των δεδομένων εκπαίδευσης, ο αλγόριθμος εκπαίδευσης θα βασίζεται πλέον στα νέα διανύσματα μέσω των εσωτερικών γινομένων τους στον χώρο Η, τα οποία και πάλι θα εμφανίζονται στις συναρτήσεις με τη μορφή $\Phi(x_i)^T \cdot \Phi(x_j)$.

Θεωρώντας ότι στο νέο χώρο τα δεδομένα εκπαίδευσης είναι γραμμικά διαχωρίσιμα, ακολουθούμε την διαδικασία που χρησιμοποιήσαμε στον ταξινομητή διανυσμάτων υποστήριξης για την εύρεση του υπερεπίπεδου.

Πιό συγκεκριμένα, η συνάρτηση Lagrange του δυϊκού προβλήματος θα είναι

$$L_D(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} a_i a_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

ενώ για τα w και b αντίστοιχα θα έχουμε λύσεις

$$w = \sum_{i=1}^{n} a_i y_i \Phi(x_i)$$

$$b = \frac{1}{N_{sv}} \sum_{i \in N_{sv}} (y_i - w^T \Phi(x_i)) = \frac{1}{N_{sv}} \sum_{i \in N_{sv}} (y_i - a_j y_j \Phi(x_j)^T \Phi(x_i))$$

Επιπλέον η βέλτιστη διαχωριστική επιφάνεια από την οποία το $sign(f(x))$ θα μας βοηθήσει στην κατηγοριοποίηση νέων παρατηρήσεων θα είναι

$$f(x) = w^T \Phi(x) + b = \sum_{i=1}^{n} a_i y_i \Phi(x_i)^T \Phi(x) + b$$

4.3.1 Το τέχνασμα του πυρήνα

Παρατηρούμε ότι στις παραπάνω εξισώσεις εμφανίζονται τα εξωτερικά γινόμενα της μορφής $\Phi(x_i)^T \Phi(x_j)$. Επομένως μπορούμε να ορίσουμε την παρακάτω συνάρτηση, η οποία ονομάζεται συνάρτηση πυρήνα (Kernel function).

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$$

Το μεγάλο πλεονέκτημα της συνάρτησης πυρήνα είναι ότι η χρήση του K στα δεδομένα εκπαίδευσης δεν προϋποθέτει τον αναλυτικό υπολογισμό της απεικόνισης Φ. Δηλαδή μπορούμε να υπολογίσουμε τα στοιχεία $y_i y_j \Phi(x_i)^T \Phi(x_j)$ της Λαγκρασιανης συνάρτησης με το εσωτερικό γινόμενο δύο διανυσμάτων χαμηλότερων διαστάσεων απ'οτι αν τα υπολογίζαμε αφότου τα διανύσματα μετασχηματιστούν σε διανύσματα υψηλότερων διαστάσεων.

Για καλύτερη κατανόηση παρατίθεται ένα παράδειγμα:

Έστω $x_i \in \mathbb{R}^2$ με $x_i = (x_{i1}, x_{i2})$ και επιλέγουμε πυρήνα $K(x_i, x_j) = (x_i \cdot x_j)^2$. Θα βρούμε έναν χώρο H και μια απεικόνιση Φ έτσι ώστε $\Phi(x_i)\Phi(x_j) = (x_i \cdot x_j)^2$. Μια επιλογή θα μπορούσε να είναι $H = \mathbb{R}^3$ και $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$ με $\Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$, και έτσι θα ισχύει

$$K(x, y) = \Phi(x)^T \Phi(y)$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$$

$$= (x_1y_1 + x_2y_2)^2$$

$$= (x \cdot y)^2$$

4.3.2 Συναρτήσεις Πυρήνα

Οι βασικότερες και πιο διαδεδομένης χρήσης συναρτήσεις πυρήνα που χρησιμοποιούνται για την αντιμετώπιση προβλημάτων μη γραμμικότητας και έχουν αποδειχθεί ότι αντιστοιχούν σε κάποιο μετασχηματισμό Φ είναι οι εξής

Γραμμικός πυρήνας:

Η Γραμμική συνάρτηση πίνακα η οποία χρησιμοποιείται μόνο για γραμμικά διαχωρίσιμα προβλήματα

$$K(x_i, x_j) = x_i^{\mathrm{T}} \cdot x_j$$

Πολυωνυμικός πυρήνας:

Μια αρκετά δημοφιλής συνάρτηση πυρήνα που χρησιμοποιείται για μη γραμμικά διαχωρίσιμα προβλήματα και το d ορίζεται απο τον χρήστη. Για $d = 2$ η συνάρτηση ονομάζεται τετραγωνικός πυρήνας.

$$K(x_i, x_j) = (x_i^T \cdot x_j + 1)^d$$

Πυρήνας ακτινικής Βάσης-Γκαουσιανός πυρήνας (RBF Gaussian Kernel):

Μια ιδιαίτερα σημαντική συνάρτηση πυρήνα λόγω της Γκαουσσιανης μορφής της. Το σ ορίζεται από τον χρήστη.

$$K(x_i, x_j) = exp\{-\frac{\|x_i - x_j\|^2}{2\sigma^2}\}$$

Σιγμοειδής Πυρήνας (Sigmoid Kernel):

Ο σιγμοειδής πυρήνας ή πυρήνας υπερβολικής εφαπτομένης είναι ένας πυρήνας που κυρίως χρησιμοποιείται στα νευρωνικά δίκτυα και ο οποίος έχει εφαρμογή για συγκεκριμένες τιμές των a και r.

$$K(x_i, x_j) = tanh(ax_i^T x_j + r)$$

4.3.3 Υλοποίηση ΜΔΥ με χρήση R

Θα χρησιμοποιήσουμε την βάση δεδομένων Social_Network_Ads. Αυτή η βάση περιέχει 400 καταχωρίσεις όπου στην κάθε μία έχουμε την ηλικία τον μισθό και το γεγονός αν το προϊόν που διαφημίζουμε αγοράστηκε ή όχι. Σκοπός του μοντέλου είναι να προβλέψει αν δοσμένης της ηλικίας και του μισθού ο πελάτης θα αγοράσει το προϊόν.

Φορτώνουμε την βάση δεδομένων και την βιβλιοθήκη που έχει έτοιμο το μοντέλο των ΜΔΥ που θα χρησιμοποιήσουμε. Θα κρατήσουμε τις στήλες που θα χρειαστούμε και θα μετατρέψουμε το χαρακτηριστικό που θα προβλέψουμε από αριθμητικο σε κατηγορικό με 0 ο πελάτης να μην έχει αγοράσει το προϊόν και με 1 να το έχει αγοράσει.

```
# Loading the library and dataset
library(e1071)
dataset3 <- read.csv("C:/Users/titOv/R/PTUXIAKH/Social_Network_Ads.csv")
dataset2 <- dataset3[3:5]  # filtering the features we gonna use
# Changing the Purchased column to factor type object
dataset2$Purchased = factor(dataset2$Purchased, levels = c(0, 1))
```

Χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 80/20. Θέτουμε το set.seed με έναν τυχαίο αριθμό για να παίρνουμε πάντα τα ίδια αποτελέσματα και αλλάζουμε την κλίμακα των δεδομένων μας για καλύτερα αποτελέσματα στο μοντέλο μας.

```r
# Create indices (1 and 2) with 80/20 proportion to split the data
# ind==1 with probability 80% for train set and ind==2 with probability
# 20% is for test set, so we filter the data using this two indices.
set.seed(1234)
ind <- sample(2,nrow(dataset2),replace=T,prob=c(0.8,0.2))
train<- dataset2[ind==1,] #filtering the dataset to create the train set
test <- dataset2[ind==2,] #filtering the dataset to create the test set
train[-3] = scale(train[-3])  # Scaling the train set
test[-3] = scale(test[-3])  # Scaling the test set
```

Ταξινομητής διανυσμάτων υποστήριξης (Γραμμικός πυρήνας)

Δημιουργούμε το μοντέλο μας. Επιλέγουμε τον σωστό τύπο για να γίνει η ταξινόμηση και για πυρήνα κρατάμε τον γραμμικό.

```r
# Creating the model, we choose Linear kernel and Classification type
svm_linear <- svm(Purchased~.,data=train,type='C-classification',
kernel='linear')
```

Προβλέπουμε αν ο πελάτης θα αγοράσει ή όχι το προϊόν και κάνουμε τον πίνακα σύγκρισης με τα πραγματικά στοιχεία.

```r
# Make the predictions and confusion matrix
pred_linear <- predict(svm_linear,newdata = test[-3])
cm_linear<- table(pred_linear,test$Purchased)
cm_linear

##
## pred_linear  0  1
```

```
##           0 48  9
##           1  0 18
```

Η ακρίβεια του μοντέλου μας θα είναι:

```
# Calculating the accuracy of our model
accuracy_svm_linear <- sum(diag(cm_linear)) / sum(cm_linear)
accuracy_svm_linear

## [1] 0.88
```

Το μοντέλο, μας δίνει ακρίβεια 88% και απο τις 69 παρατηρήσεις ταξινομεί λάθος τις 9 όπου όλες είναι για πελάτες που προβλέψαμε ότι θα αγοράσουν το προϊόν ενώ στην πραγματικότητα δεν το είχαν αγοράσει.

Μερικά στοιχεία για το μοντέλο που δημιουργήσαμε είναι:

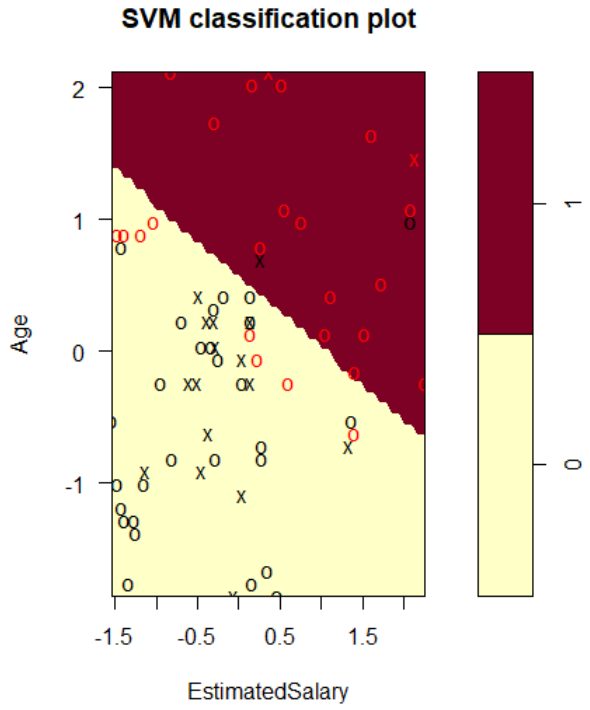
```
summary(svm_linear) # Model details

##
## Call:
## svm(formula = Purchased ~ ., data = train, type = "C-classification",
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
```

```
##
## Number of Support Vectors:  134
##
##  ( 67 67 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Το μοντέλο μας χρησιμοποίησε 134 διανύσματα υποστήριξης, 67 από αυτά για αυτούς που δεν αγοράσαν και 67 για αυτούς που αγοράσαν το προϊόν

Παραστούμε γραφικά το μοντέλο μας, όπου μπορούμε να δούμε το γραμμικό διαχωρισμό των δεδομένων

```
plot(svm_linear,data=test,Age~EstimatedSalary) # SVM plot
```



Μηχανές Διανυσμάτων Υποστήριξης (Γκαουσιανός πυρήνας Radial-RBF Gaussian Kernel)

Στο ίδιο πρόβλημα θα χρησιμοποιήσουμε τον Γκαουσιανό πυρήνα. Επιλέγουμε τον πυρήνα Radial

```
# Creating the model,now we choose Gaussian kernel.
svm_radial <- svm(Purchased~.,data=train,type='C-classification',
kernel='radial')
```

Προβλέπουμε αν ο πελάτης θα αγοράσει ή όχι το προϊόν και κάνουμε τον πίνακα σύγκρισης με τα πραγματικά στοιχεία.

```
# Prediction and confusion matrix
pred_radial <- predict(svm_radial,newdata = test[-3])
cm_radial<- table(pred_radial,test$Purchased)
cm_radial

##
## pred_radial  0  1
##           0 45  2
##           1  3 25
```

Η ακρίβεια του μοντέλου μας θα είναι:

```
# Finding the accuracy of our model
accuracy_svm_radial <- sum(diag(cm_radial)) / sum(cm_radial)
accuracy_svm_radial

## [1] 0.9333333
```

Όπως ήταν αναμενόμενο η ακρίβεια με την χρήση του Γκαουσιανού πυρήνα είναι καλύτερη από αυτή με το γραμμικό μοντέλο. Η ακρίβεια είναι 93% με 5 λάθος ταξινομήσεις με 3 λάθος προβλέψεις ότι ο πελάτης δεν θα αγοράσει το προιον και 2 ότι θα το αγοράσει ενώ στο τέλος έγινε το αντίθετο.

Οι λεπτομέρειες για το μοντέλο μας:

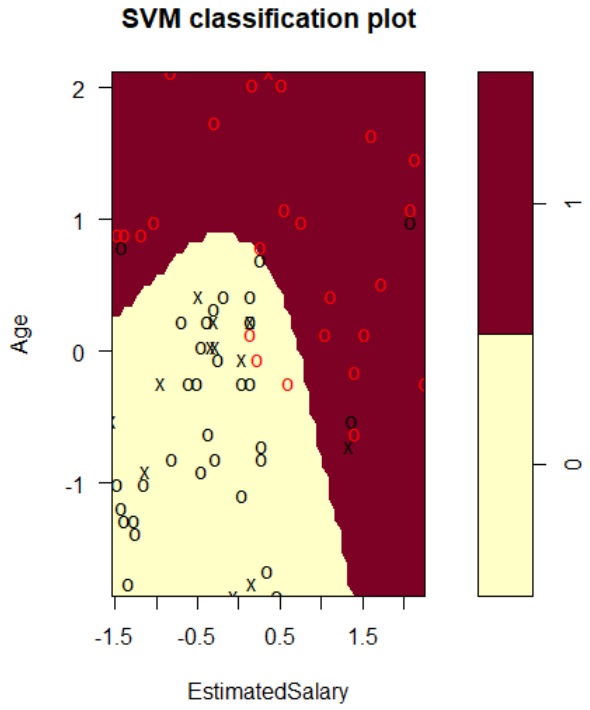
```
summary(svm_radial) # SVM radial details

##
## Call:
## svm(formula = Purchased ~ ., data = train, type = "C-classification",
```

```
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  91
##
##  ( 44 47 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Το μοντέλο μας χρησιμοποίησε 91 διανύσματα υποστήριξης, 44 από αυτά για αυτούς που δεν αγοράσαν και 47 για αυτούς που αγοράσαν το προϊόν. Παρατηρούμε ότι με τον Γκαουσιανό πυρήνα χρησιμοποιήσαμε αρκετά λιγότερα διανύσματα υποστήριξης.

Γραφική παράσταση του μοντέλου:

```
plot(svm_radial,data=test,Age~EstimatedSalary) # SVM radial plot
```



Σιγμοειδής Πυρήνας

Όμοια με τα παραπάνω για τον Σιγμοειδή πυρήνα θα έχουμε

```
# Creating the model,now we choose Sigmoid kernel.
svm_sigm <- svm(Purchased~.,data=train,type='C-classification',
                        kernel='sigmoid')
pred_sigm <- predict(svm_sigm,newdata = test[-3]) # prediction
cm_sigm<- table(pred_sigm,test$Purchased) # Confusion Matrix
cm_sigm

##
## pred_sigm  0  1
##         0 37 11
##         1 11 16
```

Με ακρίβεια

```
# Accuracy calculation
accuracy_svm_sigm <- sum(diag(cm_sigm)) / sum(cm_sigm)
accuracy_svm_sigm

## [1] 0.7066667
```

Ο Σιγμοειδής πυρήνας μας έδωσε πολύ χαμηλότερη ακρίβεια σε σχέση με τους άλλους δύο πυρήνες, με μόλις 70% ακρίβεια και 22 λάθος ταξινομήσεις.

Οι λεπτομέρειες για το μοντέλο μας

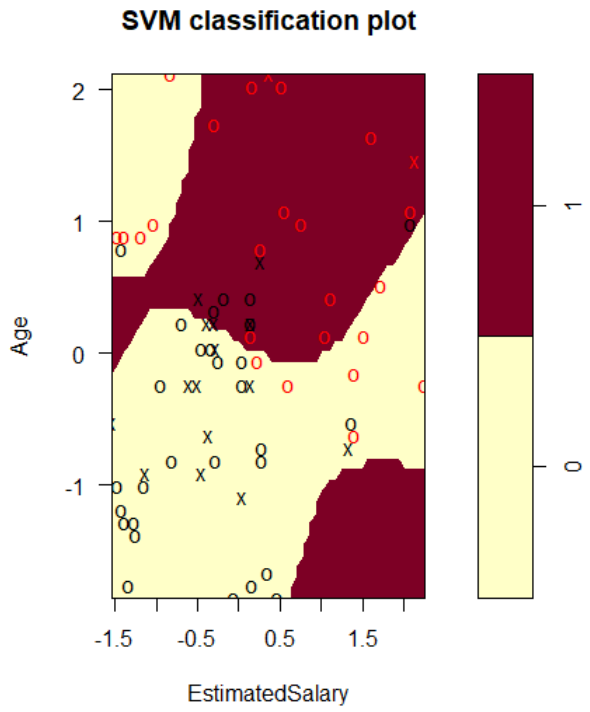
```
summary(svm_sigm) # SVM sigmoid details

##
## Call:
## svm(formula = Purchased ~ ., data = train, type = "C-classification",
##     kernel = "sigmoid")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  1
##      coef.0:  0
##
## Number of Support Vectors:  120
##
##  ( 60 60 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Για την κατασκευή του μοντέλου χρειάστηκαν 120 διανύσματα υποστήριξης, 60 για κάθε είδος ταξινόμησης.

Και γραφική παράσταση του μοντέλου

```
plot(svm_sigm,data=test,Age~EstimatedSalary) # SVM sigmoid plot
```



Πολυωνυμικός Πυρήνας

Με επιλογή του πολυωνυμικού πυρήνα τρίτου βαθμού θα έχουμε

```
# SVM model creation with polynomial kernel
svm_3d <- svm(Purchased~.,data=train,type='C-classification',
                        kernel='polynomial')
pred_3d <- predict(svm_3d,newdata = test[-3]) # predictions
cm_3d<- table(pred_3d,test$Purchased) # Confusion Matrix
cm_3d

##
## pred_3d  0  1
##       0 48  9
##       1  0 18
```

Με ακρίβεια

```
accuracy_svm_3d <- sum(diag(cm_3d)) / sum(cm_3d)
accuracy_svm_3d # Model accuracy calculation

## [1] 0.88
```

Ο αλγόριθμος μηχανών διανυσμάτων υποστήριξης με πολυωνυμικό πυρήνα τρίτου βαθμού μας δίνει ακρίβεια 88% και 9 λάθος ταξινομήσεις και ουσιαστικά τα ίδια αποτελέσματα με τον γραμμικό πυρήνα. Παρακάτω θα τρέξουμε τον αλγόριθμο σε εκατό διαφορετικές παραλλαγές των δεδομένων και θα βγάλουμε ασφαλέστερα συμπεράσματα.

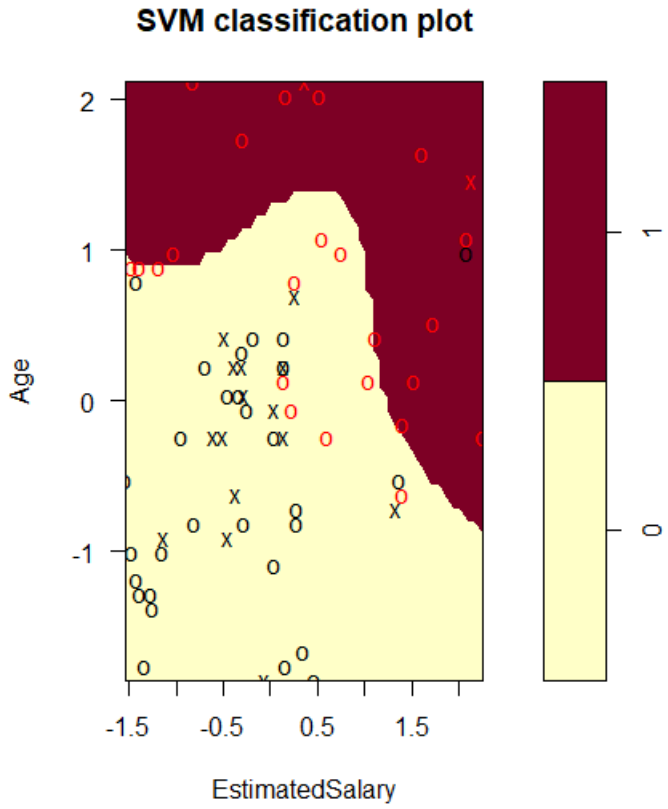
Οι λεπτομέρειες για το μοντέλο μας

```r
summary(svm_3d) # SVM Polynomial details
```

```
##
## Call:
## svm(formula = Purchased ~ ., data = train, type = "C-classification",
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  130
##
##  ( 64 66 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Και γραφική παράσταση του μοντέλου μας

```
plot(svm_3d,data=test,Age~EstimatedSalary) # SVM polynomial plot
```



4.4 Παλινδρόμηση Διανυσμάτων υποστήριξης

4.4.1 Εφαρμογή Διανυσμάτων Υποστήριξης σε προβλήματα Παλινδρόμησης

Οι μηχανές διανυσμάτων υποστήριξης μπορούν επίσης να χρησιμοποιηθούν και ως μέθοδος παλινδρόμησης, διατηρώντας όλα τα κύρια χαρακτηριστικά που χαρακτηρίζουν τον αλγόριθμο (μέγιστο περιθώριο). Η παλινδρόμηση διανυσμάτων υποστήριξης (Support Vector Regression - SVR) χρησιμοποιεί τις ίδιες αρχές με την μηχανή διανυσμάτων υποστήριξης με ελάχιστες μόνο διαφορές. Σε αυτή την περίπτωση το πρόβλημα διαμορφώνεται ως ένα πρόβλημα

βελτιστοποίησης ορίζοντας πρώτα μια συνάρτηση απώλειας ε-ευαισθησίας (ε-insensitive error function) που πρέπει να ελαχιστοποιηθεί και να βρεθεί το μικρότερο δυνατο περιθώριο που να περιέχει όσο το δυνατό περισσότερες παρατηρήσεις των δεδομένων εκπαίδευσης.

Θα βρούμε πρώτα την γραμμική συνάρτηση $f(x) = w^T x + b$ με χρήση του SVR. Στην γραμμική παλινδρόμηση ελαχιστοποιούσαμε την συνάρτηση σφάλματος

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i)) + \frac{1}{2}\lambda \| w \|^2$$

Τώρα στο SVR για να εξασφαλίσουμε ότι το περιθώριο θα είναι όσο το δυνατό στενότερο θα πρέπει το μέγεθος του w να είναι όσο το δυνατό μικρότερο, γι'αυτό θα πρέπει να ισχύει

$$min\frac{1}{2}\| w \|^2$$

με τον περιορισμό ότι

$$|y_i - (w^T x_i + b)| \leq \varepsilon$$

Θεωρούμε την συνάρτηση απώλειας ε-ευαισθησίας η οποία μας βοηθάει να αγνοήσουμε τα σφάλματα όταν βρίσκονται σε απόσταση μικρότερη του ε. Η απώλεια υπολογίζεται με βάση την απόσταση μεταξύ της παρατηρούμενης τιμής y και του ορίου ε.

$$L_\varepsilon(y, f(x)) = \begin{cases} 0 & |y - (w^T x + b)| \leq \varepsilon \\ |y - (w^T x + b)| - \varepsilon & |y - (w^T x + b)| \geq \varepsilon \end{cases}$$

Έτσι η συνάρτηση σφάλματος της γραμμικής παλινδρόμησης με χρήση της συνάρτησης απώλειας ε-ευαισθησίας με λ σταθερά κανονικοποίησης θα γίνει

$$\sum_{i=1}^{n} L_{\varepsilon}(y_i - f(x_i)) + \frac{1}{2}\lambda \| w \|^2$$

Άμα χρησιμοποιήσουμε και μεταβλητές χαλαρότητας στο πρόβλημα μας η συνάρτηση σφάλματος θα έχει την μορφή

$$\sum_{i=1}^{n} (\xi_i - \xi_i^*) + \frac{1}{2}\lambda \| w \|^2$$

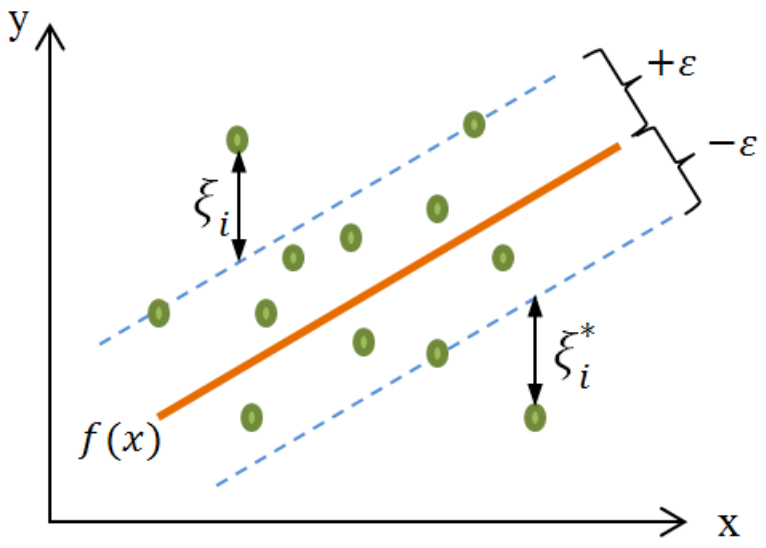
με τους παρακάτω περιορισμούς:

$$y_i - (w^T x_i + b) \leq \varepsilon + \xi_i$$

$$(w^T x_i + b) - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i \geq 0$$

$$\xi_i^* \geq 0$$



Σχήμα 4.5: Παράδειγμα γραμμικής παλινδρόμηση διανυσμάτων υποστήριξης. [26]

όπου ξ_i αντιστοιχεί σε ένα σημείο για το οποίο $y_i > f(x_i) + \varepsilon$ και ξ_i^* αντιστοιχεί σε ένα σημείο για το οποίο $y_i < f(x_i) - \varepsilon$.

