ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Εξόρυξη Γνώσης στον Τομέα του Αθλητισμού

Διπλωματική Εργασία του
Βασιλακάκη Αναστάσιου(ΑΕΜ:243)

Επιβλέπων Καθηγητής: Δασκαλοπούλου Ασπασία

ΒΟΛΟΣ
Φεβρουάριος 2011

Ευχαριστίες

Η παρούσα πτυχιακή εργασία με τίτλο “Ανακάλυψη Γνώσης στον τομέα του Αθλητισμού” υλοποιήθηκε στα πλαίσια του προπτυχιακού προγράμματος του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων, το ακαδημαϊκό έτος 2009-2010.

Θα ήθελα να εκφράσω τις αληθινές ευχαριστίες μου προς την επιβλέπουσα καθηγήτρια της πτυχιακής εργασίας κα. Δασκαλοπούλου Ασπασία, για την εμπιστοσύνη που μου έδειξε κατά την ανάθεση της πτυχιακής εργασίας, αλλά και για την αμέριστη συμπαράσταση καθ' όλη τη διάρκεια υλοποίησής της.

Θα ήθελα ακόμα να ευχαριστήσω τους φίλους και συμφοιτητές μου Σπηλιόπουλο Χρήστο, Οικονόμου Θεόδωρο, Άννα Παππά, Κεχαγιά Αθανάσιο, Νότα Ιωάννη, Παπαευθυμίου Κωνσταντίνος για την ηθική και πολλές φορές έμπρακτη βοήθεια τους κατά τη διάρκεια εκπόνησης της παρούσας πτυχιακής εργασίας.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους γονείς μου, για την ηθική και οικονομική βοήθεια που μου προσέφεραν κατά τη διάρκεια των τελευταίων ετών. Η αμέριστη συμπαράστασή τους όλα αυτά τα χρόνια υπήρξε επιπλέον κίνητρο για την ολοκλήρωση των σπουδών μου.

Βασιλακάκης Αναστάσιος

10-02-2011

Πρόλογος

Ο τίτλος της πτυχιακής εργασίας "Ανακάλυψη Γνώσης στο Τομέα του Αθλητισμού" αναφέρεται σε ένα νέο σχετικά επιστημονικό πεδίο της Πληροφορικής. Πρόκειται ουσιαστικά για την εφαρμογή τεχνικών εξόρυξης δεδομένων και ανακάλυψης γνώσης σε δεδομένα του αθλητισμού. Στις σελίδες που ακολουθούν θα αναλυθούν ο επιστημονικός τομέας, το πρόβλημα που ζητήθηκε να αντιμετωπιστεί, πως αυτό προσεγγίστηκε, η εφαρμογή που υλοποιήθηκε στα πλαίσια της εργασίας αλλά και τα αποτελέσματα που αυτή παρείχε.

Πιο συγκεκριμένα στο πρώτο κεφάλαιο της πτυχιακής εργασίας θα γίνει μια σύντομη ανάλυση του επιστημονικού τομέα της ανακάλυψης γνώσης αλλά και των πιθανών προβλημάτων που αυτή μπορεί να συναντήσει.

Στο δεύτερο κεφάλαιο θα αναφερθούμε στο τομέα της Μηχανικής Μάθησης, και ποιες είναι οι σημαντικότερες μέθοδοι του τομέα για την κατασκευή ευφυών συστημάτων. Θα δοθεί ιδιαίτερη έμφαση στις τεχνικές μηχανικής μάθησης με επίβλεψη, Δένδρα Απόφασης και Μάθηση κατά Bayes, οι οποίες χρησιμοποιήθηκαν και στην εφαρμογή που υλοποιήθηκε στα πλαίσια της εργασίας.

Στο τρίτο κεφάλαιο θα οριστεί το πρόβλημα που θα αντιμετωπιστεί και ποιοι είναι οι στόχοι της πτυχιακής. Στην συνέχεια θα γίνει καταγραφή του τρόπου αναπαράστασης του προβλήματος, περιγραφή της αρχιτεκτονικής του συστήματος, περιγραφή της βάσης δεδομένων αλλά και αναλυτική τεχνική ανάλυση του κώδικα της εφαρμογής που υλοποιήθηκε.

Στο τέταρτο κεφάλαιο θα γίνει παρουσίαση της εφαρμογής που υλοποιήθηκε τόσο από τη μεριά του χρήστη, όσο και από την μεριά του διαχειριστή της εφαρμογής. Θα γίνει αναλυτική καταγραφή όλων των λειτουργιών που η εφαρμογή προσφέρει.

Στο πέμπτο κεφάλαιο θα παρουσιαστούν τα αποτελέσματα που επιτεύχθηκαν με την χρήση της εφαρμογής. Θα αναλυθούν και θα σχολιαστούν οι επιδόσεις της εφαρμογής και θα αποτιμηθεί εάν και κατά πόσο επιτεύχθηκαν οι αρχικοί στόχοι της πτυχιακής εργασίας.

Τέλος στο έκτο κεφάλαιο θα γίνει η καταγραφή των μελλοντικών σχεδίων σχετικά με την περαιτέρω ενασχόληση με το συγκεκριμένο πρόβλημα.

Περιεχόμενα

1 Ανακάλυψη Γνώσης σε Βάσεις Δεδομένων

Εισαγωγή

Η ανακάλυψη γνώσεις σε βάσεις δεδομένων(Knowledge Discovery in Databases – KDD) είναι μία σύνθετη διαδικασία για τον προσδιορισμό έγκυρων, νέων, χρήσιμων και