Η συνάρτηση Lagrange του αρχικού προβλήματος θα είναι

$$L_P(a) = C\sum_{i=1}^{n}(\xi_i - \xi_i^*) + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}(\mu_i\xi_i + \mu_i^*\xi_i^*) + \sum_{i=1}^{n} a_i^*(y_i - w^T x_i - \varepsilon - \xi_i^*) + \sum_{i=1}^{n} a_i(-y_i + w^T x_i - \varepsilon - \xi_i)$$

Ενώ η συνάρτηση Lagrange του δυικου προβλήματος η οποία θα πρέπει να ελαχιστοποιηθεί θα είναι

$$L_D(a) =$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i - a_i^*)(a_j - a_j^*)K(x_i, x_j) + \varepsilon\sum_{i=1}^{n}(\alpha_\iota + \alpha_\iota^*) + \sum_{i=1}^{n} y_i(a_i^* - a_i)$$

με περιορισμούς

$$0 \leq \alpha_i, a_i^* \leq C$$

$$\sum_{i=1}^{n}(a_i^* - a_i) = 0$$

$$a_i a_i^* = 0$$

Με $C = \frac{1}{\lambda}$, $K(x_i, x_j)$ ο πυρήνας που θα επιλέξουμε, α_iκαι a_i^* να είναι οι πολλαπλασιαστές Lagrange

Τέλος, η συνάρτηση που θα χρησιμοποιήσουμε για να προβλέψουμε νέες τιμές θα είναι

$$f(x) = \sum_{i=1}^{n}(a_i - a_i^*)K(x, x_i) + b$$

Οι συνθήκες συμπληρωματικότητας του KKT είναι

$$a_i(\varepsilon + \xi_i - y_i + f(x_i)) = 0$$

$$a_i^*(\varepsilon + \xi_i^* + y_i - f(x_i)) = 0$$

$$\xi_i(C - a_i) = 0$$

$$\xi_i^*(C - a_i^*) = 0$$

με $b = y_i - \varepsilon - \sum_{j=1}^{n}(a_j - a_j^*)K(x_i, x_j)$.

4.4.2 Υλοποίηση ΠΔΥ στην R

Θα χρησιμοποιήσουμε ως βάση δεδομένων το Position_Salaries.csv. Η βάση περιέχει τους μισθούς μιας εταιρίας ανάλογα με τον επίπεδο και τον τίτλο της θέσης. Σκοπός μας είναι να δημιουργήσουμε ένα μοντέλο που θα προβλέπει τον μισθό ενός υπαλλήλου με βάση την θέση του.

Φορτώνουμε την βάση δεδομένων και τις βιβλιοθήκες που θα χρειαστούμε. Αφαιρούμε την πρώτη στήλη μιας και έχουμε την πληροφορία από το αριθμό του επιπέδου.

```
# Loading the libraries and datasets
library(e1071)
library(ggplot2)
dataset <- read.csv("C:/Users/titOv/R/PTUXIAKH/Position_Salaries.csv")
dataset <- dataset[2:3] # Subsetting the data we gonna use
```

Δημιουργούμε το μοντέλο μας. Επιλέγουμε τον τύπο 'eps-regression' γιατι έχουμε να κάνουμε με πρόβλημα παλινδρόμησης.

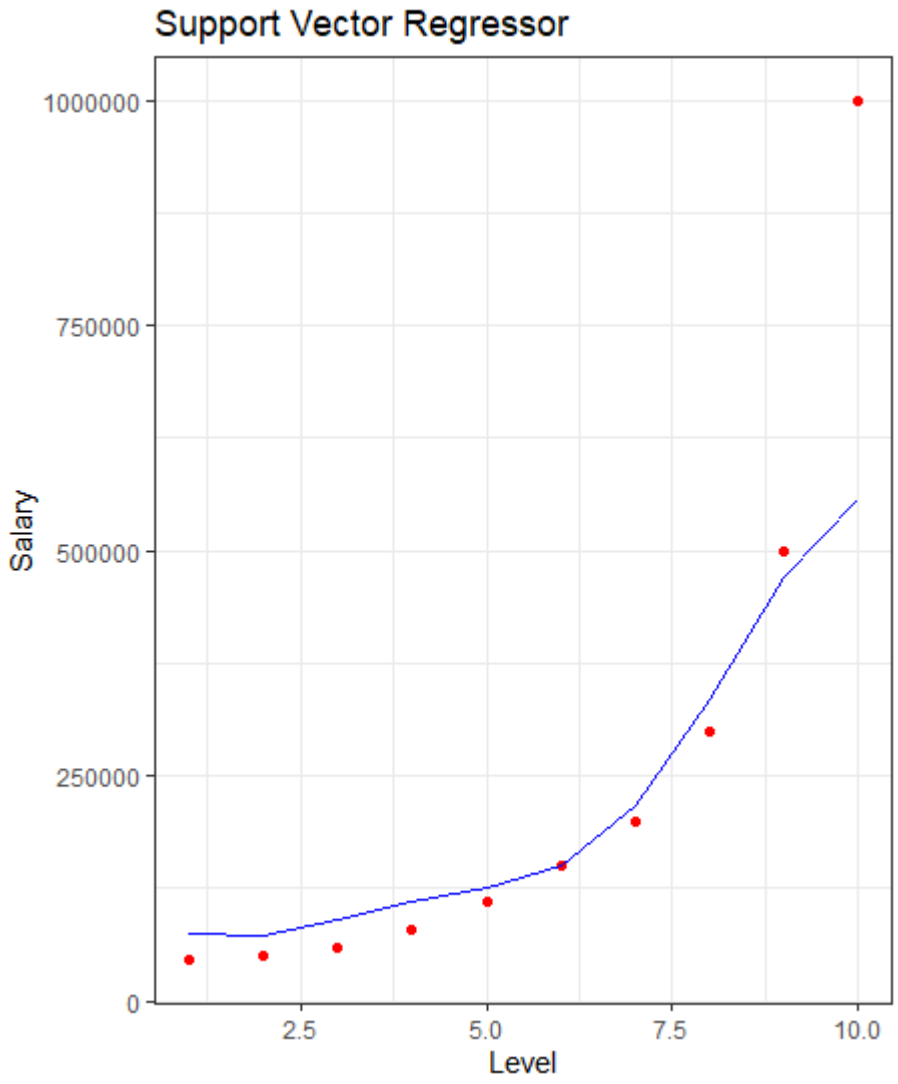
```
# creating the SVR model, we gonnna choose eps-regression for our
# regression problem
svr_model <- svm(Salary~., data=dataset, type="eps-regression")
```

Προβλέπουμε τον μισθό ενός υπαλλήλου με θέση που έχει νούμερο επιπέδου 7

```
svr_pred <- predict(svr_model, data.frame(Level=7))
svr_pred # prediction

##        1
## 217486.2
```

Κάνουμε ένα γράφημα για να δούμε πως ανταποκρίνεται το μοντέλο μας σε σχέση με τις πραγματικές τιμές.

```
# Model plot
ggplot() +
geom_point(aes(x = dataset$Level, y = dataset$Salary),colour = 'red') +
geom_line(aes(x = dataset$Level, y = predict(svr_model,
newdata = dataset)), colour = 'blue') +
ggtitle('Support Vector Regressor') +
xlab('Level') +   ylab('Salary') + theme_bw()
```



Βλέπουμε πως ο αλγόριθμος παλινδρόμησης διανυσμάτων υποστήριξης εφαρμόζει πολύ καλά στην βάση δεδομένων μας. Προβλέπει τιμές λίγο μεγαλύτερες από την πραγματική για θέσεις μέχρι το 8 επίπεδο ενώ μας δίνει κακές προβλέψεις για θέσεις επιπέδου 10 με σχεδόν τον μισό μισθό από τον πραγματικό.

4.5 Σύνοψη Κεφαλαίου - Συγκριτικό Αλγορίθμων

Σε αυτό το κεφάλαιο παρουσιάσαμε τις μηχανές διανυσμάτων υποστήριξης. Είδαμε την γεωμετρική σημασία των μηχανών διανυσμάτων υποστήριξης και πως αυτή την ιδέα μπορούμε να την μετατρέψουμε σε ένα αλγόριθμο που θα μας βοηθήσει σε προβλήματα ταξινόμησης. Παρουσιάσαμε μια πιό χαλαρή προσέγγιση των μηχανών διανυσμάτων υποστήριξης εισάγοντας μια μεταβλητή χαλαρότητας ενώ τέλος μάθαμε για το τέχνασμα του πυρήνα που μας βοήθησε να ξεφύγουμε από την γραμμικότητα του απλού ταξινομητή διανυσμάτων υποστήριξης.

Επίσης είδαμε πως μπορούν να εφαρμοστούν οι μηχανές διανυσμάτων υποστήριξης σε προβλήματα παλινδρόμησης και εφαρμόσαμε τον αλγόριθμο στην βάση δεδομένων που χρησιμοποιήσαμε στο δεύτερο κεφάλαιο. Ο αλγόριθμος ΠΔΥ μας έδωσε καλύτερο αποτέλεσμα σε σχέση με την γραμμική και την πολυωνυμική παλινδρόμηση μέχρι και τρίτου βαθμού ενώ μας δίνει το ίδιο περίπου σφάλμα με την πολυωνυμική παλινδρόμηση τετάρτου βαθμού με την διαφορά ότι ο ΠΔΥ προβλέπει τιμή μεγαλύτερη από την πραγματική (overpredicting), ενώ η πολυωνυμική παλινδρόμηση προβλέπει τιμή μικρότερη από την πραγματική (underpredicting).

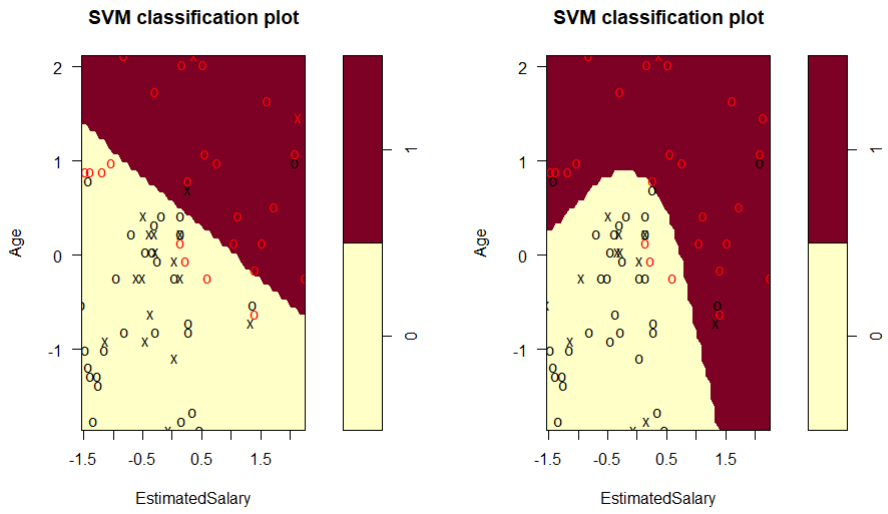
Τρόπος Υπολογισμού	Μισθός
Πραγματική τιμή	200000
Γραμμική Παλινδρόμηση	370818
Πολυωνυμική Παλινδρόμηση 2ου βαθμού	254227
Πολυωνυμική Παλινδρόμηση 3ου βαθμού	177594
Πολυωνυμική Παλινδρόμηση 4ου βαθμού	184003
Παλινδρόμηση διανυσμάτων υποστήριξης	217486

Πίνακας 5: Σύγκριση πρόβλεψης μισθών για έναν υπάλληλου επιπέδου 7

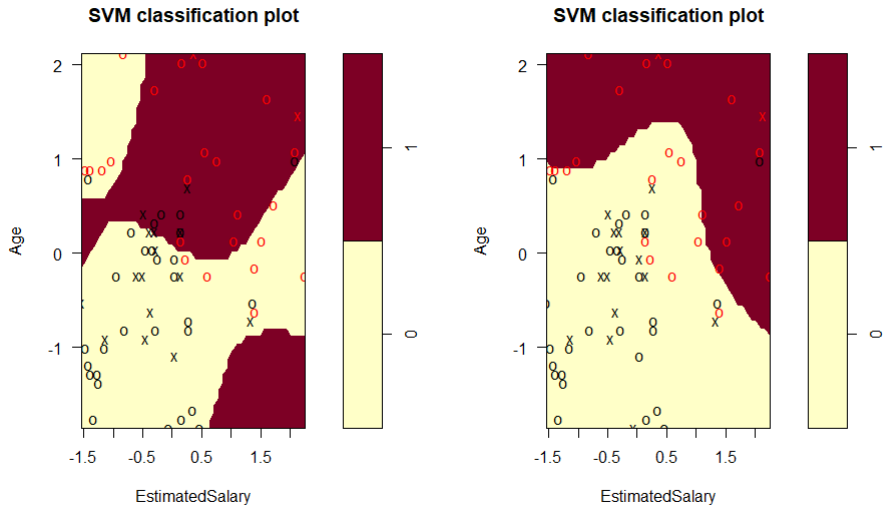
Τέλος εφαρμόσαμε τους διάφορους πυρήνες για τις μηχανές διανυσμάτων υποστήριξης στην ίδια βάση δεδομένων με σκοπό να κάνουμε σύγκριση σχετικά με την απόδοση του κάθε μοντέλου. Όπως φαίνεται στον παρακάτω πίνακα και τα γραφήματα ο γκαουσιανός πυρήνας μας δίνει καλύτερα αποτελέσματα από τους υπόλοιπους πυρήνες. Εκτός της ακρίβειας που είναι μεγαλύτερη βλέπουμε ότι με τον γκαουσιανό πυρήνα έχουμε επίσης υψηλή ειδικότητα, υψηλότερη απ'όλα τα άλλα μοντέλα η οποία είναι αρκετά σημαντική για να θυσιάσουμε την ελαφρώς χειρότερη ευαισθησία σε σχέση με τον γραμμικό και τον πολυωνυμικό πυρήνα.

Πυρήνας ΜΔΥ	Ακρίβεια	Ευαισθησία	Ειδικότητα
Γραμμικός	0.88	1	0.66
Γκαουσιανός	0.93	0.93	0.92
Σιγμοειδής	0.70	0.77	0.59
Πολυωνυμικός 3ου βαθμού	0.88	1	0.66

Πίνακας 6: Σύγκριση ΜΔΥ για διάφορους πυρήνες



(α′) Linear Kernel (β′) Gaussian Kernel



(γ′) Sigmoid Kernel (δ′) Polynomial Kernel

Σχήμα 4.6: Συγκριτικό γράφημα SVM για τους διάφορους πυρήνες

Θα τρέξουμε τώρα τα μοντέλα μας 100 φορές (20 επαναλήψεις του 5-fold Cross Validation) για εκατό διαφορετικούς διαχωρισμούς των δεδομένων έτσι ώστε να βγάλουμε ασφαλέστερα συμπεράσματα σχετικά με την ακρίβεια.

```r
# Loading the dataset and library needed
library(caret)
dataset <- read.csv("C:/Users/tit0v/R/PTUXIAKH/Social_Network_Ads.csv")
dataset <- dataset[3:5]
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
set.seed(1928)
# Creating the 5-fold cross validation and repeat it 20 teams.
cvfolds2 <- createMultiFolds(dataset$Purchased ,k=5,times=20)
# Creating the function to control the training parameters
tcontrol2 <- trainControl(method = "repeatedcv", number = 5,
repeats = 20, index=cvfolds2)
# SVM Linear model
svml2 <- train(Purchased ~ ., data = dataset, method = "svmLinear",
trControl = tcontrol2)
# SVM Radial model
svmr2 <- train(Purchased ~ ., data = dataset, method = "svmRadial",
trControl = tcontrol2)
# SVM Polynomial model
svmp2 <- train(Purchased ~ ., data = dataset, method = "svmPoly",
trControl = tcontrol2,tuneGrid = data.frame(degree=3,scale=0.100,C=1))
```

```r
t.test(svml2$resample$Accuracy)$estimate # SVM Linear mean Accuracy

## mean of x
## 0.8386074

t.test(svml2$resample$Accuracy)$conf.int # SVM Linear 95% CI
```

```
## [1] 0.8314255 0.8457893
## attr(,"conf.level")
## [1] 0.95

t.test(svmr2$resample$Accuracy)$estimate # SVM Radial mean Accuracy

## mean of x
## 0.9135806

t.test(svmr2$resample$Accuracy)$conf.int # SVM Radial 95% CI

## [1] 0.9077347 0.9194266
## attr(,"conf.level")
## [1] 0.95

t.test(svmp2$resample$Accuracy)$estimate # SVM Polynomial mean Accuracy

## mean of x
## 0.9003484

t.test(svmp2$resample$Accuracy)$conf.int # SVM Polynomial 95% CI

## [1] 0.8942939 0.9064029
## attr(,"conf.level")
## [1] 0.95
```

Ο συγκεντρωτικός πίνακας των αποτελεσμάτων μας

Πυρήνας ΜΔΥ	Mean	min	max	sd	95% CI
Γραμμικός	0.8386	0.7375	0.9241	0.0361	0.8314-0.8457
Γκαουσιανός	0.9135	0.8228	0.9873	0.0294	0.9077-0.9194
Πολυωνυμικός 3ου βαθμού	0.9	0.8354	0.9750	0.0305	0.8942-0.9064

Πίνακας 7: Σύγκριση ακρίβειας των διαφόρων μοντέλων ΜΔΥ μετά απο 100 επαναλήψεις

Παρατηρούμε ότι ο γκαουσιανός πυρήνας συνεχίζει να είναι η καλύτερη επιλογή για το πρόβλημα μας. Παρατηρούμε επίσης πόσο χαμηλότερη είναι η ακρίβεια του γραμμικού πυρήνα σε σχέση με το αρχικό πείραμα και πως ο πολυωνυμικός πυρήνας τα πάει κατά μέσο όρο καλύτερα. Άρα αν και στο αρχικό πείραμα ο γραμμικός και ο πολυωνυμικός μας έδιναν ακριβώς τα ίδια αποτελέσματα βλέπουμε πως η διαφορά είναι μεγάλη και πέριπου της τάξης του 6% . Πάλι βλέπουμε πόσο σημαντικό είναι να τρέχουμε τους αλγόριθμους μας σε διάφορες παραλλαγές των δεδομένων μας, άμα θέλουμε να έχουμε ακριβή συμπεράσματα και να μην οδηγούμαστε σε ανακρίβειες.

5 Αλγόριθμοι Δένδρων

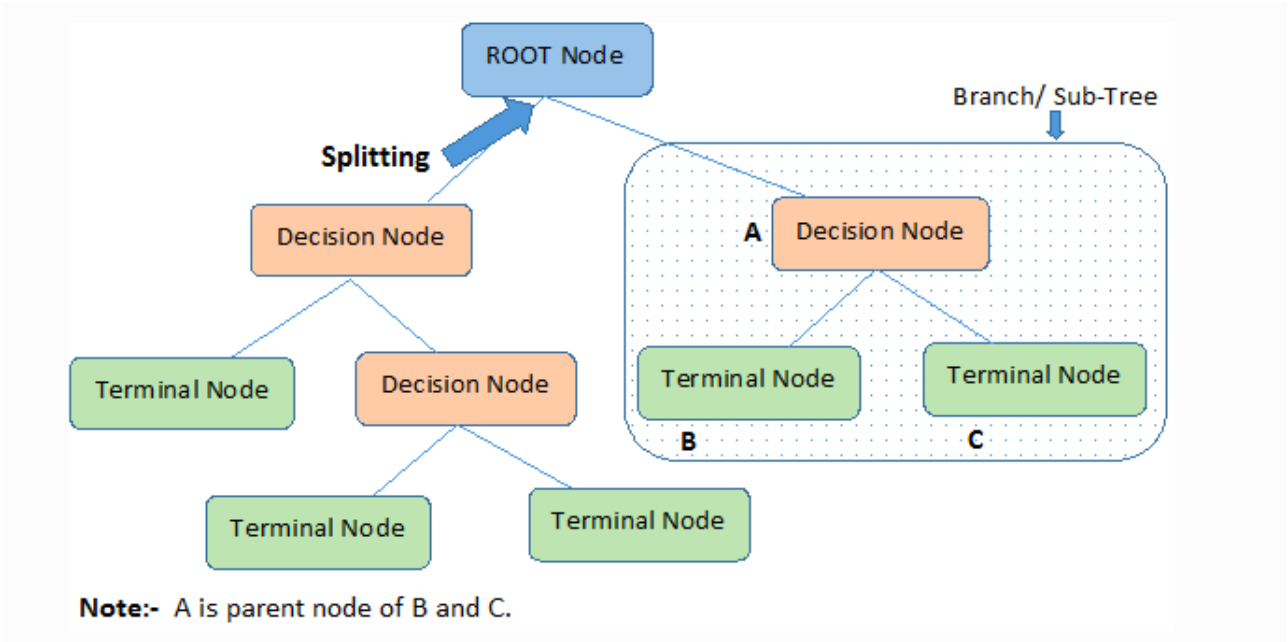
5.1 Δένδρο απόφασης

5.1.1 Παρουσίαση αλγορίθμου Δένδρου απόφασης

Ένα δέντρο απόφασης αντιπροσωπεύει μια συνάρτηση που λαμβάνει ως είσοδο ένα διάνυσμα χαρακτηριστικών τιμών και επιστρέφει μια απόφαση σαν τιμή εξόδου. Οι τιμές εισόδου και εξόδου μπορεί να είναι διακριτές ή συνεχείς. Γι'αυτό τον λόγο τα δέντρα απόφασης χρησιμοποιούνται στην εξόρυξη δεδομένων είτε σε προβλήματα ταξινόμησης είτε σε προβλήματα παλινδρόμησης.

Ένα δέντρο αποφάσης έχει την δομή ενός δένδρου, όπου ο εσωτερικός κόμβος αντιπροσωπεύει ένα χαρακτηριστικό, ο κλάδος αντιπροσωπεύει έναν κανόνα απόφασης και κάθε κόμβος φύλλων αντιπροσωπεύει το αποτέλεσμα. Ο κόμβος κορυφής σε ένα δέντρο απόφασης είναι γνωστός ως ο κόμβος ρίζας (Root node), και από αυτόν μαθαίνει να διαχωρίζεται με βάση την τιμή του χαρακτηριστικού σε επιμέρους κόμβους (Decision node), ενώ αυτός ο διαχωρισμός είναι επαναλαμβανόμενος μέχρι να φτάσουμε στον τερματικό κόμβο (Terminal node) που θα μας δώσει την απόφαση.

Κάθε κόμβος στο δέντρο λειτουργεί σαν μια δοκιμαστική συνθήκη για κάποιο χαρακτηριστικό και κάθε ακμή που φεύγει απο τον κόμβο αντιστοιχεί στις πιθανές απαντήσεις σχετικά με την δοκιμαστική συνθήκη. Αυτή η διαδικασία έχει αναδρομική φύση και επαναλαμβάνεται για κάθε κλάδο (Branch/ Sub-Tree).



Σχήμα 5.1: Γενική μορφή ενός δέντρου απόφασης [15]

Πως λειτουργεί όμως ο αλγόριθμος των δέντρων αποφάσεων;

1. Στην αρχή θεωρούμε ολό το σύνολο των δεδομένων σαν κόμβο ρίζας από το οποίο επιλέγουμε το καλύτερο χαρακτηριστικό χρησιμοποιόντας τα Μετρα Επιλογής Χαρακτηριστικών (Attribute Selection Measures-ASM) .

2. Καθιστά πλέον το επιλεγμένο χαρακτηριστικό σαν κόμβο ρίζας και χωρίζει τα δεδομένα σε μικρότερα υποσύνολα δεδομένων.

3. Ο αλγόριθμος επαναλαμβάνει την διαδικασία για τα εκάστοτε υποσύνολα των δεδομένων για τα εναπομείνατα χαρακτηριστικα.

4. Ο αλγόριθμος σταματά όταν καλυφθούν όλες οι περιπτώσεις για όλα τα

χαρακτηριστικά.

Τα δέντρα αποφάσεων χρησιμοποιούν διάφορους αλγορίθμους για να αποφασίσουν να χωρίσουν ένα κόμβο σε δύο ή περισσότερους υπόκομβους. Η δημιουργία υποκόμβων αυξάνει την την ομοιογένεια των υποκόμβων που επακολουθούν. Επίσης η επιλογή ενός αλγορίθμου εξαρτάται και από είδος του προβλήματος (παλινδρόμηση ή ταξινόμηση). Οι πιό διάσημοι αλγόριθμοι των δέντρων αποφάσεων είναι οι εξής:

1. ID3 (Iterative Dichotomiser 3)
2. C4.5 (Διάδοχος του ID3)
3. CART (Classification And Regression Tree)
4. CHAID (Chi-square automatic interaction detection)
5. MARS (Multivariate Adaptive Regression Splines)

Παρά τις διαφορές τους, όλοι οι αλγόριθμοι χτίζουν δέντρα αποφάσεων χρησιμοποιώντας μία προσέγγιση από πάνω προς τα κάτω άπληστης αναζήτησης (greedy search) μέσα από το χώρο των πιθανών κλάδων χωρίς να γυρίζουν ξανά προς τα πάνω. Ένας άπληστος αλγόριθμος, όπως υποδηλώνει το όνομα, κάνει πάντα την επιλογή που φαίνεται να είναι η καλύτερη την δεδομένη στιγμή.

5.1.2 Μετρα Επιλογής Χαρακτηριστικών (ASM)

Εάν το σύνολο των δεδομένων αποτελείται απο n χαρακτηριστικά, τότε το να αποφασίσουμε ποιό χαρακτηριστικό θα τοποθετηθεί στην ρίζα ή στα δια-

φορετικά επίπεδα του δέντρου ως εσωτερικοί κόμβοι είναι ένα αρκετά πολύπλοκο βήμα. Επιλέγοντας τυχαίο χαρακτηριστικό για τον κάθε κόμβο θα μας οδηγήσει σε κακά αποτελέσματα με χαμηλή ακρίβεια. Για να λυθεί το πρόβλημα της εύρεσης του καταλληλότερου χαρακτηριστικού, μερικά από τα πιο δημοφιλή κριτήρια είναι τα εξής: Entropy, Information gain, Gini index, Gain ratio, Reduction in Variance, Chi-Square. Ανάλογα το είδος του προβλήματος και του αλγόριθμου θα πρέπει να επιλέξουμε και το κατάλληλο ASM.

Εντροπία

Η εντροπία (Entropy) είναι ένα μέτρο της τυχαιότητας στις πληροφορίες που επεξεργαζόμαστε. Όσο υψηλότερη είναι η εντροπία, τόσο πιο δύσκολο είναι να αντληθούν συμπεράσματα από αυτές τις πληροφορίες.

Ο τύπος για τον υπολογισμό της εντροπίας για μία μεταβλητή είναι ο εξής

$$E(S) = \sum_i -p_i log_2 p_i$$

όπου p_i η πιθανότητα του γεγονότος που βρίσκεται σε τωρινή κατάσταση S να ανήκει στην κατάσταση C_i

ενώ η εντροπία για πολλαπλά χαρακτηριστικά θα δίνεται από τον τύπο

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

με Τ την τωρινή κατάσταση και X τα επιλεγμένα χαρακτηριστικά.

Κέρδος Πληροφορίας

Το κέρδος πληροφορίας (Information Gain) είναι η μείωση της εντροπίας. Το κέρδος πληροφορίας υπολογίζει τη διαφορά της εντροπίας πριν την διάσπαση και της μέσης εντροπίας μετά την διάσπαση του συνόλου δεδομένων με βάση τις δεδομένες τιμές των χαρακτηριστικών. Μαθηματικά το κέρδος πληροφορίας δίνεται από τον τύπο

$$IG(T, X) = E(T) - E(T, X)$$

ή αλλιώς

$$IG = Entropy(\pi\rho\iota\nu) - \sum_{j=1}^{K} Entropy(j, \mu\varepsilon\tau\alpha)$$

με "πριν" να είναι το σύνολο των δεδομένων πριν την διάσπαση, K ο αριθμός των υποσυνόλων και $(j, \mu\varepsilon\tau\alpha)$ το υποσύνολο j μετά την διάσπαση.

Δείκτης Gini

Ο δείκτης Gini (Gini index) είναι ένα μέτρο για το πόσο συχνά ένα τυχαία επιλεγμένο στοιχείο από το σετ θα χαρακτηριστεί λανθασμένα αν ήταν τυχαία επισημασμένο σύμφωνα με την κατανομή ετικετών στο υποσύνολο. Ο δείκτης Gini θεωρεί ότι ο διαχωρισμός για κάθε χαρακτηριστικό είναι δυαδικός και υπολογίζεται αφαιρώντας από την μονάδα το άθροισμα των τετραγώνων των πιθανοτήτων κάθε κατηγορίας.

$$Gini(S) = 1 - \sum_{i=1}^{n} (p_i)^2$$

όπου p_i η πιθανότητα του γεγονότος που βρίσκεται σε τωρινή κατάσταση S να ανήκει στην κατάσταση C_i.

Εαν ένας δυαδικός χωρισμός ενός χαρακτηριστικού A χωρίζει τα δεδομένα S σε S_1 και S_2, τότε ο δείκτης Gini θα είναι

$$Gini_A(S) = \frac{|S_1|}{|S|}Gini(S_1) + \frac{|S_2|}{|S|}Gini(S_2)$$

Στην περίπτωση προβλημάτων με διακριτές τιμές, το υποσύνολο με το χαμηλότερο δείκτη Gini επιλέγεται ως το χαρακτηριστικό ως προς το οποίο θα γίνει ο διαχωρισμός. Στην περίπτωση προβλημάτων με συνεχής τιμές, η στρατηγική είναι να επιλεγεί από κάθε σημείο κοντινών τιμών το σημείο με τον χαμηλότερο δείκτη Gini και ως προς αυτό το σημείο να γίνει ο διαχωρισμός.

Αναλογία κέρδους

Το κέρδος πληροφορίας μεροληπτεί όταν πρόκειται για χαρακτηριστικά με πολλά αποτελέσματα. Αυτό σημαίνει ότι προτιμά χαρακτηριστικά με μεγάλο αριθμό διακριτών τιμών. Η αναλογία κέρδους (Gain ratio) υπερνικά το πρόβλημα λαμβάνοντας υπόψη τον αριθμό των κλάδων που θα προκύψουν πριν γίνει η διάσπαση.

$$Gain\ Ratio = \frac{Information\ Gain}{Split\ Info} = \frac{Entropy(\pi\rho\iota\nu) - \sum_{j=1}^{K} Entropy(j, \mu\varepsilon\tau\alpha)}{\sum_{j=1}^{K} w_j log_2 w_j}$$

Μείωση διακύμανσης

Η μείωση της διακύμανσης (Variance Reduction) είναι ένας αλγόριθμος που χρησιμοποιείται για συνεχείς μεταβλητες. Αυτός ο αλγόριθμος χρησιμοποιεί τον κλασικό τύπο της διακύμανσης για να επιλέξει τον καλύτερο διαχωρισμό. Η διαίρεση με την χαμηλότερη διακύμανση επιλέγεται ως κριτήριο για την διάσπαση του συνόλου.

$$Variance = \frac{\sum(X - \bar{X})^2}{n}$$

με $\bar{X}$ την μέση τιμή όλων των τιμών και X την συγκεκριμένη τιμή, ενώ n είναι ο αριθμός των τιμών. Για τον υπολογισμό της διακύμανσης πρώτα υπολογίζουμε την διακύμανση του κάθε κόμβου και μετά την διακύμανση της κάθε διάσπασης ως το σταθμισμένο μέσο όρο σε σχέση με την αντίστοιχη διακύμανση του κάθε κόμβου.

5.1.3 Υλοποίηση δένδρου αποφάσεων στην R

Θα υλοποιήσουμε τον αλγόριθμο των δένδρων αποφάσης στην βάση δεδομένων iris. Η βάση αποτελείται από 120 καταχωρήσεις στις οποίες έχει γίνει μέτρηση στο μήκος και πλάτος του φύλλου κάλυκος (sepal.length sepal.width), στο μήκος και πλάτος του πέταλου άνθους (petal.width, petal.lenght) καθώς και το είδος του λουλουδιού (Species).

Φορτώνουμε την βάση iris και τις βιβλιοθήκες που θα χρειαστούμε.

```
# Loading the libraries and dataset
library(rpart)
library(rpart.plot)
data("iris")
```

Χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 80/20.

```
set.seed(123)
# Create indices (1 and 2) with 80/20 proportion to split the data
# ind==1 with probability 80% for train set and ind==2 with probability
# 20% is for test set, so we filter the data using this two indices.
```

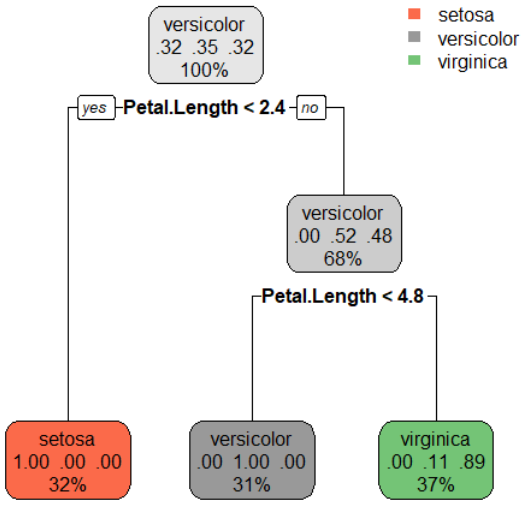
```
ind <- sample(2,nrow(iris),replace=T,prob=c(0.8,0.2))
train<- iris[ind==1,] #filtering the dataset to create the train set
test <- iris[ind==2,] #filtering the dataset to create the test set
```

Δημιουργούμε το μοντέλο το οποίο θα εκπαιδεύσουμε πάνω στα δεδομένα εκπαίδευσης με σκοπό να προβλέψουμε το είδος του λουλουδιού χρησιμοποιόντας τα μήκη και πλάτη του φύλλου και του πέταλου.

```
# fitting the model to train set using the classification method
fit <- rpart(Species~., data = train, method = 'class')
```

Παράγουμε γραφικά το δένδρο που δημιουργήσαμε.

```
# Plotting the tree we create
rpart.plot(fit)
```



Προβλέπουμε το είδος του λουλουδίου των δεδομένων δοκιμής χρησιμοποιώντας το μοντέλο που δημιουργήσαμε.

```
# Creating the vector of prediction
predict <-predict(fit, test, type = 'class')
```

Δημιουργούμε τον πίνακα σύγκρισης μεταξύ των πραγματικών ειδών και τον προβλεπόμενων ειδών

```
# Creating and printing the confusion matrix
confusion_matrix<-table(predict,test$Species)
confusion_matrix

##
## predict      setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0          7         2
##   virginica       0          0         9
```

Η ακρίβεια του μοντέλου μας θα είναι:

```
# Calculating the accuracy of the model
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
accuracy

## [1] 0.9310345
```

Το μοντέλο μας έχει 93% ακρίβεια. Από τον πίνακα σύγκρισης βλέπουμε ότι ταξινόμησε σωστά όλα τα λουλούδια setosa και virginica ενώ δεν τα

πήγε τόσο καλά με το versicolor όπου έκανε δύο λάθη ταξινομώντας τα σαν virginica.

5.2 Τυχαία Δάση

5.2.1 Συνδιαστικοί Ταξινομητές

Τα δένδρα απόφασης που εξετάσαμε στο προηγούμενο κεφάλαιο έχουν ένα σημαντικό μειονέκτημα. Υποφέρουν από πρόβλημα μεγάλης διακύμανσης το οποίο οδηγεί τον αλγόριθμο μας σε πρόβλημα υπερεκπαίδευσης (overfitting). Δηλαδή ο αλγόριθμος εκπαιδεύεται αρκετά καλά πάνω στα δεδομένα εκπαίδευσης αλλά η ακρίβεια του στα νέα δεδομένα δεν είναι αρκετά υψηλή. Επίσης άμα χωρίσουμε τυχαία τα δεδομένα εκπαίδευσης σε δύο μέρη και εκπαιδεύσουμε δένδρα απόφασης στο καθένα τα αποτελέσματα που θα πάρουμε θα είναι αρκετά διαφορετικά.

Ένας τρόπος να βελτιωθούν οι αλγόριθμοι με μεγάλη διακύμανση είναι η κατασκευή συνδυαστικών ταξινομητών (Ensemble Classifiers). Ουσιαστικά η τεχνική που ακολουθούμε είναι να συνδιάσουμε παράλληλα πολλούς αλγόριθμους μάθησης. Οι πιο κλασικές μέθοδοι συνδυαστικής μάθησης είναι η μέθοδος bootstrap aggregation ή bagging και η μέθοδος boosting.

Η μέθοδος bagging δημιουργεί πολλά “αντίγραφα” των δεδομένων εκπαίδευσης (κάθε αντίγραφο είναι ελαφρώς διαφορετικό από το άλλο) και στην συνέχεια εφαρμόζουμε τον εκάστοτε αλγόριθμο σε κάθε αντίγραφο με σκοπό να αποκτήσουμε πολλά μοντέλα τα οποία στην συνέχεια θα τα συνδυάσουμε. Η μέθοδος δεν είναι επιρρεπής σε φαινόμενα υπερεκπαίδευσης καθώς αυξάνεται το πλήθος των παραγόμενων υποθέσεων.

Η μέθοδος boosting χρησιμοποιεί τα πρωτότυπα δεδομένα εκπαίδευσης για την επαναληπτική δημιουργία πολλαπλών μοντέλων. Κάθε νέο μοντέλο θα είναι διαφορετικό από το προηγούμενο αφού κάθε δημιουργία νέου μοντέλου θα προσπαθεί να "διορθώσει" τα λάθη που κάνουν τα προηγούμενα μοντέλα. Το τελικό μοντέλο θα είναι ένας συνδιασμός όλων των προηγούμενων μοντέλων που κατασκευάστηκαν διαδοχικά. Παρόλο που η τεχνική boosting τείνει να μειώνει εκθετικά το σφάλμα εκπαίδευσης, είναι επιρρεπής σε φαινόμενα υπερεκπαίδευσης λόγω του ότι εστιάζει σε παραδείγματα που δεν ταξινομούνται σωστά.

5.2.2 Αλγόριθμος Τυχαίου Δάσους

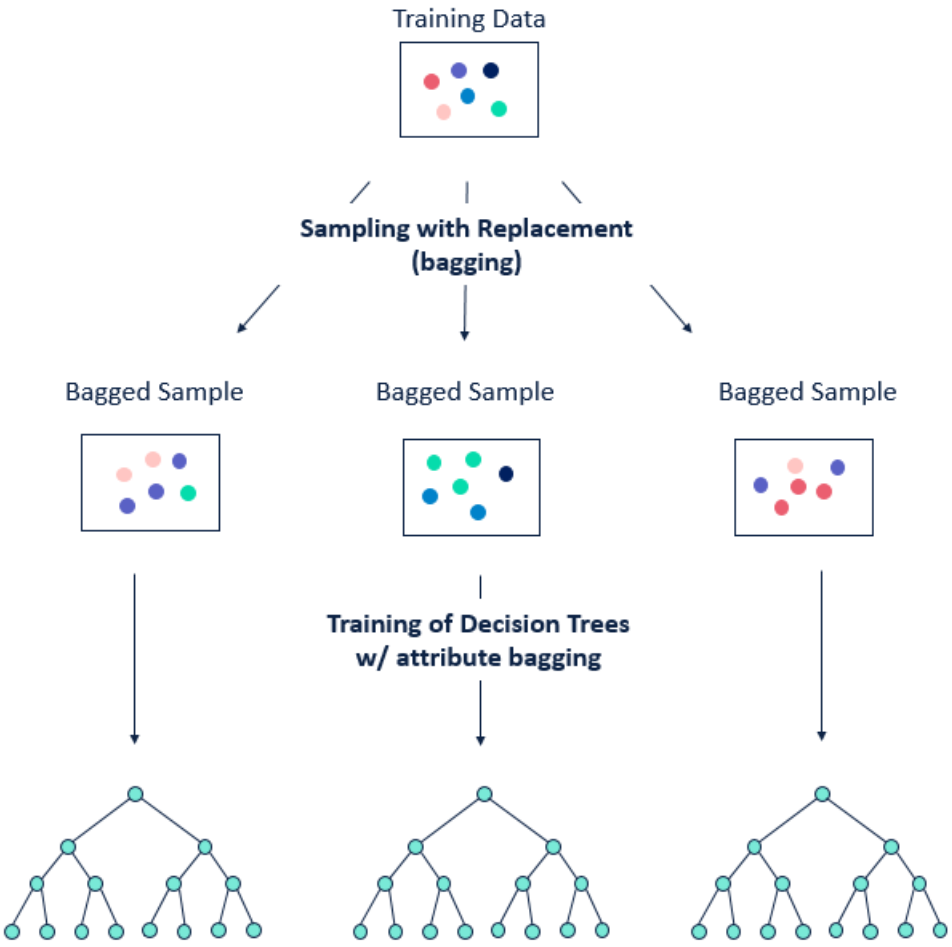
Στον αλγόριθμο Τυχαίου Δάσους (Random Forest) ουσιαστικά εφαρμόζουμε την μέθοδο bagging και χρησιμοποιόντας σε κάθε αντίγραφο τον αλγόριθμο των δένδρων ταξινόμησης παράγουμε πολλά δένδρα αποφάσης τα οποία λύνουν το πρόβλημα της μεγάλης διακύμανσης που παρατηρούμε όταν παράγουμε μοναδικό δένδρο απόφασης. Πως λειτουργεί όμως ο αλγόριθμος τυχαίου δάσους; Σε δοσμένο σύνολο δεδομένων εκπαίδευσης δημιουργούμε Β τυχαία δείγματα S_b (για κάθε $b = 1, 2, ..., B$) των δεδομένων εκπαίδευσης και κατασκευάζουμε δένδρα απόφασης f_b χρησιμοποιόντας κάθε δείγμα S_b σαν σύνολο δεδομένων. Αυτό σημαίνει ότι ξεκινάμε με ένα κενό σύνολο και στην συνέχεια επιλέγουμε τυχαία ένα δείγμα απο τα δεδομένα εκπαίδευσης το οποίο το αποθηκεύουμε στο S_b , διατηρώντας το αρχικό σύνολο των δεδομένων εκπαίδευσης.

Μετά την εκπαίδευση θα έχουμε B δένδρα απόφασης. Η πρόβλεψη για ένα νέο παράδειγμα x' λαμβάνεται ως ο μέσος όρος των B προβλέψεων. Έτσι για προβλήματα παλινδρόμησης η πρόβλεψη δίνεται από τον τύπο

$$y \leftarrow \hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

ενώ για προβλήματα ταξινόμησης θα έχουμε

$$\hat{C}_B(x') = majority\ vote\ \{\hat{C}_b(x')\}_1^B$$



Σχήμα 5.2: Γραφική παράσταση Τυχαίου Δάσους. [23]

με $\hat{C}_b(x')$ να είναι η πρόβλεψη του b δένδρου απόφασης του τυχαίου δάσους.

Η διαφορά ανάμεσα στην μέθοδο Bagging και στον αλγόριθμο Τυχαίου Δάσους είναι στο γεγονός ότι στην μέθοδο Bagging σε κάθε επανάληψη χρησι-

μοποιούνται όλα τα χαρακτηριστικά p των δεδομένων εκπαίδευσης ενώ στον αλγόριθμο τυχαίου δάσους κάθε δένδρο απόφασης που δημιουργείται χρησιμοποιεί $m < p$ χαρακτηριστικά, με την πίο συνήθη επιλογή να είναι $m \approx \sqrt{p}$ για προβλήματα ταξινόμησης και $m \approx p/3$ για προβλήματα παλινδρόμησης. Με άλλα λόγια, κατά την δημιουργία ενός τυχαίου δάσους, σε κάθε διάσπαση σε δενδρα απόφασης ο αλγόριθμος δεν έχει καν την δυνατότητα να εξετάσει την πλειοψηφία των διαθέσιμων τιμών που βοηθάν στην πρόβλεψη. Αυτός ο τρόπος λειτουργίας των τυχαίων δασών μας βοηθάει να λύσουμε το πρόβλημα της υψηλής διακύμανσης που παρατηρείται στα δένδρα απόφασης ενώ η επιλογή του m γίνεται με σκοπό να ελαττωθεί το OOB σφάλμα (out of bag error).

5.2.3 Υλοποίηση Τυχαίων δασών στην R

Θα χρησιμοποιήσουμε πάλι την βάση δεδομένων iris.

Φορτώνουμε την βάση iris και την βιβλιοθήκη που θα χρειαστούμε.

```
# Loading the libraries and dataset
library(randomForest)
data("iris")
```

Χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής με αναλογία 80/20.

```
set.seed(123)
# Create indices (1 and 2) with 80/20 proportion to split the data
# ind==1 with probability 80% for train set and ind==2 with probability
# 20% is for test set, so we filter the data using this two indices.
```

```
ind_rf <- sample(2,nrow(iris),replace=T,prob=c(0.8,0.2))
train_rf<- iris[ind_rf==1,]
test_rf <- iris[ind_rf==2,]
```

Δημιουργούμε το μοντέλο των τυχαίων δασών. Επιλέγουμε να δημιουργηθούν 500 δένδρα και σε κάθε δένδρο επιλέγουμε να κρατάμε 2 χαρακτηριστικά. Αυτό γίνεται γιατί σε προβλήματα ταξινόμησης η σύνηθες τακτική είναι να κρατάμε $\sqrt{p}$ χαρακτηριστικά. Η βάση δεδομένων μας έχει τέσσερα χαρακτηριστικά άρα κάθε δένδρο μας θα χρησιμοποιεί μόνο τα δύο από αυτα.

```
# Creating the random forest model with 500 trees using 2 feature
# for every tree
rf_model <- randomForest(Species~., data = train_rf, ntree=500, mtry=2)
```

Προβλέπουμε το είδος του λουλουδίου των δεδομένων δοκιμής χρησιμοποιόντας το μοντέλο που δημιουργήσαμε.

```
# Using the model we create to make the predictions
pred_rf <- predict(rf_model,test_rf)
```

Δημιουργούμε τον πίνακα σύγκρισης μεταξύ των πραγματικών ειδών και τον προβλεπόμενων ειδών.

```
# Creating the confusion matrix
cm_rf<-table(pred_rf,test_rf$Species)
cm_rf
```

```
##
## pred_rf      setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0          7         2
##   virginica       0          0         9
```

Η ακρίβεια του μοντέλου μας θα είναι:

```
# Calculating the accuracy
rf_accuracy <- sum(diag(cm_rf)) / sum(cm_rf)
rf_accuracy

## [1] 0.9310345
```

Το μοντέλο μας έχει 93% ακρίβεια. Από τον πίνακα σύγκρισης βλέπουμε ότι ταξινόμησε σωστά όλα τα λουλούδια setosa και virginica ενώ δεν τα πήγε τόσο καλά με το versicolor όπου έκανε δύο λάθη ταξινομώντας τα σαν virginica. Παρατηρούμε ότι το μοντέλο του τυχαίου δάσους μας έδωσε ακριβώς τα ίδια αποτελέσματα με αυτό του δένδρου απόφασης, πράγμα περίεργο. Το πόσο σωστά και αξιόπιστα είναι αυτά τα νούμερα θα το δούμε παρακάτω όπου θα δοκιμάσουμε τα ίδια μοντέλα σε διάφορες παραλλαγές διαχωρισμού της βάσης δεδομένων.

5.3 Σύνοψη Κεφαλαίου - Συγκριτικό Αλγορίθμων

Σε αυτό το κεφάλαιο είδαμε κάποιους από τους πιό διαδεδομένους αλγόριθμους που χρησιμοποιούνται σε καθημερινή βάση. Ξεκινήσαμε με τον

αλγόριθμο του δένδρου απόφασης, έναν αλγόριθμο αρκετά απλό και εύκολα κατανοητό και αναφέραμε τα πλεονεκτήματα και μειονεκτήματα του. Το μεγαλύτερο μειονέκτημα του ήταν η υπερεκπαίδευση. Έτσι εισαγάγαμε τις τεχνικές boosting και bagging.

Συνδιάζοντας την μέθοδο bagging με τα δένδρα απόφασης δημιουργήσαμε ένα πανίσχυρο μοντέλο, αυτό του τυχαίου δάσους στο οποίο μπορούσαμε να χρησιμοποιήσουμε τον αλγόριθμο των δένδρων απόφασης χώρις να έχουμε την ανησυχία ότι το μοντέλο μας θα υπερεκπαιδευτεί. Το μοντέλο των τυχαίων δασών μπορεί να χρησιμοποιηθεί τόσο σε προβλήματα ταξινόμησης όσο και σε προβλήματα παλινδρόμησης.

Στο κεφάλαιο αυτό εφαρμόσαμε τους αλγόριθμους στην ίδια βάση δεδομένων. Παρόλο που παρουσιάσαμε το Τυχαίο δάσος σαν μια βελτιωμένη έκδοση των δένδρων απόφασεων, αυτό δε το είδαμε στα αποτελέσματα που πήραμε εφαρμόζοντας τα στην βάση δεδομένων iris.

Αλγόριθμος	Ακρίβεια
Δένδρο απόφασης	0.9310
Τυχαίο Δάσος	0.9310

Πίνακας 8: Σύγκριση αλγορίθμων δένδρου απόφασης και τυχαίου δάσους

Τώρα θα τρέξουμε τους ίδιους αλγόριθμους σε 100 διαφορετικούς διαχωρισμούς της ίδιας βάσης δεδομένων έτσι ώστε να βγάλουμε ασφαλέστερα συμπεράσματα για την επίδοση και ακρίβεια των μοντέλων μας. Έτσι θα δούμε πίο πραγματικά μας δίνει μεγαλύτερη ακρίβεια, μιας και το συγκεκριμένο πεί-

ραμα που δοκιμάσαμε παραπάνω μας έδωσε ακριβώς τα ίδια αποτελέσματα.

```
# loading the dataset and libraries needed
library(caret)
data("iris")
set.seed(1928)

# We split the dataset to 5 folds and we repeat it 20 times
# so we gain the 100 different folds
cvfolds3 <- createMultiFolds(iris$Species ,k=5,times=20)

# Creating the train control using repeatedcv as method to repeat it
# 20 times
tcontrol3 <- trainControl(method = "repeatedcv", number = 5,
                          repeats = 20, index=cvfolds3)

# Creating the decision tree model on 100 folds using the train control
# we create for the 100 folds
dt2 <- train(Species ~ ., data = iris, method = "rpart",
             trControl = tcontrol3)

# Creating the random forest model on 100 folds using the train control
rf2 <- train(Species ~ ., data = iris, method = "rf", ntree = 100,
             trControl = tcontrol3)
```

```
t.test(dt2$resample$Accuracy)$estimate # Decision tree mean accuracy

## mean of x
```

```
## 0.9376667

t.test(dt2$resample$Accuracy)$conf.int # Decision tree 95% CI

## [1] 0.9307806 0.9445527
## attr(,"conf.level")
## [1] 0.95

t.test(rf2$resample$Accuracy)$estimate # Random forest mean accuracy

## mean of x
## 0.9546667

t.test(rf2$resample$Accuracy)$conf.int # Random forest 95% CI

## [1] 0.9483241 0.9610093
## attr(,"conf.level")
## [1] 0.95
```

Συγκεντρώνοντας όλα τα στοχεία στον παρακάτω πίνακα θα έχουμε

Αλγόριθμος	Mean	min	max	sd	95% CI
Δένδρο Απόφασης	0.9376	0.8667	1	0.0347	0.9307-0.9445
Τυχαίο Δάσος	0.9543	0.8667	1	0.033	0.9477-0.9608

Πίνακας 9: Σύγκριση ακρίβειας Δένδου απόφασης και τυχαίου δάσους μετά απο 100 επαναλήψεις

Μετά από εκατό επαναλήψεις σε διάφορους διαχωρισμούς τον δεδομένων μας, παρατηρούμε ότι το ποσοστό ακρίβειας παραμένει σταθερά υψηλό, με το αλγόριθμο τυχαίου δάσους πλέον να μας δίνει καλύτερα αποτελέσματα. Ακόμα βλέπουμε ότι σε συγκεκριμένους διαμοιρασμούς και οι δύο αλγόριθ-

μοι μας δίναν 100% ακρίβεια. Γι’αυτό είναι αναγκαίο αν θέλουμε να έχουμε ακριβέστερα αποτελέσματα να μην τρέχουμε μόνο ένα πείραμα γιατί μπορεί να μας οδηγήσει σε λάθος συμπεράσματα σχετικά με την αποδοτικότητα του μοντέλου μας. Βλέπουμε ότι το αρχικό πείραμα μας έδωσε χειρότερα αποτελέσματα και για τους δύο αλγόριθμους ενώ δεν μας βοήθησε να καταλάβουμε τα βελτιωμένα αποτελέσματα του Τυχαίου Δάσους.

6 Μη επιβλεπόμενη μάθηση

Τα προηγούμενα κεφάλαια αφορούσαν την πρόβλεψη τιμών για μια ή περισσότερες τιμές εξόδου $Y = (Y_1, Y_2, ..., Y_n)$ για δοσμένες τιμές εισόδου $X^T = (X_1, X_2, ..., X_p)$. Συμβολίζοντας με $x_i^T = (x_{i1}, .., x_{ip})$ τις τιμές εισόδου για την i-οστη περίπτωση εκπαίδευσης και y_i να είναι το αποτέλεσμα της μέτρησης. Οι προβλέψεις βασίζονται στο δείγμα εκπαίδευσης $(x_1, y_1), ..., (x_N, y_N)$ των προηγούμενων περιπτώσεων όπου οι τιμές των μεταβλητών είναι γνωστές. Συνήθως σκοπός μας ήταν με χρήση κάποιας συνάρτησης σφάλματος να μειώσουμε το σφάλμα ανάμεσα στην πρόβλεψη $\hat{y}_i$και την πραγματική τιμή y_i, για παράδειγμα $L(y, \hat{y}) = (y - \hat{y})^2$.

Σε αυτό το κεφάλαιο θα εκικεντρωθούμε στην μη επιβλεπόμενη μάθηση, ένα σύνολο στατιστικών εργαλείων που προορίζονται γιατι τη περίπτωση που έχουμε ένα σύνολο με χαρακτηριστικές τιμές $X_1, X_2, ..., X_p$ για n παρατηρήσεις. Δεν μας ενδιαφέρει να κάνουμε κάποια πρόβλεψη, επειδή δεν έχουμε κάποια μεταβλητή απόκρισης Y που να σχετίζεται με τις χαρακτηριστικές τιμές. Αντίθετα, ο στόχος μας είναι να ανακαλύψουμε ενδιαφέροντα πράγματα για τις τιμές $X_1, X_2, ..., X_p$. Μερικές από τις σημαντικότερες μεθόδους στην μη επιβλεπόμενη μάθηση είναι η συσταδοποίηση (Cluster Analysis), οι κανόνες συσχέτισης (Association Rules) και η ανάλυση κύριου στοιχείου (Princial Components Analysis-PCA).

6.1 Αλγόριθμος K-μέσων

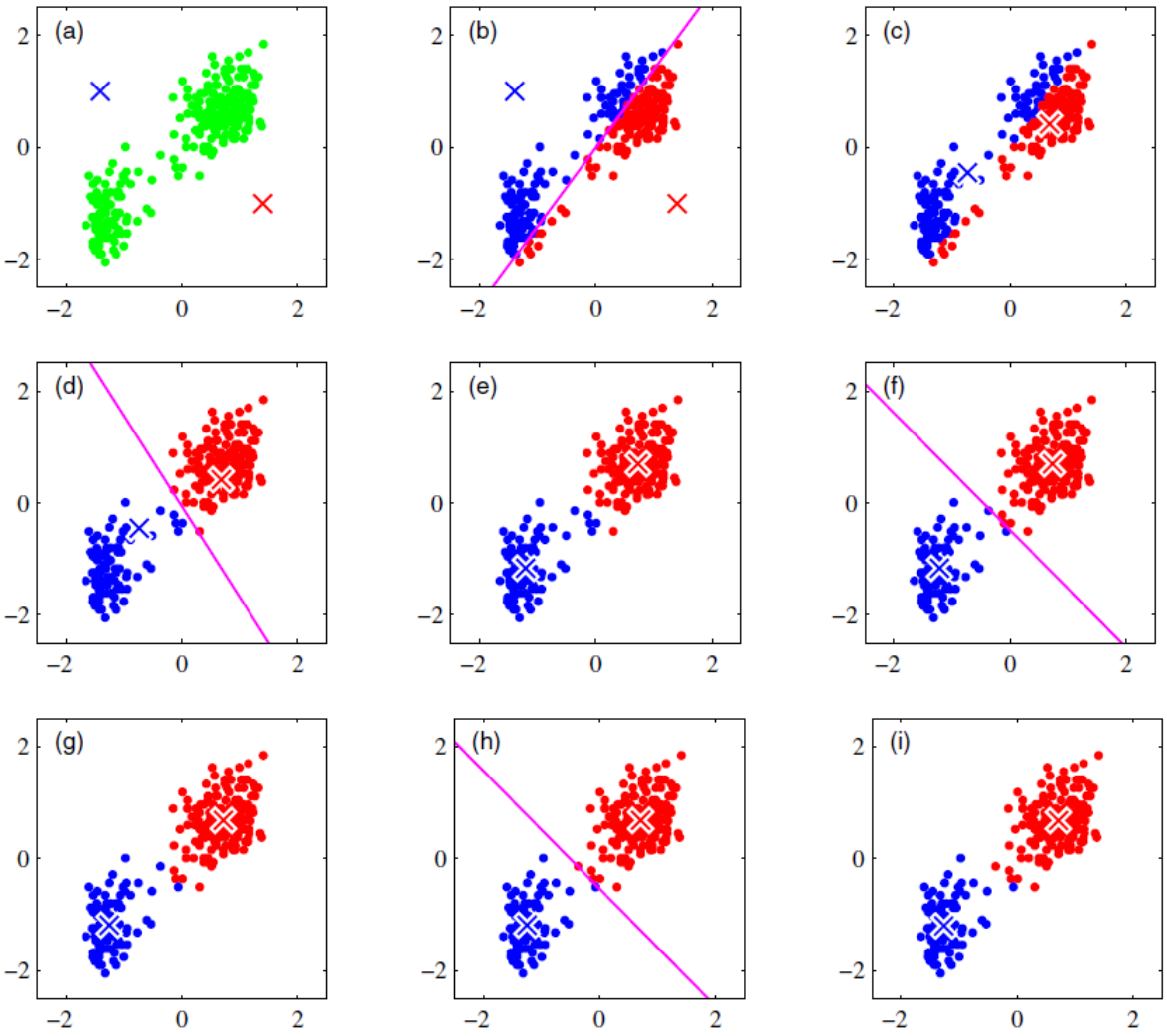
Σε προβλήματα συσταδοποίησης έχουμε το σύνολο δεδομένων, χωρίς τις αντίστοιχες κλάσεις (labels) και χρειάζεται να δημιουργήσουμε ένα μοντέλο, το οποίο θα ομαδοποιεί αυτόματα τα δεδομένα σε συστάδες. Οι συστάδες που θα δημιουργηθούν θέλουμε να χωρίζουν ορθά τα δεδομένα. Έτσι πρακτικά θα

έχουμε συστάδες που θα απαρτίζονται από αντικείμενα, τα οποία θα είναι πιο κοντά στα άλλα αντικείμενα της ίδιας συστάδας σε σχέση με τα αντικείμενα διαφορετικής συστάδας. Οι δύο διασημότεροι αλγόριθμοι συσταδοποίησης είναι η συσταδοποίηση k-μέσων (k-means) και οι ιεραρχικοί αλγόριθμοι συσταδοποίσης (Hierarchical).

Ξεκινάμε εξετάζοντας το πρόβλημα αναγνώρισης ομάδων ή συμπλεγμάτων των δεδομένων σε έναν πολυδιάστατο χώρο. Ας υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων $\{x_1, x_2, .., x_N\}$ που αποτελείται από N παρατηρήσεις μιας τυχαίας D διαστάσεων μεταβλητής x. Στόχος μας είναι να χωρίσουμε το σύνολο των δεδομένων σε ορισμένο αριθμό K ομάδων, όπου στην δεδομένη στιγμή ας υποθέσουμε ότι δίνεται η τιμή του K. Ουσιαστικά θα πρέπει το δούμε ως ένα σύμπλεγμα που περιλαμβάνει μια ομάδα σημειακών δεδομένων των οποίων οι μεταξύ τους αποστάσεις είναι μικρές σε σύγκριση με τις αποστάσεις σε σημεία έξω από το σύμπλεγμα.

6.1.1 Περιγραφή αλγορίθμου K-μέσων

Στον αλγόριθμο K-μέσων ξεκινάμε με K τυχαία σημεία, τα οποία τα ονομάζουμε κεντροειδή της συστάδας και μας δίνουν το κέντρο βάρους της συστάδας. Ο αριθμός K δηλώνει των αριθμό των συστάδων που θέλουμε το μοντέλο μας να δημιουργήσει. Ο αλγόριθμος εκτελεί επαναληπτικά δύο βήματα. Στο πρώτο βήμα γίνεται η ανάθεση των δεδομένων μας σε κάποια συστάδα, ενώ στο δεύτερο βήμα επαναπροσδιορίζεται και μετατοπίζεται το κεντροειδές της κάθε συστάδας.



Σχήμα 6.1: Παράδειγμα διαδικασίας αλγορίθμου K-μέσων για K=2 [23]

Συγκεκριμένα, στο πρώτο βήμα ο αλγόριθμος εξετάζει κάθε δείγμα σε σχέση με τα κεντροειδή των συστάδων και χρήσιμοποιώντας κάποιο μέτρο απόστασης αναθέτει το δείγμα στην συστάδα με το κοντινότερο κεντροειδές. Στο δεύτερο βήμα υπολογίζοντας τον μέσο όρο των δειγμάτων της εκάστοτε συστάδας, επανυπολογίζουμε τα κεντροειδή της κάθε συστάδας έτσι ώστε το κεντροειδές να αντιπροσωπεύει κατάλληλα την νέα συστάδα. Ο αλγόριθμος

ακολουθεί την ίδια επαναληπτική διαδικασία εκτελώντας τα δύο αυτα βήματα έως ότου η μετατόπιση των συστάδων να είναι ελάχιστη και να μην ξεπερνάει κάποια συγκεκριμένη τιμή κατωφλίου.

6.1.2 Μαθηματική παρουσίαση αλγορίθμου K-μέσων

Έστω $C_1, C_2, ..., C_k$ υποδηλώνουν σύνολα που περιέχουν το συνόλο των παρατηρήσεων σε κάθε σύμπλεγμα. Αυτά τα σύνολα ικανοποιούν δύο ιδιότητες.

1. $C_1 \cup C_2 \cup ... \cup C_k = \{x_1, x_2, ..., x_N\}$. Με άλλα λόγια, κάθε παρατήρηση ανήκει σε κάποια από τις συστάδες K.

2. $C_k \cap C_m = \emptyset$ για όλα τα $k \neq m$. Με άλλα λόγια, οι συστάδες είναι ανεξάρτητες, καμία παρατήρηση δεν ανήκει σε περισσότερες από μια συστάδες.

Ορίζουμε σαν απόσταση ανάμεσα σε δύο σημεία το τετράγωνο της ευκλείδιας απόστασης:

$$d(x_i, x_{i'}) = \sum_{j=1}^{d} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$$

Είναι βολικό σε αυτό το σημείο να ορίσουμε κάποιους συμβολισμούς για να μπορέσουμε να περιγράψουμε τον αλγόριθμο. Για κάθε σημείο x_n, εισάγουμε ένα αντίστοιχο σετ δυαδικών μεταβλητών $z_{nk} \in \{0, 1\}$, όπου $k = 1, .., K$ που περιγράφει από ποιά συστάδα K παίρνουμε το εκάστοτε σημείο x_n, έτσι ώστε αν το σημείο x_n ανήκει στην συστάδα K τότε $z_{nk} = 1$ και $z_{nj} = 0$ για $j \neq k$.

Ορίζουμε την συνάρτηση που συνήθως αναφέρεται ως μέτρο παραμόρφωση η οποία υπολογίζει το άθροισμα των αποστάσεων κάθε σημείου της συστάδας από το κεντροειδή μ_k.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} d(x_n - \mu_k) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} ||x_n - \mu_k||^2$$

Σκοπός μας είναι να βρούμε τα κατάλληλα z_{nk} και μ_k έτσι ώστε να ελαχιστοποιήσουμε το J. Μπορούμε να το κάνουμε αυτό μέσω μιας επαναληπτικής διαδικασίας στην οποία κάθε επανάληψη περιλαμβάνει δύο διαδοχικά βήματα που αντιστοιχούν σε διαδοχικές βελτιστοποιήσεις των z_{nk} και μ_k. Αρχικά επιλέγουμε κάποιες αρχικές τιμές για το μ_k. Στην συνέχεια, στην πρώτη φάση ελαχιστοποιούμε το J σε σχέση με το z_{nk} διατηρώντας το μ_k σταθερό. Στην δεύτερη φάση ελαχιστοποιούμε το J σε σχέση με το μ_k διατηρώντας το z_{nk} σταθερό. Αυτά τα δύο στάδια βελτιστοποίησης επαναλαμβάνονται μέχρι να υπάρχει σύγκλιση.

Ας δούμε πρώτα τον προσδιορισμό του z_{nk}. Επειδή το J είναι γραμμικό ως προς το z_{nk}, η βελτιστοποίηση μπορεί να πραγματοποιηθεί εύκολα και να δώσει μια λύση κλειστής μορφής. Οι όροι που αφορούν διαφορετικά n είναι ανεξάρτητοι και έτσι μπορούμε να βελτιστοποιήσουμε για καθέ ένα n ξεχωριστά επιλέγοντας z_{nk} να είναι 1 για οποιαδήποτε τιμή του k η οποία δίνει την ελάχιστη τιμή του $||x_n - \mu_k||^2$. Πιο τυπικά, αυτό μπορεί να εκφραστεί ως:

$$z_{nk} = \begin{cases} 1 & \alpha\nu\ k = argmin_j ||x_n - \mu_j||^2 \\ 0 & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Για την βελτιστοποίηση του μ_k με το z_{nk} θα έχουμε. Η συνάρτηση J είναι δευτέρου βαθμού ως προς το μ_k και μπορεί να ελαχιστοποιηθεί μηδενίζοντας την πρώτη παράγωγο και θα έχουμε

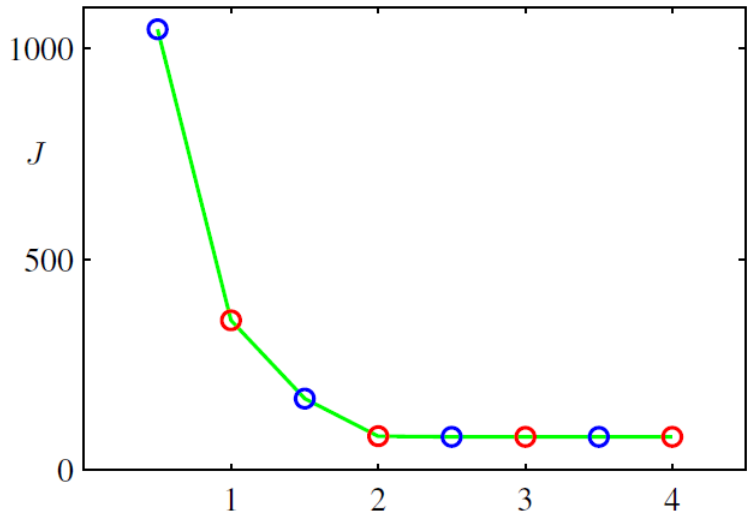
$$2 \sum_{n=1}^{N} z_{nk}(x_n - \mu_k) = 0$$

και έτσι θα έχουμε για το μ_κ

$$\mu_k = \frac{\sum_n z_{nk} x_n}{\sum_n z_{nk}}$$

6.1.3 Επιλογή του Αριθμού K

Ένα σημαντικό μειονεκτήμα του αλγορίθμου k-μέσων είναι το γεγονός ότι δεν υπάρχει κάποια αυτοματοποιημένη διαδικασία για την επιλογή του k. Ο αριθμός των συστάδων είναι μια υπερπαράμετρος που ορίζεται από τον χρήστη και η επιλογή του σωστού αριθμού επαφίεται στη δική του γνώση και εμπειρία. Μιας και ο αλγόριθμος των k-μέσων είναι ένας αλγόριθμος μη επιβλεπόμενης μάθησης τα δεδομένα μας δεν έχουν το επιπλέον χαρακτηριστικό κλάσης το οποίο θα μας οδηγήσει στην σωστή επιλογή του k, έτσι απαιτείται περαιτέρω εξερεύνηση και κατανόηση των δεδομένων του προβλήματος, προκειμένου να καταλήξουμε στον σωστό αριθμό συστάδων.



Σχήμα 6.2: Γραφική παράστασης συνάρτησης κόστους J συναρτήσει του αριθμού K [23]

Δυστυχώς, δεν υπάρχει κάποιος γενικός κανόνας ο οποίος να μας βοηθάει σε όλα τα προβλήματα να επιλέξουμε το κατάλληλο K. Μια συνήθης τεχνική η οποία μπορεί να φανεί χρήσιμη σε ορισμένα προβλήματα, είναι «ο κανόνας του αγκώνα» (the elbow rule). Ουσιαστικά στον κανόνα του αγκώνα κοιτώντας το γράφημα προσπαθούμε να βρούμε το κατάλληλο K στο οποίο σταματάει η μείωση του J να είναι τόσο απότομη δίνοντας μας στο γράφημα μια γωνία όσο το δυνατόν πιό κοντά στην ορθή. Στο παρακάτω γράφημα ο κανόνας του αγκώνα υποδυκνύει ότι η κατάλληλη επιλογή για το πρόβλημα μας θα ήταν για $k = 2$. Ωστόσο, σε πολλά προβλήματα η γραφική είναι πιο ομαλή και δεν μπορεί να εφαρμοστεί ο κανόνας του αγκώνα, με αποτέλεσμα η επιλογή του K να μην είναι ξεκάθαρη.

6.1.4 Υλοποίηση k-Μέσων στην R

Θα χρησιμοποιήσουμε την βάση δεδομένων USArrests . Αυτή η βάση περιέχει στατιστικά στοιχεία σε συλλήψεις ανά 100.000 κατοίκους για κάθε επίθεση, δολοφονία και βιασμό σε κάθε μία από τις 50 πολιτείες των ΗΠΑ το 1973. Δίνεται επίσης το ποσοστό του πληθυσμού που ζει σε αστικές περιοχές.

Φορτώνουμε την βάση δεδομένων μας καθώς και την βιβλιοθήκη για τα γραφήματα.

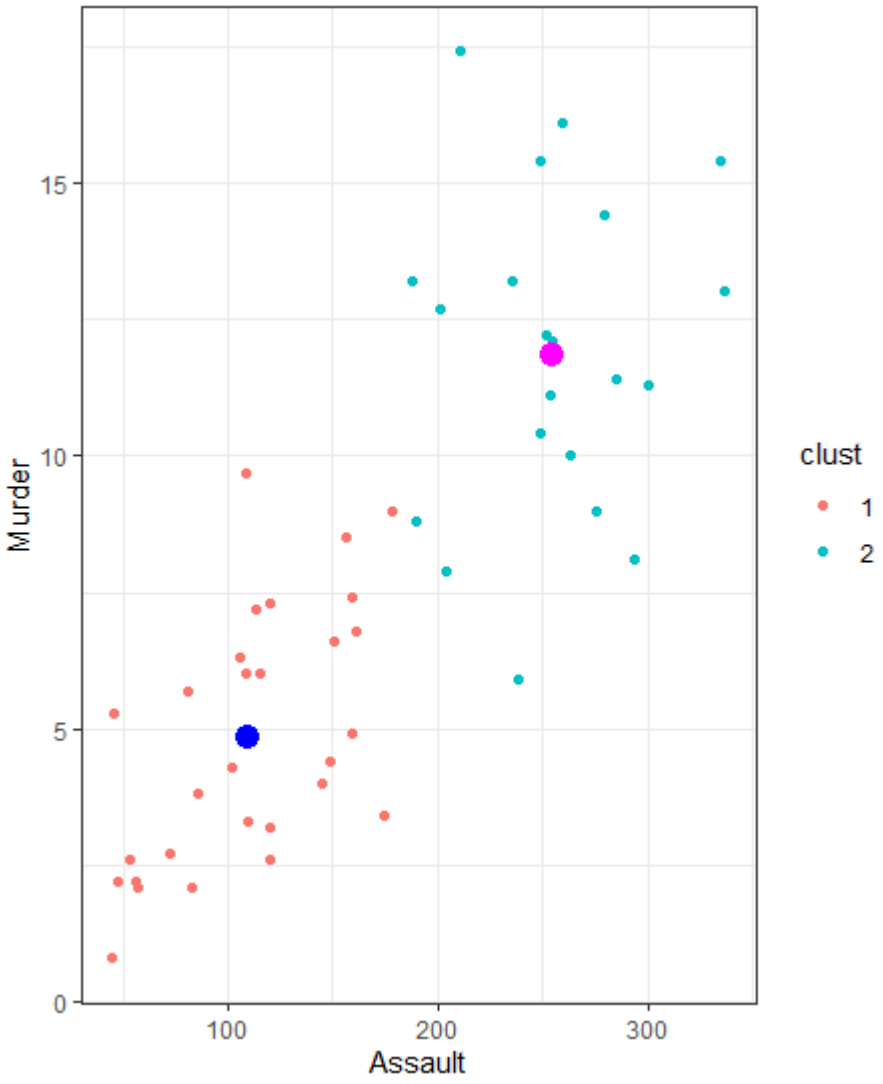
```
# loading the dataset and libraries needed
library(ggplot2)
library(factoextra)
data("USArrests")
```

Τρέχουμε τον έτοιμο αλγόριθμο της R kmeans και επιλέγουμε το k=2, δηλαδή να δημιουργηθούν δύο συστάδες.

```
# Creating the kmeans model choosing 2 centers
k2 <- kmeans(USArrests, centers = 2, nstart = 25)
```

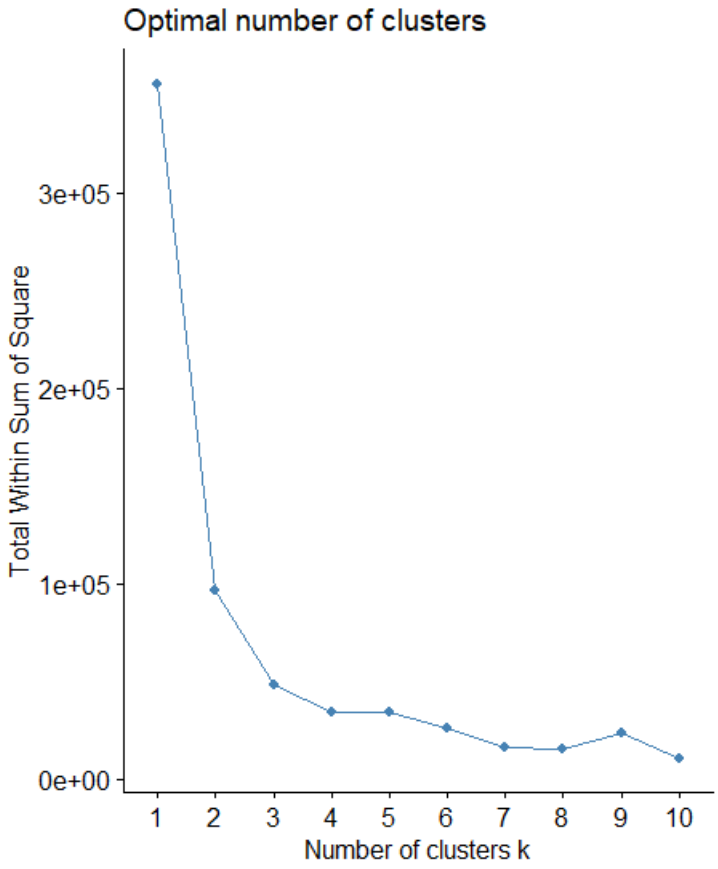
Επειδή η βάση δεδομένων μας έχει 4 χαρακτηριστικά δεν μπορούμε να την παραστήσουμε γραφικά. Γι'αυτό επιλέγουμε να παραστήσουμε γραφικά τις επιθέσεις και τις δολοφονίες. Χρωματίζουμε με κόκκινο και γαλάζιο τα σημεία ανάλογα σε ποιά συστάδα ανήκουν ενώ με την μπλε και ροζ βούλα δείχνουμε ποιά είναι τα κέντρα των δύο συστάδων μας

```
# Changing the clusters to factor so we can use them as color
# to our visualization
clust <- as.factor(k2$cluster)

# Creating a data frame with the coordinates of cluster's centers.
centers <- as.data.frame(k2$centers[,1:2])

# Plotting the Assault and Murder features to visualize the clusters
# and the centers
ggplot(USArrests,aes(x=Assault,y=Murder,colour=clust)) + geom_point() +
theme_bw() +
geom_point(aes(centers[1,2],centers[1,1]), colour='Blue', size=4) +
geom_point(aes(centers[2,2],centers[2,1]), colour='Magenta', size=4)
```



Τέλος θα παραστήσουμε γραφικά τον κανόνα του αγκώνα

```
# Creating the graph for elbow rule
fviz_nbclust(USArrests, kmeans, method = "wss")
```



Παρατηρούμε από τον κανόνα του αγκώνα ότι το $k = 3$ θα ήταν μια πολύ καλή επιλογή για το μοντέλο μας, ενώ για $k > 3$ η μείωση του σφάλματος δεν είναι αρκετά μεγάλη.

6.2 Ιεραρχικοί Αλγόριθμοι Συσταδοποίησης

Ένα σημαντικό μειονέκτημα της συσταδοποίησης K-μέσων είναι ότι απαιτεί από τον χρήση να προκαθορίσει τον αριθμό K των συστάδων. Η ιεραρχική συσταδοποίηση είναι μια εναλλακτική προσέγγιση που δεν απαιτεί να δεσμευόμαστε σε μία συγκεκριμένη επιλογή K.

Οι ιεραρχικοί αλγόριθμοι συσταδοποίσης παράγουν μια ιεραρχία εμφωλιασμένων συσταδοποιήσεων. Δηλαδή, μπορεί να υπάρχουν συστάδες που να περιέχουν μεμονωμένα στοιχεία και άλλες συστάδες που μπορεί να περιέχουν μικρότερες συστάδες, δημιουργώντας έτσι τα διάφορα επίπεδα της ιεραρχίας.

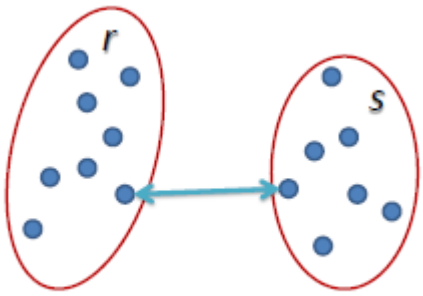
Οι ιεραρχικοί αλγόριθμοι χωρίζονται σε δύο υποκατηγορίες: τους συσσωρευτικούς (agglomerative) και τους διαιρετικούς (divisive) . Η εξέλιξη του αλγορίθμου μπορεί να αναπαρασταθεί γραφικά με ένα δενδρόγραμμα ανομοιότητας. Το δενδρόγραμμα περιέχει $n-1$ επίπεδα με το κάθε επίπεδο να αποτελεί ένα βήμα του αλγορίθμου. Επιπλέον περιλαμβάνει και την ποσότητα ανομοιότητας μεταξύ των ομάδων που συγχωνεύονται. Το βασικό πλεονέκτημα των ιεραρχικών αλγορίθμων σε σχέση με τον αλγόριθμο των k-μέσων είναι ότι δεν χρειάζεται να δηλώσουμε εξαρχής τον αριθμό των συστάδων, αφου οποιοσδήποτε αριθμός συστήδων μπορεί να επιτευχθεί κόβοντας το δενδρόγραμμα στο κατάλληλο επίπεδο.

6.2.1 Ορισμός Απόστασης Συστάδων

Πριν ξεκινήσουμε όμως την ανάλυση των συσσωρευτικών ιεραρχικών αλγορίθμων, θα πρέπει αρχικά να ορίσουμε τον τρόπο υπολογισμού της απόστασης μεταξύ δύο συστάδων.

Στην ιεραρχική συσταδοποίηση με βάση το κριτήριο απλού συνδέσμου (Simple Linkage), η απόσταση μεταξύ δύο συστάδων ορίζεται ως η μικρότερη απόσταση μεταξύ δύο σημείων σε κάθε σύμπλεγμα. Για παράδειγμα, η απόσταση μεταξύ των συστάδων "r" και "s" προς τα αριστερά είναι ίση με το μήκος του βέλους μεταξύ των δύο πιο κοντινών σημείων.

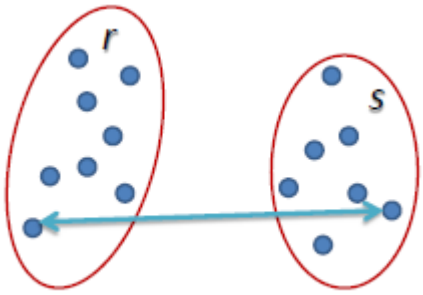
$$L(r,s) = min(D(x_{ri}, x_{sj})$$



Σχήμα 6.3: Απλός σύνδεσμος [18]

Στο κριτήριο πλήρους συνδέσμου (Complete Linkage) , η απόσταση μεταξύ δύο συστάδων ορίζεται ως η μεγαλύτερη απόσταση μεταξύ δύο σημείων σε κάθε σύμπλεγμα. Για παράδειγμα, η απόσταση μεταξύ των συστάδων "r" και "s" προς τα αριστερά είναι ίση με το μήκος του βέλους μεταξύ των δύο πλέον απομακρυσμένων σημείων.

$$L(r,s) = max(D(x_{ri}, x_{sj}))$$

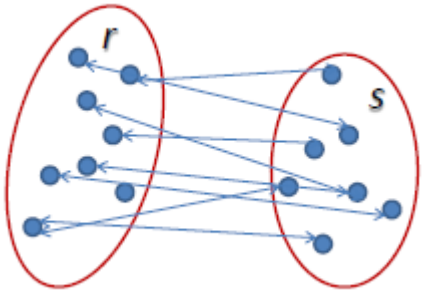


Σχήμα 6.4: Πλήρης σύνδεσμος [18]

Ο μέσος όρος συστάδων (Average Linkage) είναι ουσιαστικά η μέση τιμή των αποστάσεων μεταξύ κάθε πιθανού ζεύγους μεταξύ των σημείων των δύο συστάδων. Βρίσκεται κάπου ανάμεσα στην ελάχιστη και τη μέγιστη απόσταση.

Έχει μικρότερη ευαισθησία σε θόρυβο και σε ακραίες τιμές (outliers), αλλά ευνοεί τις συστάδες με κυκλικό σχήμα.

$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$



Σχήμα 6.5: Σύνδεσμος μέσου όρου [18]

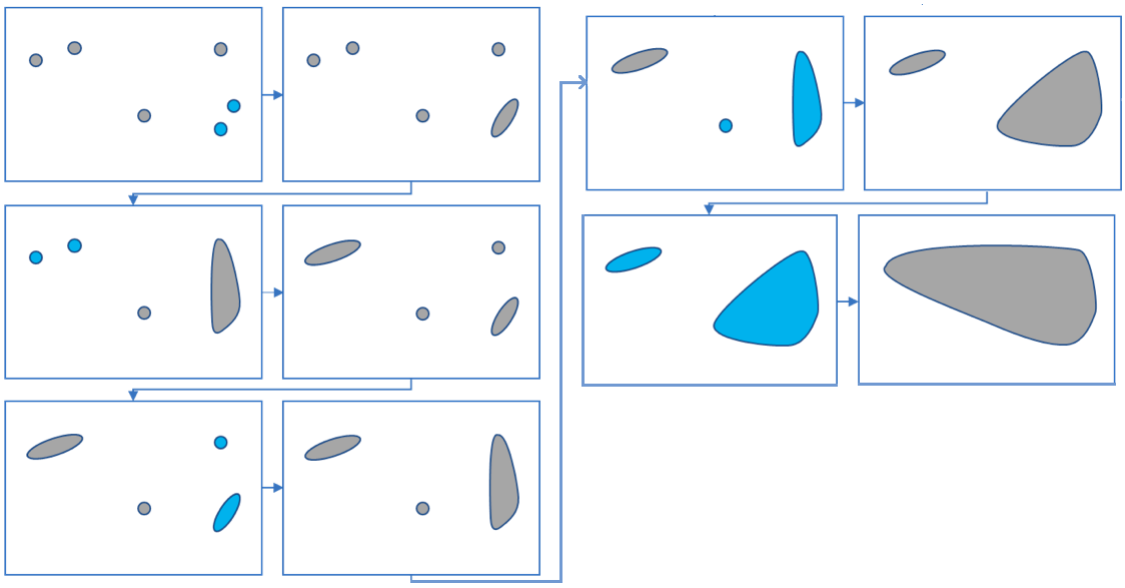
Τέλος, για τη μέθοδο του Ward η απόσταση μεταξύ δύο συστάδων, r και s είναι ίση με το πόσο θα αυξηθεί το άθροισμα των τετραγώνων της απόστασης των στοιχείων της κάθε συστάδας από το αντίστοιχο κεντροειδές της κάθε συστάδας μετά τη συγχώνευση τους, rs, δηλαδή:

$$L(r,s) = \sum_{x \in r}(x - c_r)^2 + \sum_{x \in s}(x - c_s)^2 - \sum_{x \in rs}(x - c_{ij})^2$$

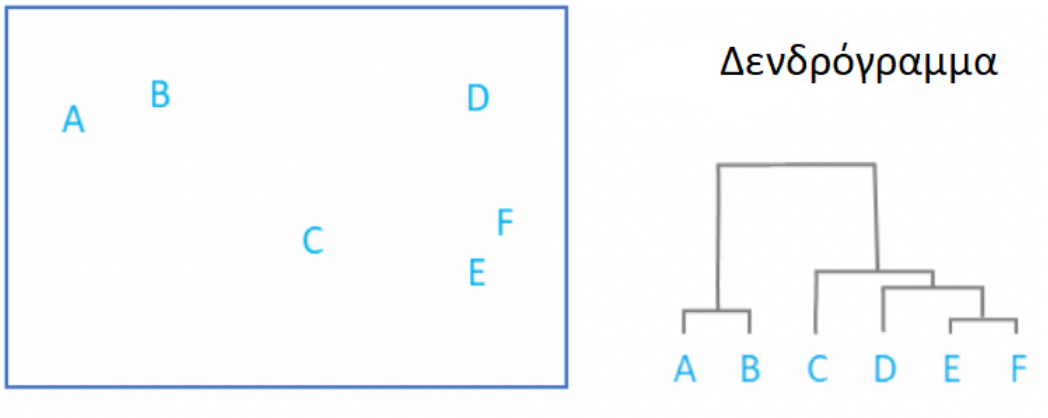
όπου c_r είναι το κεντροειδές της συστάδας r, c_s το κεντροειδές της συστάδας s και c_{rs} το κεντροειδές της συστάδας rs που προκύπτει από τη συγχώνευσή τους. Πρόκειται για το ιεραρχικό ανάλογο του αλγόριθμου k-μέσων.

6.2.2 Συσσωρευτικοί Αλγόριθμοι

Οι συσσωρευτικοί (από κάτω προς τα πάνω) αλγόριθμοι (Agglomerative algorithms) ξεκινάνε θεωρώντας αρχικά κάθε ένα από τα n σημεία του δείγματος σαν μια ξεχωριστή συστάδα δημιουργόντας n συστάδες. Σε κάθε επόμενο βήμα οι δύο κοντινότερες συστάδες συγχωνεύονται έχοντας πλέον μια συστάδα λιγότερη από αυτές που είχαμε πριν την αρχή του κάθε βήματος. Αυτή η διαδικασία επαναλαμβάνεται έως ότου όλα τα σημεία να καταλήξουν σε μία και μοναδική συστάδα. Η όλη διαδικασία του αλγορίθμου μπορεί να αναπαρασταθεί με δενδρόγραμμα.



Σχήμα 6.6: Παράδειγμα συσσωρευτικού αλγόριθμου βήμα-βήμα [20]



Σχήμα 6.7: Παράδειγμα δενδρογράμματος [20]

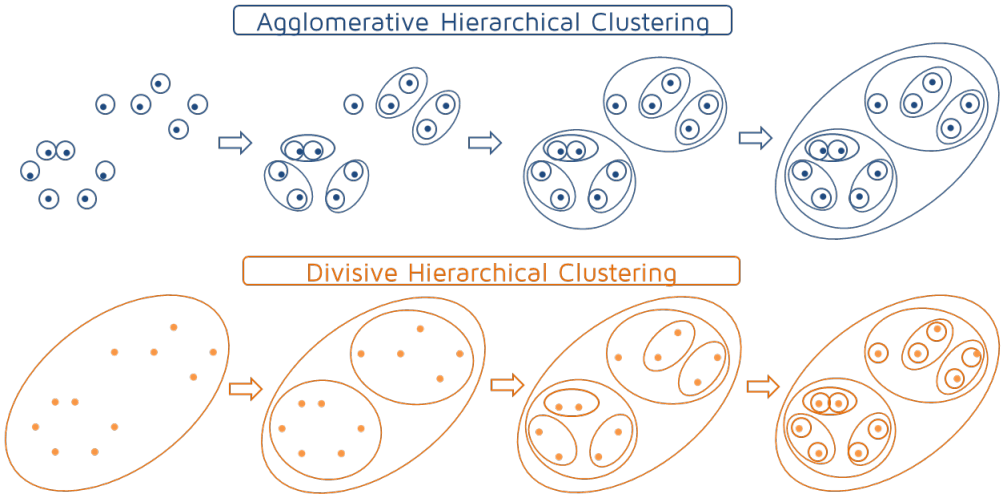
Στα δενδρογράμματα, οι αρχικές ομάδες (αντικείμενα) βρίσκονται στα φύλλα στην βάση του δενδρογράμματος, και κάθε φορά που συγχωνεύονται δύο ομάδες τους εντάσουμε σε ένα δένδρο. Το ύψος των κλαδιών αντιπροσωπεύει την ανομοιογένεια μεταξύ των ομάδων που ενώθηκαν. Η ρίζα του δένδρου που βρίσκεται στην κορυφή αντιπροσωπεύει μια ομάδα που περιέχει όλα τα δεδομένα. Εάν κόψουμε το δέντρο σε οποιοδήποτε δεδομένο ύψος, προκαλούμε μια ομαδοποίηση ενός δεδομένου μεγέθους.

Είναι δύσκολο να επιλέξουμε τον "σωστό" αριθμό συστάδων και το σημείο στο οποίο θα κόψουμε το δενδρο, μιας και ο ιεραρχικός αλγόριθμος θα δημιουργεί πάντα μια ιεραρχία ακόμα και αν τα δεδομένα είναι εντελώς τυχαία. Αλλά όπως και με την επιλογή του K στον αλγόριθμο των K-μέσων ελπίζουμε ότι θα υπάρχει ένα ορατό "κενό" στα μήκη των συνδέσμων στο δενδροδιάγραμμα (που αντιπροσωπεύουν την ανομοιογένεια μεταξύ συγχωνευμένων ομάδων). Μια σύνηθης τακτική για το που θα κοπεί το δενδροδιάγραμμα είναι κοιτόντας το να βρούμε την μεγαλύτερη ανομοιογένεια (ύψος του κλάδου) που υπάρχει ανάμμεσα σε οποιαδήποτε νοητικη προέκταση του κάθε

φύλλου.

6.2.3 Διαιρετικοί Αλγόριθμοι

Οι διαιρετικοί αλγόριθμοι (Divisive Algorithms) ακολουθούν αντίστροφη διαδικασία από αυτή των συσσωρευτικών. Ξεκινούν από μία ομάδα η οποία εμπεριέχει όλα τα διανύσματα και σε κάθε βήμα μία ομάδα διασπάται σε δύο μέχρι να καταλήξουμε σε n ομάδες. Η πολυπλοκότητα τους είναι μεγαλύτερη από τους συσσωρευτικούς αφού η διάσπαση μίας ομάδας σε δύο μπορεί να γίνει κατά $2^{n-1}-1$ τρόπους και η επιλογή της βέλτιστης διάσπασης πρακτικά είναι αδύνατη ακόμα και για μικρό n. Στην πράξη αυτό που γίνεται ότι ο αλγόριθμος σε κάθε βήμα διασπά μία ομάδα αλλά όχι κατά βέλτιστο τρόπο. Η όλη διαδικασία του αλγορίθμου μπορεί να αναπαρασταθεί, όπως και στους συσσωρευτικούς, με δενδρόγραμμα.



Σχήμα 6.8: Διαφορά συσσωρευτικού με διαιρετικου αλγόριθμου [21]

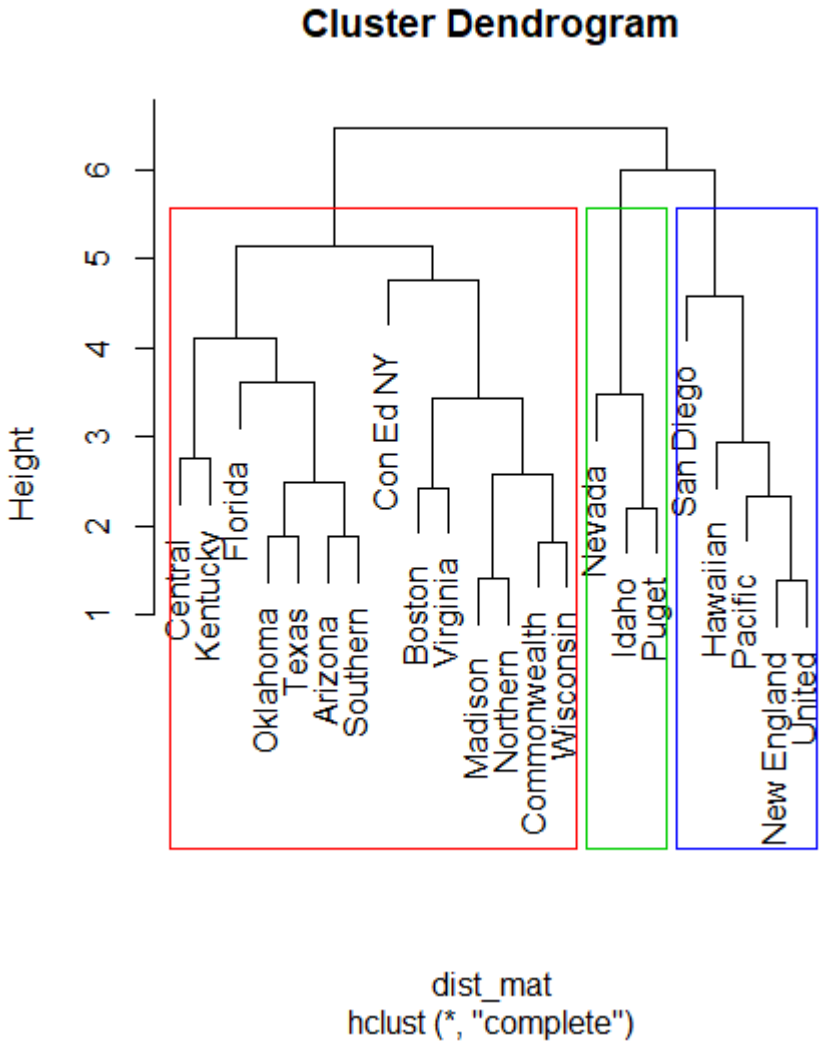
6.2.4 Υλοποίηση Ιεραρχικής Συσταδοποίησης στην R

Θα χρησιμοποιήσουμε μια βάση δεδομένων με 22 εταιρίες από διάφορες πολιτείες των ΗΠΑ και τα έξοδα τους για έναν χρόνο. Επίσης με εξαίρεση την πρώτη στήλη θα αλλάξουμε την κλίμακα των αριθμών έτσι ώστε να είναι όλα κοντά στο μηδέν και να μην επηρεάζει το μοντέλο ένα χαρακτηριστικό περισσότερο απότι τα άλλα

```
# Loading and scaling the data
mydata <- read.csv("C:/Users/tit0v/R/PTUXIAKH/utilities.csv")
scaled <- as.data.frame(scale(mydata[,-1]))
```

Υπολογίζουμε την ευκλείδια απόσταση μεταξύ των σημείων και δημιουργούμε την ιεραρχική συσταδοποίηση χρησιμοποιώντας το κριτήριο πλήρους συνδέσμου και δημιουργούμε τρεις συστάδες.

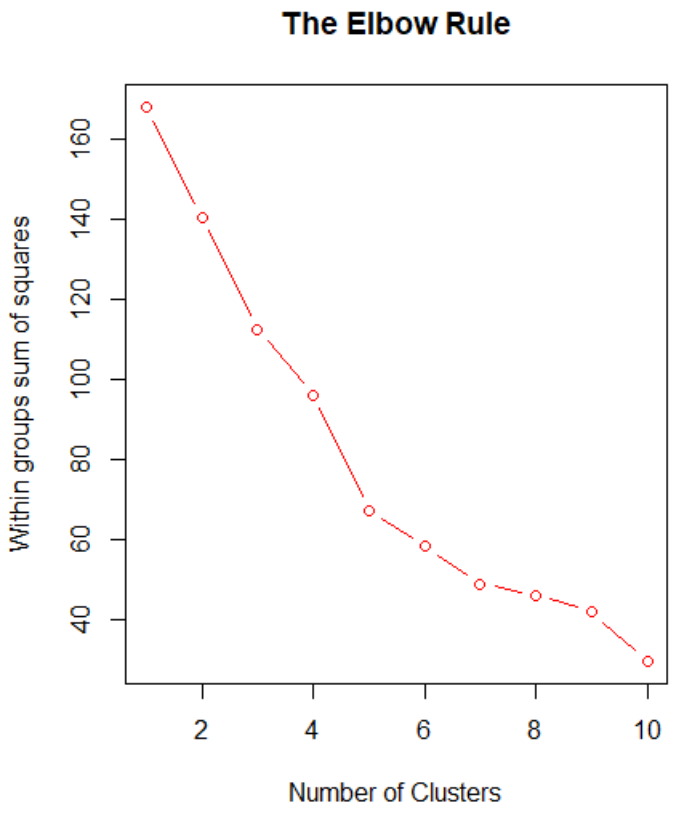
```
# Calculating the distances
dist_mat <- dist(scaled, method = 'euclidean')

# Creating the hierarchical cluster using complete method
hclust_com <- hclust(dist_mat, method = 'complete')

# Plotting the dendogram and creating three clusters
plot(hclust_com,labels=mydata$Company)
rect.hclust(hclust_com , k = 3, border = 2:6)
```



Δημιουργούμε το γράφημα για τον κανόνα του αγκώνα που θα μας βοηθήσει να επιλέξουμε τον αριθμό των συστάδων.

```
# Creating the graph for elbow rule
wss <- (nrow(scaled)-1)*sum(apply(scaled,2,var))
for (i in 2:10) wss[i] <- sum(kmeans(scaled, centers=i)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
```

```
ylab="Within groups sum of squares", col="red",
main="The Elbow Rule")
```



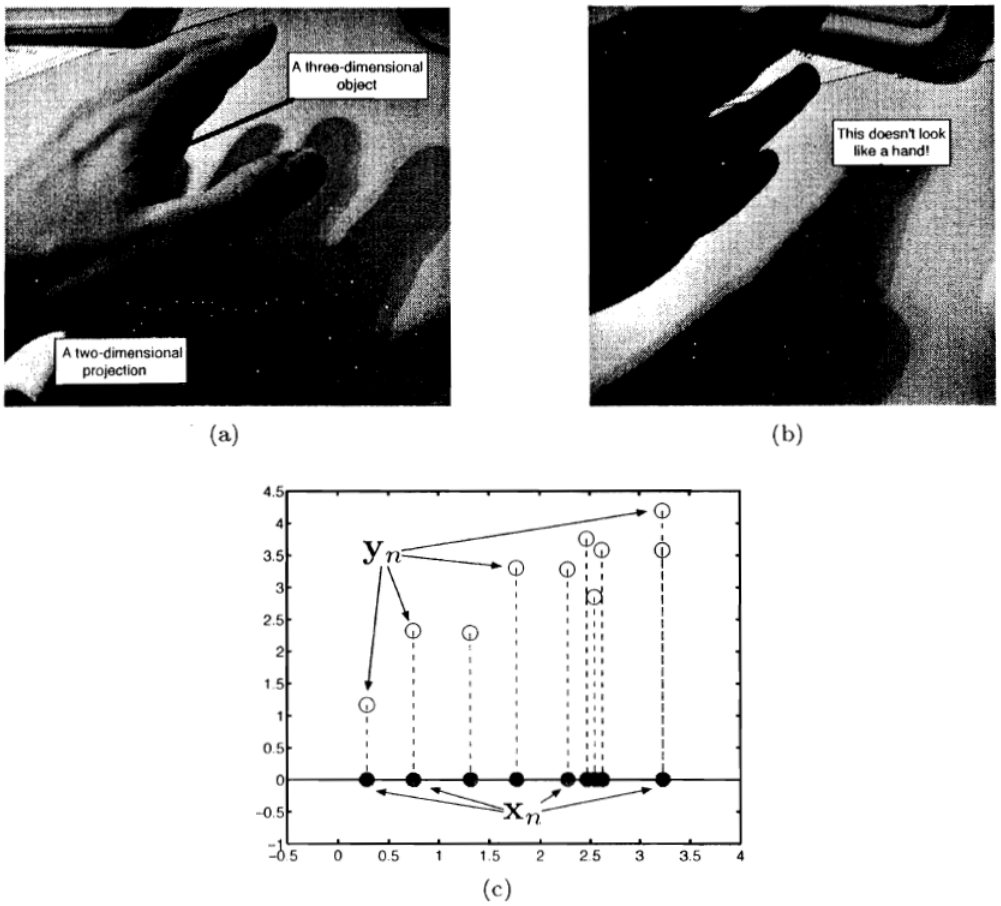
Βλέποντας το γράφημα, αν και δεν είναι αρκετά ξεκάθαρο, οι 5 συστάδες θα ήταν μια καλή επιλογή.

6.3 Ανάλυση Κύριων Συνιστωσών - PCA

6.3.1 Το γενικό πρόβλημα των πολλών διαστάσεων

Η αφετηρία μας είναι ένα σύνολο δεδομένων από N αντικείμενα y_n. Κάθε αντικείμενο είναι ένα M-διάστατο διάνυσμα. Ο αριθμός των παραμέτρων σε

πολλά μοντέλα αυξάνεται όταν έχουμε προβλήματα με πολλές διαστάσεις, έτσι αν το M είναι μεγάλο μπορεί να κάνει την εκτίμηση των παραμέτρων ένα δύσκολο εγχείρημα. Σε αυτές τις περιπτώσεις, είναι συχνά χρήσιμο να μετασχηματίζουμε την M-διάστατη παρουσίαση του y_n σε μία D-διάστατη παρουσίαση x_n. Αυτή η διαδικασία είναι γνωστή ως προβολή. Προβάλουμε ένα M-διάστατο σύνολο δεδομένων σε D-διαστάσεις ελπίζοντας να διατηρηθούν όλες οι χρήσιμες ιδιότητες.



Σχήμα 6.9: Η ιδέα της προβολής. στο (a) και (b) ένα χέρι (τρισδιάστατο) προβάλλεται σε δύο διαστάσεις. (c) Το δισδιάστατο y_n προβάλλεται στο μονοδιάστατο x_n. [7]

Όταν εκτελούμε μια προβολή, σκοπός μας είναι να κρατήσουμε όσες περισσότερες πληροφορίες μπορούμε από τα δεδομένα. Όπως βλέπουμε στην εικόνα 24a και 24b τα δεδομένα (χέρι) που προβάλλουμε είναι ίδια όμως η πληροφορία (σκιά) που έχουμε μετά την προβολή δεν είναι η ίδια και στις δύο εικόνες. Ο τρόπος για να το πετύχουμε αυτό είναι να επιλέξουμε την προβολή με την μεγαλύτερη διακύμανση.

Ας υποθέσουμε ότι επιθυμούμε να παρουσιάσουμε M παρατηρήσεις με μετρήσεις σε ένα σύνολο από N χαρακτηριστικά y_n, ως μέρος μια διερευνητικής ανάλυσης δεδομένων. Θα μπορούσαμε να το κάνουμε αυτό εξετάζοντας δισδιάστατα γραφήματα των δεδομένων, όπου το κάθε ένα θα περίεχει τις μετρήσεις N παρατηρήσεων για 2 χαρακτηριστικά. Ωστόσο θα χρειαστούν $M * (M-1)/2$ από αυτά τα γραφήματα αν θέλουμε να παραστήσουμε όλους τους συνδιασμούς των χαρακτηριστικών. Εάν το M είναι μεγάλο τότε σιγουρά δεν θα είναι δυνατό να τα εξετάσουμε όλα, ενώ επιπλέον καθένα από τα γραφήματα δεν θα είναι αρκετά χρήσιμο γιατί θα περιέχει ένα αρκετά μικρό μέρος του συνόλου των πληροφοριών που υπάρχουν στο σύνολο των δεδομένων. Είναι σαφές ότι απαιτείται μια καλύτερη μέθοδος έτσι ώστε να απεικονίσουμε το πρόβλημα ειδικά όταν το M είναι αρκετά μεγάλο. Συγκεκριμένα, θα θέλαμε να βρεθεί μια χαμηλής διάστασης αναπαράσταση των δεδομένων που να συλλαμβάνει όσο το δυνατό περισσότερες πληροφορίες. Για παράδειγμα αν μπορούμε να αποκτήσουμε μια δισδιάστατη αναπαράσταση των δεδομένων που συλλαμβάνει τις περισσότερες πληροφορίες, τότε θα μπορούσαμε να σχεδιάσουμε τις παρατηρήσεις σε αυτό το χώρο χαμηλής διάστασης.

6.3.2 Ο αλγόριθμος PCA

Ο αλγόριθμος PCA είναι ίσως η ευρέως χρησιμοποιούμενη στατιστική τεχνική για την προβολή δεδομένων σε χώρους χαμηλότερης διάστασης. Είναι αρκετά διάσημη τεχνική στην μηχανική μάθηση κυρίως στον τομέα της απεικόνησης και της επιλογής χαρακτηριστικών. Ο PCA καθορίζει μια γραμμική προβολή, όπου κάθε μία από τις προβαλλόμενες διαστάσεις είναι γραμμικός συνδιασμός των αυθεντικών διαστάσεων. Έτσι αν προβάλλουμε M διαστάσεις σε D, το PCA θα ορίσει D διανύσματα w_d κάθε ένα από τα οποία θα είναι N-διαστάσεων. Το d στοιχείο της προβολής, x_{nd} (όπου $x_n = [x_{n1}, ..., x_{nD}]^T$) υπολογίζεται ως:

$$x_{nd} = w_d^T y_n$$

Ο αλγόριθμος PCA χρησιμοποιεί την διακύμανση στον προβαλλόμενο χώρο ως κριτήριο για την επιλογή του w_d. Συγκεκριμένα, το w_1 θα είναι η προβολή για την οποία η διακύμανση του x_{n1} να είναι όσο το δυνατό μεγαλύτερη. Η δεύτερη προβαλλόμενη διάσταση θα πρέπει επίσης να μεγιστοποιεί την διακύμανση αλλά το w_2 θα πρέπει να είναι ορθογώνιο στο w_1 $(w_1^T x_2 = 0)$. Ομοίως το w_3, θα πρέπει να μεγιστοποιεί την διακύμανση και να είναι ορθογώνιο και στο w_1 και στο w_2, και ούτω καθεξής. Γενικά

$$w_i^T w_j = 0 \; \forall j \neq i$$

Επιπλέον, ο PCA επιβάλλει τον περιορισμό ότι κάθε w_i πρέπει να έχει μήκος 1. Αυτό δεν περιορίζει την τεχνική, αφού αυτό που μας ενδιαφέρει είναι η κατεύθυνση.

$$w_i^T w_i = 1$$

Πριν ξεκινήσουμε είναι αρκετά βολικό να θεωρήσουμε ότι κάθε μια από τις αρχικές διαστάσεις έχουμε μέση τιμή 0.

$$\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n = 0$$

Θα αρχίσουμε βρίσκοντας την προβολή για $D = 1$. Με άλλα λόγια, ενδιαφερόμαστε να βρούμε ένα διάνυσμα w. Σε αυτή την περίπτωση η προβολή μας δίνει σαν αποτέλεσμα το x_n το οποίο δίνεται από τον τύπο:

$$x_n = w^T y_n$$

Η διακύμανση σ_x^2 δίνεται από τον τύπο:

$$\sigma_x^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})^2$$

Μπορούμε να απλοποιήσουμε την έκφραση χρησιμοποιόντας την υπόθεση μας ότι $\bar{y} = 0$.

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} w^T y_n$$

$$= w^T (\frac{1}{N} \sum_{n=1}^{N} y_n)$$

$$= w^T \bar{y} = 0$$

Έτσι η διακύμανση θα γίνει

$$\sigma_x^2 = \frac{1}{N} \sum_{n=1}^{N} x_n^2$$

Χρησιμοποιόντας τον ορισμό για το x_n θα έχουμε

$$\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(w^T y_n)^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} w^T y_n y_n^T w$$

$$= w^T(\frac{1}{N}\sum_{n=1}^{N} y_n y_n^T)w$$

$$\sigma_x^2 = w^T C w$$

όπου C είναι ο πίνακας διακύμανσης που ορίζεται ως:

$$C = \frac{1}{N}\sum_{n=1}^{N}(y_n - \bar{y})(y_n - \bar{y})^T$$

όπου στην δική μας περίπτωση το $\bar{y} = 0$. Παρατηρώντας τον τύπο του C μπορούμε να καταλάβουμε ότι δεν χάνουμε τίποτα στο πρόβλημα μετασχηματίζοντας τα δεδομένα μας να έχουν μέση τιμή 0. Το C θα εμφανίζεται είτε το κάνουμε είτε όχι.

Στόχος μας είναι να βρούμε την τιμή του w η οποία μεγιστοποιεί το σ^2 και κατά συνέπεια το $w^T C w$. Θα προσπαθήσουμε να αυξήσουμε την τιμή του $w^T C w$ αυξάνοντας την τιμή των στοιχείων στο w και γι'αυτό το λόγο περιορίζουμε το μήκος του να είναι, έχοντας $w^T w = 1$. Επειδή ψάχνουμε να βρούμε μέγιστο μίας συνάρτησης έχοντας έναν περιορισμό θα χρησιμοποιήσουμε την Λανγκρασιανη και τους συντελεστές λανγκρανς που χρησιμοποιήσαμε στις μηχανές διανυσμάτων υποστήριξης.

$$L = w^T C w - \lambda(w^T w - 1)$$

Μηδενίζοντας την μερική παράγωγο ως προς w , θα έχουμε:

$$\frac{\partial L}{\partial w} = 2Cw - 2\lambda w = 0$$

$$Cw = \lambda w$$

Η παραπάνω εξίσωση είναι η γνώστη μας εξίσωση από την γραμμική άλγεβρα η οποία μας δίνει τις ιδιοτιμές και τα ιδιοδιανύσματα. Έτσι μπορούμε να συμπεράνουμε ότι η προβολή w που μεγιστοποιεί την διακύμανση είναι ένα από τα ιδιοδιανύσματα του πίνακα C. Όμως θα υπάρχουν M ιδιοδιανύσματα, άρα πως θα βρούμε ποιό μας δίνει την μέγιστη διακύμανση; Το σ_x^2 όμως δίνεται από τον τύπο:

$$\sigma_x^2 = w^T C w$$

Προσθέτοντας στο πρώτο μέλος το $w^T w = 1$ και αφαιρόντας και από τα δυο μέλη το w^T θα έχουμε

$$w^T w \sigma^2 = w^T C w$$

$$\sigma^2 w = Cw$$

για το οποίο θα έχουμε ότι για δοσμένο ζεύγος ιδιοτιμών/ιδιοδιανυσμάτων (λ, w), το λ αντιστοιχεί στην διακύμανση των δεδομένων στον προβαλλόμενο χώρο που ορίζεται από το w. Εαν βρούμε M ζεύγη από ιδιοτιμές/ιδιοδιανύσματα για τον πίνακα C, το ζεύγος με την μεγαλύτερη ιδιοτιμή αντιστοιχεί στην προβολή με την μεγαλύτερη διακύμανση, w_1. Η δεύτερη μεγαλύτερη ιδιοτιμή αντιστοιχεί στο w_2 και ούτω καθεξής.

Συνοψίζοντας, για να εκτελέσουμε τον αλγόριθμο PCA σε ένα σύνολο δεδομένων, $y_1, y_2, ..., y_N$ χρειάζεται να ακολουθήσουμε τα παρακάτω βήματα (θεωρούμε $\Upsilon = [y_1, ..., y_N]^T$)

1. Μετασχηματίζουμε τα Μ-διάστατα δεδομένα έτσι ώστε να έχουν μέση τιμή 0, δηλαδή $\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n = 0$.

2. Υπολογίζουμε τον πίνακα $C = \frac{1}{N} Y^T Y$.

3. Βρίσκουμε τα Μ ζεύγη των ιδιοτιμών/ιδιοδιανυσμάτων του πίνακα C.

4. Βρίσκουμε τα ιδιοδιανύσματα που σχετίζονται με τις D μεγαλύτερες ιδιοτιμές, $w_1, ..., w_D$.

5. Παράγουμε τη d-ιοστή διάσταση για το αντικείμενο n στην προβολή $x_{nd} = w_d^T y_n$ ή $X = YW$ με $W = [w_1, ..., w_D]$ ένας $M \times D$ πίνακας με στήλες τις ιδιοτιμές του πίνακα C και το Χ ένας $N \times D$ που ορίζεται ως $X = [x_1, ..., x_N]^T$.

6.3.3 Υλοποίηση PCA στην R

Φορτώνουμε την βάση δεδομένων mtcars. Θα κρατήσουμε τις στήλες 1-7 και 10,11 μιας και ο αλγόριθμος PCA λειτουργεί καλύτερα με αριθμητικά δεδομένα και οι στείλες 8 και 9 είναι διακριτές μεταβλητές.

```
# Loading the dataset and libraries needed
library(AMR)
library(ggplot2)
data("mtcars")

# Filtering the dataset and creating the PCA
mtcars <- mtcars[,c(1:7,10,11)]
mtcars.pca <- prcomp(mtcars, center = TRUE,scale. = TRUE)
```

Βλέπουμε ότι τα δεδομένα μας έχουν 9 διαστάσεις, πράγμα που δεν μας βοηθάει να τα παραστήσουμε γραφικά

```
summary(mtcars.pca) # PCA summarization

## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.3782 1.4429 0.71008 0.51481 0.42797 0.35184 0.32413
## Proportion of Variance 0.6284 0.2313 0.05602 0.02945 0.02035 0.01375 0.01167
## Cumulative Proportion  0.6284 0.8598 0.91581 0.94525 0.96560 0.97936 0.99103
##                           PC8     PC9
## Standard deviation     0.2419 0.14896
## Proportion of Variance 0.0065 0.00247
## Cumulative Proportion  0.9975 1.00000
```

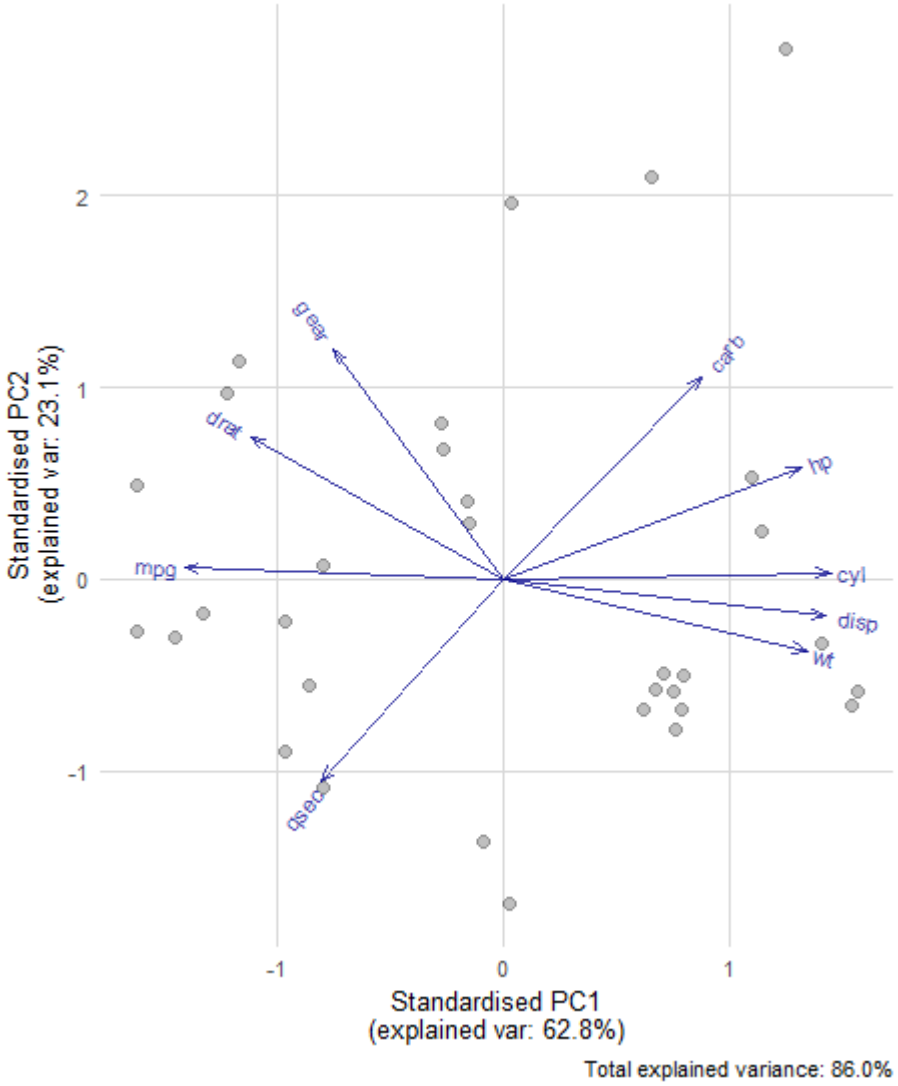
Βλέπουμε τις συντεταγμένες για τα 6 πρώτα αμάξια με την χρήση των PC1 και PC2

```
head(mtcars.pca$x[,1:2])
```

```
##                           PC1        PC2
## Mazda RX4          -0.66422351  1.1734476
## Mazda RX4 Wag      -0.63719807  0.9769448
## Datsun 710         -2.29973601 -0.3265893
## Hornet 4 Drive     -0.21529670 -1.9768101
## Hornet Sportabout   1.58697405 -0.8287285
## Valiant             0.04960512 -2.4466838
```

Παρατηρούμε από τον παραπάνω πίνακα ότι το διάνυσμα PC1 εξηγεί περίπου το 63% της συνολικής διακύμανσης και το PC2 το 23% της συνολικής διακύμανσης, άρα αυτά τα δύο διανύσματα εξηγούν συνολικά το 86% της συνολικής διακύμανσης .

```
ggplot_pca(mtcars.pca) # Plotting the PCA
```



Σύνοψη κεφαλαίου

Στο παραπάνω κεφάλαιο αρχικά έγινε μια παρουσίαση της μη επιβλεπόμενης μάθησης και των διαφορών της σε σχέση με την επιβλεπόμενη μάθηση.

Παρουσιάσαμε τους δύο αντιπροσωπευτικότερους αλγόριθμους συσταδοποίησης, τον αλγόριθμο των k-μέσων και την ιεραρχική συσταδοποίηση

καθώς επίσης δόθηκε και η γεωμετρική ερμηνεία αυτών των αλγορίθμων. Ακόμα εφαρμόσαμε τα μοντέλα μας σε βάσεις δεδομένων δημιουργόντας συστάδες που θα μας βοηθήσουν να βρούμε μοτίβα σχέσεων ανάμεσα στα δεδομένα. Δυστυχώς στην μη επιβλεπόμενη μάθηση η αξιολόγηση του μοντέλου μας είναι δυσκολότερη σε σύγκριση με την επιβλεπόμενη μιας και δεν υπάρχουν στατιστικά που να μας βοηθούν γι'αυτό κυρίως η αξιολόγηση γίνεται κυρίως εμπειρικά.

Τέλος είδαμε τον αλγόριθμο PCA, ένα σημαντικό εργαλείο το οποίο μας βοηθά να κατανοήσουμε τα δεδομένα μας και να τα παραστήσουμε γραφικά ακόμα και σε βάσεις δεδομένων με πολλά χαρακτηριστικά και κατά συνέπεια πολλές διαστάσεις.

7 Σύγκριση αλγορίθμων ταξινόμησης

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε μια κάπως πολυπλοκότερη βάση δεδομένων με σκοπό να αξιολογήσουμε και να συγκρίνουμε όλα τα μοντέλα επιβλεπόμενης μάθησης για προβλήματα ταξινόμησης που είδαμε σε αυτή την Διπλωματική Εργασία. Μία βάση με ένα αρκετά ευαίσθητο θέμα, αυτό του καρκίνου του μαστού. Τα μοντέλα μας αξιοποιώντας τις μετρήσεις ενός όγκου θα προσπαθήσουν να προβλέψουν αν είναι καλοήθης (B - Benign) ή κακοήθης (M - Malignant).

Ξεκινάμε φορτώνοντας τις βιβλιοθήκες που θα χρησιμοποιήσουμε καθώς επίσης και την βάση δεδομένων μας ενώ αφαιρούμε την στήλη 1 με τον κωδικό του ασθενή και την στήλη 33 μιας και δεν έχει καθόλου στοιχεία.

```
# Loading the libraries and dataset
library(dplyr)
library(caret)
library(randomForest)
library(MASS)
library(klaR)
library(kernlab)
library(AMR)
library(factoextra)
library(pca3d)
can <- read.csv("C:/Users/tit0v/R/PTUXIAKH/Breast_Cancer_Wisconsin.csv")
can <- can[,c(-1,-33)] # removing id and X columns
```

Θα χωρίσουμε τα δεδομένα μας σε δεδομένα εκπαίδευσης και δεδόμενα δοκιμής με αναλογία 80/20. Θέτουμε το set.seed με 1234 για να μπορούμε να αναπαράγουμε πάντα τα ίδια αποτελέσματα.

```
# Create indices (1 and 2) with 80/20 proportion to split the data
# ind==1 with probability 80% for train set and ind==2 with probability
# 20% is for test set, so we filter the data using this two indices.
set.seed(1234)
indi <- sample(2,nrow(can),replace=T,prob=c(0.8,0.2))
trainset<- can[indi==1,] #filtering the dataset to create the train set
testset <- can[indi==2,] #filtering the dataset to create the test set
```

Για να μπορέσουμε να πάρουμε καλύτερες ρυθμίσεις των υπερπαραμέτρων μας για τα διάφορα μοντέλα ταξινόμησης θα χρησιμοποιήσουμε την τεχνική K- fold cross validation όπου ουσιαστικά θα χωρίσουμε τα δεδομένα εκπαίδευσης σε K μέρη παίρνοντας το ένα από αυτά σαν δεδομένα δοκιμών και τα υπόλοιπα K-1 σας δεδομένα εκπαίδευσης. Σε κάθε παραλλαγή θα δοκιμάζονται διάφοροι υπερπαράμετροι πράγμα που θα μας βοηθήσει να βελτιώσουμε την απόδοση του μοντέλου μας.

```
# Creating the train control for 10-fold cross validation
tcontrol <- trainControl(method = "cv", number = 10)
```

Δημιουργούμε τα μοντέλα μας

```r
# K-Nearest Neighbors model for centered and scaled set
modelKNN <- train(diagnosis ~ ., data = trainset, method = "knn",
preProcess = c("center", "scale"), trControl = tcontrol)
# Naive Bayes model
modelNB <- train(diagnosis ~ ., data = trainset, method = "nb",
trControl = tcontrol)
# Random Forest model with 100 trees
modelRF <- train(diagnosis ~ ., data = trainset, method = "rf",
ntree = 100,importance = T, trControl = tcontrol)
# Logisitic Regression
modelLG <- train(diagnosis ~ ., data = trainset, method = "glm",
family = binomial,trControl = tcontrol)
# Desicion Tree
modelDT <- train(diagnosis ~ ., data = trainset, method = "rpart",
trControl = tcontrol)
# SVM Linear
modelSVML <- train(diagnosis ~ ., data = trainset, method = "svmLinear",
trControl = tcontrol)
# SVM Radial
modelSVMR <- train(diagnosis ~ ., data = trainset, method = "svmRadial",
trControl = tcontrol)
```

Ας δούμε τώρα μερικά στοιχεία για τα μοντέλα μας

```r
modelKNN$results # KNN detailed results

##   k  Accuracy     Kappa AccuracySD    KappaSD
## 1 5 0.9649697 0.9230712 0.02095547 0.04610994
```

```
## 2 7 0.9583513 0.9078679 0.02602043 0.05825018
## 3 9 0.9626991 0.9175648 0.02535944 0.05591176
```

Παρατηρούμε ότι για k=5 το μοντέλο μας δίνει μεγαλύτερη ακρίβεια.

```
modelNB$results # NB detailed results

##   usekernel fL adjust  Accuracy     Kappa AccuracySD    KappaSD
## 1     FALSE  0      1 0.9272423 0.8400410 0.05247220 0.11725281
## 2      TRUE  0      1 0.9427999 0.8754005 0.03791512 0.08362355
```

Ο αλγόριθμος Naive Bayes μας δίνει καλύτερα αποτελέσματα όταν χρησιμοποιούμε πυρήνα

```
modelRF$results # RF detailed results

##   mtry  Accuracy     Kappa AccuracySD    KappaSD
## 1    2 0.9627053 0.9183788 0.02080450 0.04499366
## 2   16 0.9627053 0.9185200 0.02319154 0.05018545
## 3   30 0.9582609 0.9093250 0.02421806 0.05280394
```

Στο αλγόριθμο τυχαίου δάσους χρησιμοποιήσαμε 100 δένδρα ενώ παρόλο που η ακρίβεια για 2 ή 16 χαρακτηριστικά είναι η ίδια η τυπική απόκλιση για 2 είναι ελαφρώς χαμηλότερη άρα τα 2 χαρακτηριστικά ίσως είναι καλύτερη επιλογή, αν και κάποιος αριθμός ανάμεσα στο 2 και το 16 ίσως να μας έδινε ακόμα καλύτερα αποτελέσματα.

```
modelSVMR$results # SVM Radial detailed results

##        sigma    C  Accuracy     Kappa AccuracySD    KappaSD
## 1 0.04862012 0.25 0.9516405 0.8940219 0.03097323 0.06778323
## 2 0.04862012 0.50 0.9582126 0.9088252 0.03014779 0.06505044
## 3 0.04862012 1.00 0.9648792 0.9238652 0.02766358 0.05851441
```

Παρατηρούμε ότι για το μοντέλο τον μηχανών διανυσμάτων υποστήριξης με χρήση του γκαουσιανού πυρήνα για C=1 έχουμε καλύτερα αποτελέσματα.

Χρησιμοποιόντας τα μοντέλα που δημιουργήσαμε θα κάνουμε τις προβλέψεις μας πάνω στα δεδομένα δοκιμών.

```
pKNN <- predict(modelKNN, testset) # K-NN prediction
pNB <- predict(modelNB, testset) # Naive Bayes prediction
pRF <- predict(modelRF, testset) # Random forest prediction
pLG <- predict(modelLG, testset) # Logistic Regression prediction
pDT <- predict(modelDT,testset) # Desicion Tree prediction
pSVML <- predict(modelSVML, testset) # SVM Linear prediction
pSVMR <- predict(modelSVMR, testset) # SVM Radial prediction
```

Ακόμα θα δημιουργήσουμε τους πίνακες σύγκρισης για το κάθε μοντέλο

```
# Confusion Matrix for all models
cmKNN <- confusionMatrix(testset$diagnosis, pKNN)
cmNB <- confusionMatrix(testset$diagnosis, pNB)
cmRF <- confusionMatrix(testset$diagnosis, pRF)
cmLG <- confusionMatrix(testset$diagnosis, pLG)
cmDT <- confusionMatrix(testset$diagnosis, pDT)
```

```
cmSVML <- confusionMatrix(testset$diagnosis, pSVML)
cmSVMR <- confusionMatrix(testset$diagnosis, pSVMR)
```

Επίσης θα δημιουργήσουμε τον πίνακα με τα συγκεντρωτικά στοιχεία όλων τον μοντέλων μας.

```
# Model vector column
Model_Type <- c("K nearest neighbor", "Naive Bayes", "Random forest",
"Logistic regression", "Desicion Tree","SVM Linear", "SVM Radial")
# Accuracy vector column
Test_Accuracy <-c(cmKNN$overall[1], cmNB$overall[1], cmRF$overall[1],
cmLG$overall[1], cmDT$overall[[1]], cmSVML$overall[[1]],
cmSVMR$overall[[1]])
# Sensitivity vector column
Test_sensitivity <- c(cmKNN$byClass[[1]],cmNB$byClass[[1]],
cmRF$byClass[[1]],cmLG$byClass[[1]],cmDT$byClass[[1]],
cmSVML$byClass[[1]],cmSVMR$byClass[[1]])
# Specificity vector column
Test_specificity <-c(cmKNN$byClass[[2]],cmNB$byClass[[2]],
cmRF$byClass[[2]],cmLG$byClass[[2]],cmDT$byClass[[2]],
cmSVML$byClass[[2]],cmSVMR$byClass[[2]])

# Creating and printing the table
metrics <- data.frame(Model_Type,Test_Accuracy,Test_sensitivity,
Test_specificity)
names(metrics)<-c("Model","Test Accuracy","Sensitivity","Specificity")
knitr::kable(metrics, digits = 4)
```

Model	Test Accuracy	Sensitivity	Specificity
K nearest neighbor	0.9646	0.9420	1.0000
Naive Bayes	0.9292	0.9524	0.9000
Random forest	0.9381	0.9531	0.9184
Logistic regression	0.9735	0.9697	0.9787
Desicion Tree	0.9027	0.8971	0.9111
SVM Linear	0.9735	0.9559	1.0000
SVM Radial	0.9823	0.9846	0.9792

Παρατηρούμε από τον παραπάνω πίνακα ότι την καλύτερη απόδοση έχει το μοντέλο των μηχανών διανυσμάτων υποστήριξης με γκαουσιανό πυρήνα.

```
cmSVMR$table # SVM Radial Confusion matrix

##           Reference
## Prediction  B  M
##          B 64  1
##          M  1 47
```

Βλέπουμε ότι από τα 113 άτομα μόνο δύο είναι λάθος ταξινομημένα. Βέβαια επειδή έχουμε να κάνουμε με ένα αρκετά ευαίσθητο θέμα μερικές φορές η ακρίβεια δεν είναι το μόνο ζητούμενο. Στο συγκεκριμένο πίνακα σύγκρισης βλέπουμε ότι ένας ασθενής με κακοήθη όγκο θα ταξινομηθεί ως καλοήθης. Αυτό το σφάλμα ονομάζεται σφάλμα τύπου-ΙΙ και είναι αρκετά επικίνδυνο μιας και η ζωή του ασθενή θα κινδυνεύει ενώ ο αλγόριθμος θα μας δείχνει ότι είναι ασφαλής. Άρα σε τέτοιου είδους προβλήματα αρκετά σημαντικό είναι να κοιτάμε και την ειδικότητα, γιατί είναι προτιμότερο να χαρακτηρίσεις

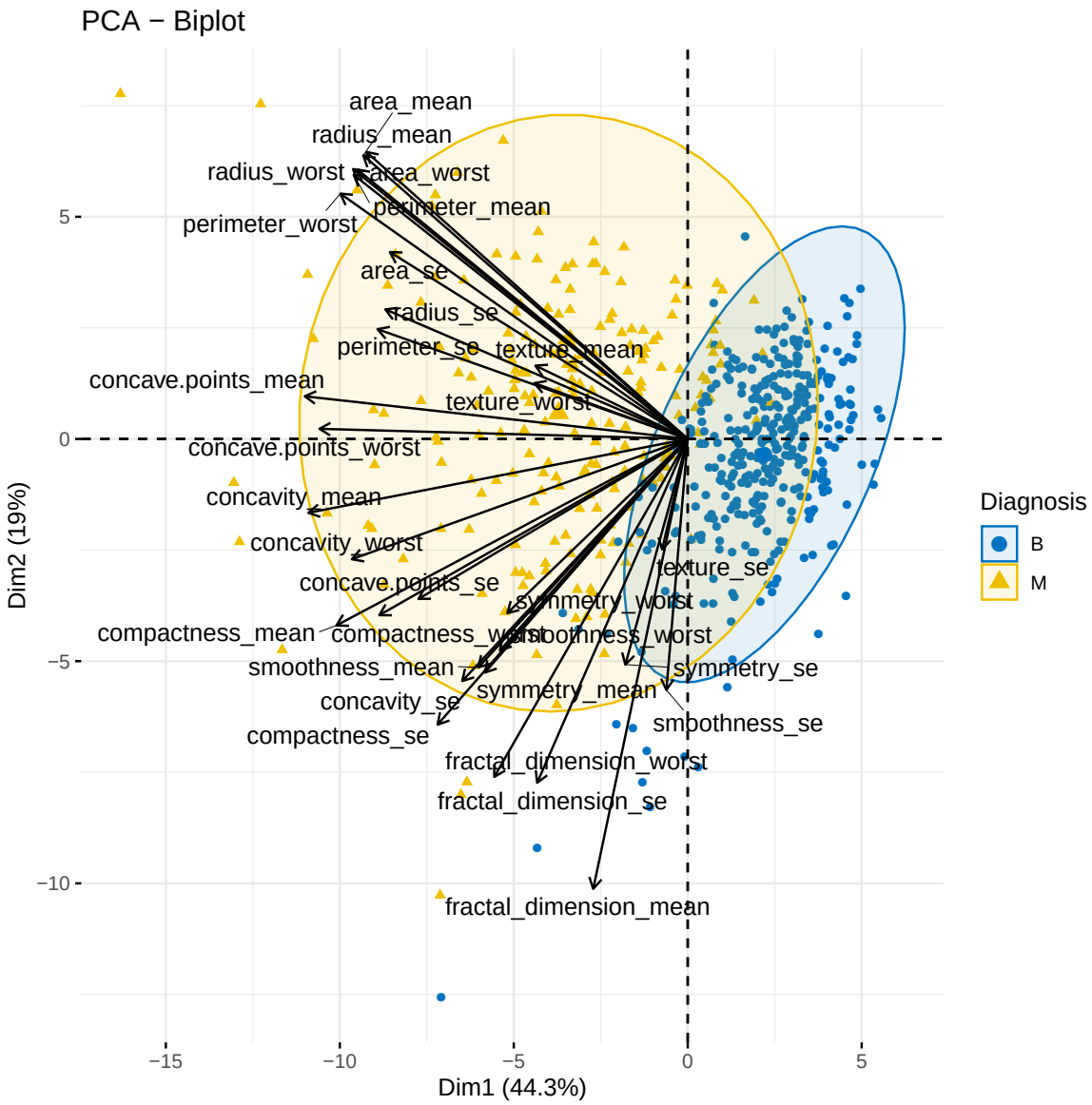
έναν καλοήθη όγκο ως κακοήθη και στην πορεία να διορθώσεις το πόρισμα παρά να υποβαθμίσεις έναν κακοήθη όγκο. Για παράδειγμα το μοντέλο των μηχανών διανυσμάτων υποστήριξης με γραμμικό πυρήνα μπορεί να έχει χαμηλότερη ακρίβεια αλλά έχει 100% ειδικότητα. Έχει 3 λάθος ταξινομήσεις αλλά καμία τύπου-ΙΙ. Άρα κανείς σοβαρά άρρωστος δεν θα ταξινομηθεί λάθος. Έτσι για το συγκεκριμένο πρόβλημα οι καταλληλότεροι αλγόριθμοι είναι οι μηχανές διανυσμάτων υποστήριξης με γραμμικό πυρήνα και ο αλγόριθμος k-κοντινότερων γειτόνων.

```
cmSVML$table # SVM Linear Confusion matrix

##           Reference
## Prediction  B  M
##          B 65  0
##          M  3 45
```

Επειδή η βάση δεδομένων μας έχει 31 χαρακτηριστικά αυτό την κάνει αδύνατο να μπορέσουμε να την παραστήσουμε γραφικά και να δούμε κάποια γεωμετρικά μοτίβα ή συστάδες. Μπορούμε όμως να αποφύγουμε αυτό το πρόβλημα και να παραστήσουμε κατά κάποιο τρόπο τα δεδομένα μας χρησιμοποιόντας την ανάλυση κύριων συνιστωσών (PCA).

Στο παρακάτω γράφημα θα δούμε τα δεδομένα μας χρησιμοποιόντας τα PCA1 και PCA2 τα οποία εξηγούν το 63.3% των συνολικών δεδομένων μας

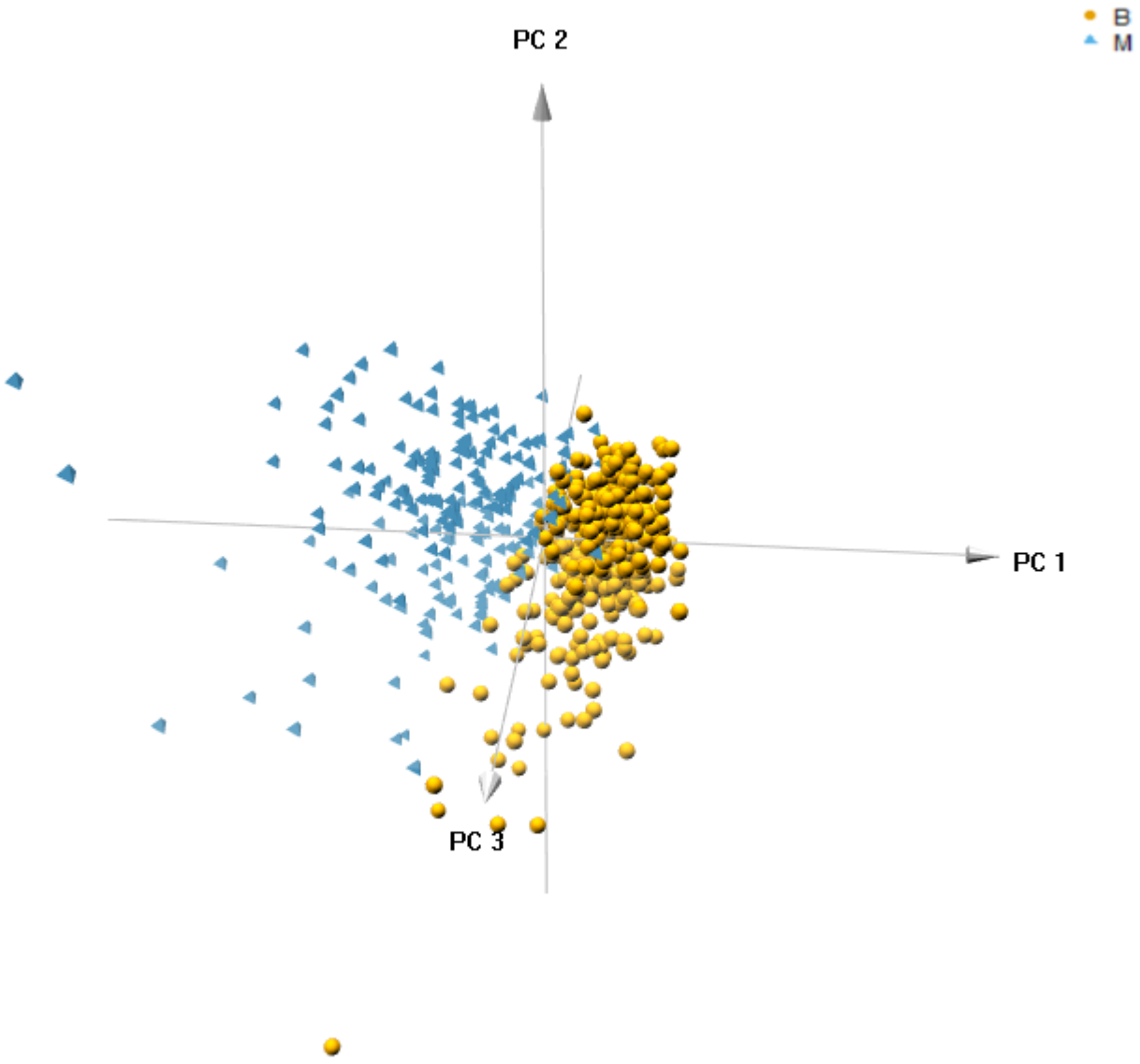
```r
# Creating PCA biplot
all_pca <- prcomp(can[,-1], center = TRUE, scale. = TRUE)
fviz_pca_biplot(all_pca,col.ind = can$diagnosis, col="black",
palette = "jco", geom = "point", repel=TRUE,
legend.title="Diagnosis", addEllipses = TRUE)
```



Ενώ στις τρεις διαστάσεις έξηγόντας το 72% των δεδομένων θα έχουμε το

τρισδιάστατο γράφημα

```
pca3d(all_pca, group=can$diagnosis) # PCA 3d plot
```



Όπως και στα άλλα παραδείγματα, έτσι και εδώ θα τρέξουμε τα μοντέλα

σε διάφορες εκδοχές διαχωρισμού των δεδομένων έτσι ώστε να βγάλουμε ασφαλέστερα συμπεράσματα για τις επιδόσεις των μοντέλων. Θα τρέξουμε τον αλγόριθμο k-fold cross validation για K=10 και αυτό θα το επαναλάμβουμε δέκα φορές, έτσι ώστε στο τέλος το μοντέλο μας να εκπαιδευτεί σε 100 διαφορετικά πειράματα πάνω στην βάση δεδομένων μας.

```
set.seed(1928)
# Creating the folds, we use 10-fold cross validation and
# we gonna repeat it for 10 times.
cvfolds7 <- createMultiFolds(can$diagnosis ,k=10,times=10)
# Generating the function that control how models are created
tcontrol7 <- trainControl(method = "repeatedcv", number = 10,
                          repeats = 10, index=cvfolds7)
```

```
# K-Nearest Neighbors model for centered and scaled set
modelKNN2 <- train(diagnosis ~ ., data = trainset, method = "knn",
preProcess = c("center", "scale"), trControl = tcontrol7)
# Naive Bayes model
modelNB2 <- train(diagnosis ~ ., data = trainset, method = "nb",
trControl = tcontrol7)
# Random Forest model with 100 trees
modelRF2 <- train(diagnosis ~ ., data = trainset, method = "rf",
ntree = 100,importance = T, trControl = tcontrol7)
# Logisitic Regression
modelLG2 <- train(diagnosis ~ ., data = trainset, method = "glm",
family = binomial,trControl = tcontrol7)
# Desicion Tree
```

```
modelDT2 <- train(diagnosis ~ ., data = trainset, method = "rpart",
trControl = tcontrol7)
# SVM Linear
modelSVML2 <- train(diagnosis ~ ., data = trainset, method = "svmLinear",
trControl = tcontrol7)
# SVM Radial
modelSVMR2 <- train(diagnosis ~ ., data = trainset, method = "svmRadial",
trControl = tcontrol7)
```

Τα στοιχεία των μοντέλων μας σχετικά με την ακρίβεια βρίσκονται συγκεντρωμένα στον παρακάτω πίνακα.

Μοντέλο	Mean	Min	Max	sd	95% CI
K Nearest Neighbor	0.9685	0.8947	1	0.0217	0.9642-0.9728
Naive Bayes	0.9397	0.8596	1	0.0292	0.9339-0.9455
Random Forest	0.9643	0.8947	1	0.0245	0.9594-0.9691
Logistic Regression	0.9481	0.8793	1	0.0288	0.9424-0.9538
Desicion Tree	0.9270	0.8596	0.9825	0.0314	0.9208-0.9333
SVM Linear	0.9727	0.8947	1	0.0203	0.9687-0.9767
SVM Radial	0.9752	0.9123	1	0.0186	0.9715-0.9789

Πίνακας 10: Σύγκριση αλγορίθμων ταξινόμησης μετά από 100 επαναλήψεις

Παρατηρούμε ότι και μετά από 100 επαναλήψεις ο αλγόριθμος των μηχανών διανυσμάτων υποστήριξης με γκαουσιανό πυρήνα μας δίνει την μεγαλύτερη ακρίβεια, ενώ έχοντας πολύ μικρή απόκλιση τον κάνει και αρκετά πιό ασφαλή όσο αφορά τις διάφορες παραλλαγές στον διαχωρισμό των δεδομένων. Τέλος μπορούμε να δούμε ότι σε συγκεκριμένες εκδοχές των δεδομένων σχεδόν όλοι οι αλγόριθμοι έχουν πετύχει απόλυτη ταξινόμηση. Από την άλλη ο αλγόριθμος με τα χειρότερα αποτελέσματα είναι τα δένδρα απόφασης με

μόλις 92% ακρίβεια και την μεγαλύτερη απόκλιση, ενώ το πάνω άκρο του 95% διαστήματος εμπιστοσύνης είναι χαμηλότερο από το κάτω άκρο όλων των υπόλοιπων αλγορίθμων.

ΠΑΡΑΡΤΗΜΑ Α΄ : Βάσεις Δεδομένων

Στην πτυχιακή χρησιμοποιήσαμε διάφορες βάσεις δεδομένων για να υλοποιήσουμε στην R τους αλγόριθμους που παρουσιάσαμε. Σε αυτό το παράρτημα θα δούμε πως μπορούμε να κατεβάσουμε αυτές τις βάσεις δεδομένων, τι δεδομένα περιέχουν και πως μπορούν σε προβλήματα επιβλεπόμενης μάθησης να μας οδηγήσουν σε προβλέψεις.

Οι βάσεις δεδομένων Salary_Data , Position_Salaries και 50_Startups

Στο δεύτερο κεφάλαιο χρησιμοποιήσαμε τη βάση δεδομένων Salary_Data.csv . Την βάση δεδομένων μπορούμε να την κατεβάσουμε από το `https://www.kaggle.com/vihansp/salary-data`. Αποτελείται από 30 γραμμές και δύο στήλες. Κάθε γραμμή είναι μια καταχώριση όπου αναφέρονται τα χρόνια εμπειρίας και ο μισθός. Σκοπός της βάσης μας είναι με την χρήση μοντέλων παλινδρόμησης να βρούμε το κατάλληλο μισθό όταν μας είναι γνωστό τα χρόνια προυπηρεσίας. Για την σύγκριση του γραμμικού μοντέλου μας με το πολυωνυμικό δουλέψαμε πάνω στην βάση δεδομένων Position_Salaries.csv η οποία ήταν παρόμοια με την Salary_Data.csv και την οποία μπορούμε να την βρούμε στο `https://www.kaggle.com/akram24/position-salaries` . Τέλος η γραμμική παλινδρόμηση πολλών μεταβλητών εφαρμόστηκε στην 50_Startups .csv απο το `https://www.kaggle.com/farhanmd29/50-startups` , μια βάση δεδομένων η οποία αποτελείται από 50 καταχωρίσεις εταιρειών σταρταπ και τα έσοδα/έξοδα τους σε διάφορους τομείς.

```r
dim(df)
```

```
## [1] 30  2
```

```r
head(df,3)
```

```
##   YearsExperience Salary
## 1             1.1  39343
## 2             1.3  46205
## 3             1.5  37731
```

```r
dim(datas)
```

```
## [1] 10  3
```

```r
head(datas,3)
```

```
##            Position Level Salary
## 1  Business Analyst     1  45000
## 2 Junior Consultant     2  50000
## 3 Senior Consultant     3  60000
```

```r
dim(dataf)
```

```
## [1] 50  5
```

```r
head(dataf,3)
```

```
##   R.D.Spend Administration Marketing.Spend      State   Profit
## 1  165349.2       136897.8        471784.1   New York 192261.8
## 2  162597.7       151377.6        443898.5 California 191792.1
## 3  153441.5       101145.6        407934.5    Florida 191050.4
```

Η βάση δεδομένων Binary

Για το τρίτο κεφάλαιο εφαρμόσαμε τα μοντέλα ταξινόμησης στην βάση δεδομένων Binary.csv . Μπορούμε να βρούμε την βάση στο `https://stats.idre.ucla.edu/stat/data/binary.csv` . Η βάση είναι ένας πίνακας 400 γραμμών και 4 στηλών. Η πρώτη στήλη admit έχει να κάνει με το γεγονός αν ο υποψήφιος έγινε δεκτός ή όχι στο πανεπιστήμιο με 1 να σημαίνει ότι έγινε δεκτός και 0 ότι δεν έγινε. GPA είναι η στήλη με τον μέσο όρο του μαθητή, GRE είναι ο βαθμός που πέτυχε ο μαθητής στο GRE τεστ και rank είναι η κλάση του πανεπιστημίου με 1 να είναι η κλάση για τα πανεπιστήμια με το υψηλότερο κύρος και 4 τα πανεπιστήμια με το χαμηλότερο κύρος. Σκοπός της βάσης μας είναι δοσμένου των GPA, GRE και rank να προβλέψουμε αν ο υποψήφιος θα γίνει δεκτός ή όχι στο πανεπιστήμιο για το οποίο έκανε αίτηση.

```
dim(binary)

## [1] 400   4

head(binary,3)

##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
```

Η βάση δεδομένων Social_Network_Ads

Στο τέταρτο κεφάλαιο χρησιμοποιήσαμε δύο βάσεις. Την βάση Salary_Data .csv που είδαμε και στο δεύτερο κεφάλαιο και την βάση δεδομένων Social Network Ads.csv την οποία μπορούμε να την βρούμε στο `https://www.kaggle.com/rakeshrau/social-network-ads` . Η βάση έχει 400 καταχωρίσεις όπου κάθε μία απο αυτές έχει 5 χαρακτηριστικά. User id που είναι ο αριθμός του χρήστη, Gender το φύλο του χρήστη, Age η ηλικία του χρήστη, Estimated Salary ο εκτιμώμενος μισθός και Purchased όπου για 1 ο πελάτης αγόρασε το προιόν της διαφήμισης και 0 δεν το αγόρασε. Σκοπός την βάσης είναι για δοσμένη ηλικία, φύλο και εκτιμώμενο μισθό με χρήση των αλγορίθμων ταξινόμησης να προβλέψουμε αν ο πελάτης θα αγόρασει το προιόν ή όχι.

```
dim(dataset3)

## [1] 400   5

head(dataset3,3)

##    User.ID Gender Age EstimatedSalary Purchased
## 1 15624510   Male  19           19000         0
## 2 15810944   Male  35           20000         0
## 3 15668575 Female  26           43000         0
```

Η βάση δεδομένων iris

Η βάση δεδομένων που θα χρησιμοποιήσουμε στο πέμπτο κεφάλαιο θα

είναι η iris. Η βάση περιέχεται μέσα στην R άρα δεν χρειάζεται να την κατεβάσουμε. Αρκεί να τρέξουμε την εντολή data(”iris”) για να φορτώσουμε την βάση δεδομένων στο σύστημα μας. Η βάση iris αποτελείται από 150 καταχωρίσεις λουλουδιών όπου για την κάθε μια έχουμε μετρήσεις σχετικά με το μηκος και πλάτος των πετάλων και των φύλλων και στην στήλη Species έχουμε το είδος του λουλουδιού. Σκοπός μας είναι έχοντας τις μετρήσεις του μήκους και πλάτους των πετάλων και των φύλλων να προβλέψουμε το είδος του λουλουδιού.

```
dim(iris)

## [1] 150   5

head(iris,3)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Η βάση δεδομένων USArrests

Για την εφαρμογή των αλγορίθμων μη επιβλεπόμενης μάθησης στο έκτο κεφάλαιο θα χρησιμοποιήσουμε τρεις βάσεις δεδομένων. Η υλοποίηση των k-μέσων θα γίνει πάνω στην βάση USArrests που περιέχεται στην R και μπορείς να την φορτώσεις με την εντολή data(”USArrests”) . Η βάση έχει 50 κα-

ταχωρίσεις, μία για κάθε πολιτεία της αμερικής καθως επίσης και των αριθμό των φόνων, των επιθέσεων και των βιασμών ανά 100.000 κατοίκους όπως επίσης και το αριθμητικό ποσοστό του αστικού πληθυσμού.

```
dim(USArrests)

## [1] 50  4

head(USArrests,3)

##         Murder Assault UrbanPop Rape
## Alabama   13.2     236       58 21.2
## Alaska    10.0     263       48 44.5
## Arizona    8.1     294       80 31.0
```

Η βάση δεδομένων utilities

Για την υλοποίηση της ιεραρχικής συσταδοποίησης χρησιμοποιήσαμε την βάση δεδομένων utilities.csv την οποία μπορούμε να κατεβάσουμε από το `https://drive.google.com/drive/folders/1dZSlyTUIGlWsKoCENDIrCdovcRFs7xx8` . Περιέχει διάφορα έξοδα, έσοδα και οικονομικά στοιχεία για 22 εταιρίες.

```
dim(mydata)

## [1] 22  9

head(mydata,3)

##    Company Fixed_charge  RoR Cost Load D.Demand Sales Nuclear Fuel_Cost
```

```
## 1 Arizona          1.06  9.2  151 54.4      1.6  9077     0.0     0.628
## 2  Boston          0.89 10.3  202 57.9      2.2  5088    25.3     1.555
## 3 Central          1.43 15.4  113 53.0      3.4  9212     0.0     1.058
```

Η βάση δεδομένων mtcars

Για τον αλγόριθμο PCA χρησιμοποιήσαμε την έτοιμη βάση δεδομένων mtcars από την R στην οποία έχουμε καταχωριμένα 22 αυτοκίνητα και 11 χαρακτηριστικά σχετικά με στοιχεία και τις επιδόσης του κάθε αυτοκινήτου.

```
dim(mtcars)

## [1] 32  9

head(mtcars,3)

##                mpg cyl disp  hp drat    wt  qsec gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61    4    1
```

Η βάση δεδομένων Breast Cancer Wisconsin

Τέλος για το έβδομο κεφάλαιο θα χρησιμοποιήσουμε την βάση δεδομένων Breast_Cancer_Wisconsin.csv την οποία μπορούμε να αντλήσουμε από το `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data` . Η βάση αποτελείται από έναν πίνακα 569 γραμμών και 33 στηλών. Κάθε

γραμμή αντιπροσωπεύει έναν ασθενή και οι στήλες έχουν να κάνουν με τις διάφορες μετρήσεις που γίνονται στο συγκεκριμένο πυρήνα του κυττάρου. Ουσιαστικά οι μετρήσεις χωρίζονται σε τρείς κατηγορίες, στις _mean (μέση μέτρηση), στην _se (ελάχιστη μέτρηση) και στην _worst(μέγίστη μέτρηση). Οι μετρήσεις έχουν να κάνουν με την

- ακτίνα (radius)
- περίμετρο (perimeter)
- επιφάνεια (area)
- υφή (texture)
- ομαλότητα (smoothness)
- συμπαγότητα (compactness)
- συμμετρία (symmetry)
- κοιλότητα (concavity)

Ακόμα υπάρχει και η στήλη της διάγνωσης (diagnosis) η οποία είναι το χαρακτηριστικό στόχευσης στο οποίο θα στήσουμε το μοντέλο μας για να κάνει την πρόβλεψη. Παίρνει δύο τιμές B(Benign) αν ο όγκος είναι καλοήθης και M(Malignant) αν ο όγκος είναι κακοήθης.

```r
dim(can)
```

```
## [1] 569  31
```

```r
head(can, 3)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         M       17.99        10.38          122.8      1001         0.11840
## 2         M       20.57        17.77          132.9      1326         0.08474
## 3         M       19.69        21.25          130.0      1203         0.10960
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33           184.6       2019
## 2             0.003532        24.99         23.41           158.8       1956
## 3             0.004571        23.57         25.53           152.5       1709
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
```

```
## 3           0.1444            0.4245          0.4504                0.2430
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
```

ΠΑΡΑΡΤΗΜΑ Β΄ : Εγκατάσταση R και RStudio

1. Ανοίγουμε το `http://lib.stat.cmu.edu/R/CRAN/` και επιλέγουμε το λειτουργικό σύστημα το οποίο χρησιμοποιούμε για να κατεβάσουμε το σωστό αρχείο για την εγκατάσταση της R. Επιλέγουμε το base και μετα κατεβάζουμε το πρόγραμμα.

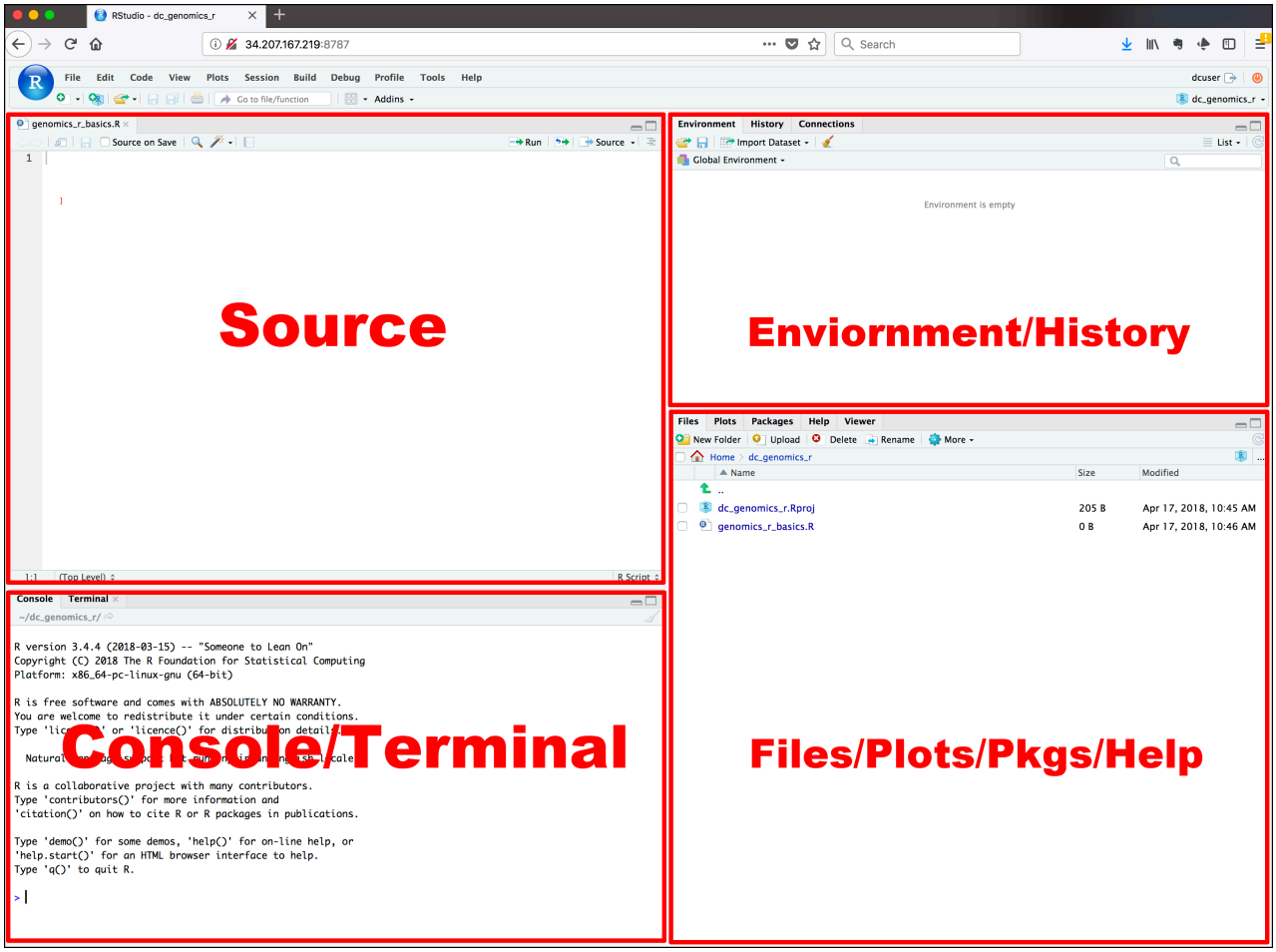
2. Εγκαθιστούμε την R. Αφήνουμε όλες τις προεπιλεγμένες ρυθμίσεις στις επιλογές εγκατάστασης. Όπου χρειαστεί πατάμε Επόμενο/Next και OK.

3. Κατεβάζουμε το RStudio από το `https://rstudio.com/products/rstudio/download/` . Επιλέγουμε την δωρεάν έκδοση και επιλέγουμε το κατάλληλο πρόγραμμα για το λειτουργικό σύστημα που χρησιμοποιούμε. Αφήνουμε όλες τις προεπιλεγμένες ρυθμίσεις στις επιλογές εγκατάστασης. Όπου χρειαστεί πατάμε Επόμενο/Next και OK.

4. Όταν η εγκατάσταση τελειώσει ανοίγουμε το RStudio. Πάντα τρέχουμε το RStudio σαν Διαχειριστές για να μην έχουμε προβλήματα στις εγκαταστάσεις των βιβλιοθηκών. Πατάμε δεξι κλικ στο RStudio και επιλέγουμε το Run as Administrator.

Ανοίγοντας το RStudio πάμε πάνω αριστερά, πατάμε File > New File > R Script για να δημιουργήσουμε ένα νέο script στην R. Το περιβάλλον της R θα αποτελείται από 4 παράθυρα. Πάνω αριστερά είναι το μέρος που γράφουμε τον κώδικα μας, κατω αριστερά είναι η κονσόλα-τερματικό που τρέχουμε τον

κώδικα, πανω δεξιά βλέπουμε τα δεδομένα που έχουμε φορτωμένα στο περιβάλλον μας καθώς και το ιστορικό των εντολών μας και κάτω δεξιά εμφανίζονται τα γραφήματα που δημιουργούμε, τα αρχεία που έχουμε στον φάκελο εργασίας, τα πακέτα που έχουμε εγκαταστημένα και ενεργά καθώς και το παράθυρο με την βοήθεια.



Σχήμα 7.1: Περιβάλλον RStudio [19]

Για να τρέξουμε ένα κομμάτι του κώδικα που γράψαμε στο script μας το επιλέγουμε και πατάμε Ctrl+Enter ή το κουμπάκι Run στο πάνω δεξιά μέ-

ρος του παράθυρου Source. Το αποτέλεσμα μαζί με το κώδικα που γράψαμε εμφανίζεται στην κονσόλα.

Αρκετά συχνά στην R χρησιμοποιούμε βιβλιοθήκες με έτοιμο κώδικα. Το μόνο που χρειάζεται είναι να εγκαταστηθούν την πρώτη φορά και να φορτώνονται κάθε φορά που τις χρειαζόμαστε. Αυτό γίνεται με τον εξής τρόπο:

```
install.packages("ggplot2")
library(ggplot2)
```

Χρησιμοποιούμε πάντα ” ” όταν κάνουμε εγκατάσταση μιας βιβλιοθήκης. Με την συνάρτηση library() φορτώνουμε την βιβλιοθήκη που χρειαζόμαστε.

Για να χρησιμοποιήσουμε κάποια βάση δεδομένων το μόνο που χρειάζεται είναι να την βάλουμε στον φάκελο εργασίας μας. Για να δούμε ποιός είναι ο φάκελος εργασίας τρέχουμε την εντολή

```
getwd()
```

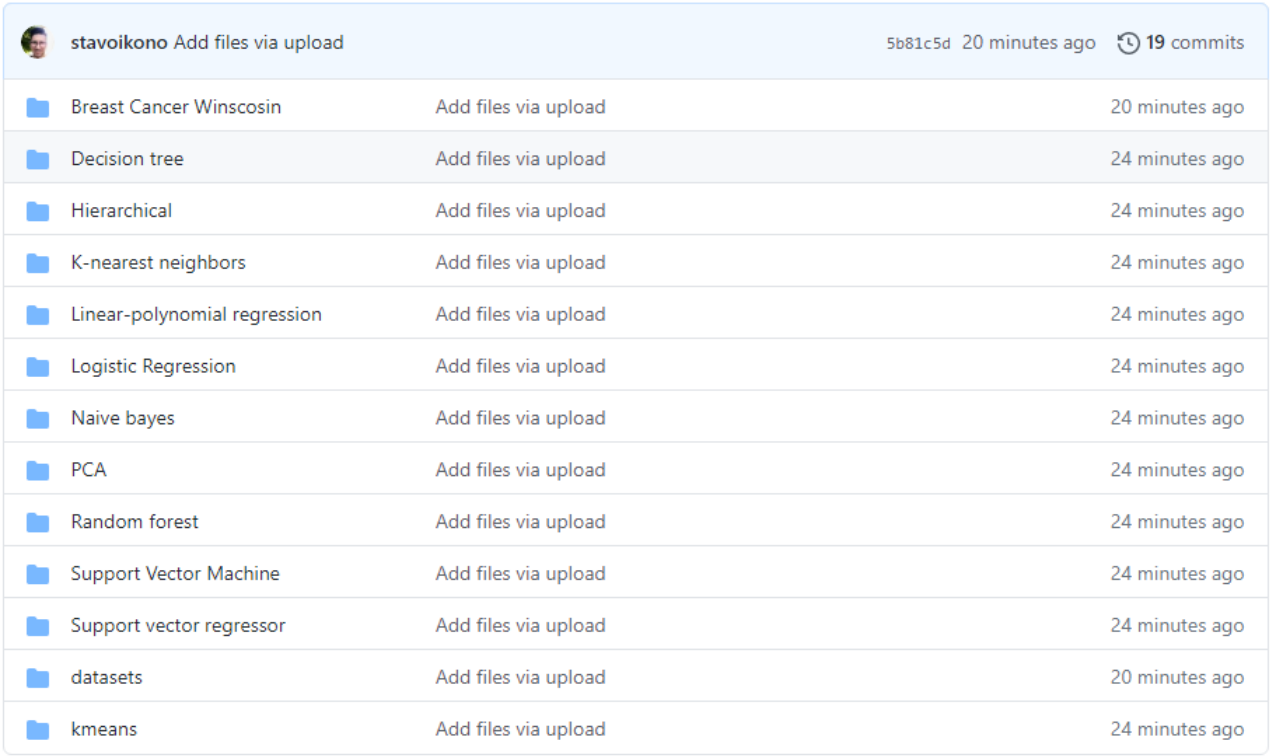
Για να φορτώσουμε την βάση δεδομένων μας αρκεί η εντολή

```
read.csv("onoma_arxeiou.csv")
```

Αλλίως μπορούμε να φορτώσουμε μια βάση δεδομένων σε οποιοδήποτε φάκελο γράφοντας ακριβώς την θέση στην οποία βρίσκεται και να την αποθηκεύσουμε με ένα συγκεκριμένο όνομα χρησιμοποιόντας την εντολή

```
data <- read.csv("C:/Users/Onoma_xrhsth/.../onoma_arxeiou.csv")
```

Όλες οι βάσεις δεδομένων και ο κώδικας για τα μοντέλα που δημιουργήθηκαν στην εργασία βρίσκονται στο `https://github.com/stavoikono/Thesis` .



stavoikono Add files via upload	5b81c5d 20 minutes ago	**19** commits
Breast Cancer Winscosin	Add files via upload	20 minutes ago
Decision tree	Add files via upload	24 minutes ago
Hierarchical	Add files via upload	24 minutes ago
K-nearest neighbors	Add files via upload	24 minutes ago
Linear-polynomial regression	Add files via upload	24 minutes ago
Logistic Regression	Add files via upload	24 minutes ago
Naive bayes	Add files via upload	24 minutes ago
PCA	Add files via upload	24 minutes ago
Random forest	Add files via upload	24 minutes ago
Support Vector Machine	Add files via upload	24 minutes ago
Support vector regressor	Add files via upload	24 minutes ago
datasets	Add files via upload	20 minutes ago
kmeans	Add files via upload	24 minutes ago

Αναφορές

[1] Andriy Burkov. *The hundred page machine learning book*. Andriy Burkov Quebec City, Can., 2019.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Number 10. Springer series in statistics New York, 2001.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer, 2013.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.

[6] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[7] Simon Rogers and Mark Girolami. *A first course in machine learning*. CRC Press, 2016.

[8] Ian H Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. ACM New York, NY, USA, 2002.

[9] Hsuantien Linlearning Yaser S. Abumostafa, Malik Magdonismail. *Learning From Data A Short Course*. AMLbook.com, 2012.

[10] I Vlachavas, P Kefalas, N Vasiliadis, F Kokkoras, and E Sakellariou. *Artificial Intelligence (Τεχνητή Νοημοσύνη) Β΄ Έκδοση*. Εκδόσεις Παν/μίου Μακεδονίας. 2005.

[11] Νικόλαος Σεργίου. *Μαθηματική ανάλυση τεχνικών κατηγοριοποίησης δεδομένων με χρήση μηχανών διανυσμάτων υποστήριξης*. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης 2017.

[12] Βασίλειος Βερύκιος, Βασίλειος Καγκλης, and Ηλίας Σταυρόπουλος. *Η επιστήμη των δεδομένων μέσα από την γλώσσα R*. www.kallipos.gr . 2015.

[13] Awad M., Khanna R. *Support Vector Regression. In: Efficient Learning Machines*. Apress, Berkeley, CA. 2015

[14] https://www.l2drive.com/storage-challenges/

[15] https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[16] https://barnraisersllc.com/2018/10/01/data-mining-process-essential-steps/

[17] https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regressi

[18] https://www.saedsayad.com/clustering_hierarchical.html

[19] https://datacarpentry.org/genomics-r-intro/01-introduction/index.
html

[20] https://www.displayr.com/what-is-hierarchical-clustering/

[21] https://towardsdatascience.com/https-towardsdatascience-com-hierarchical-clusteri

[22] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[23] https://community.alteryx.com/t5/Alteryx-Designer-Knowledge-Base/
Seeing-the-Forest-for-the-Trees-An-Introduction-to-Random-Forest/
ta-p/158062

[24] https://xingewang.wordpress.com/2018/08/02/
the-math-behind-linear-svc-classifier/

[25] http://efavdb.com/svm-classification/

[26] https://www.researchgate.net/figure/Example-of-linear-support-vector-regression_
fig1_323588842

[27] https://brilliant.org/wiki/naive-bayes-classifier/

[28] http://notesmedicalstudent.blogspot.com/2018/05/
sensitivity-specificity-ppv-npv.html

[29] https://scikit-learn.org/stable/modules/cross_validation.html

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.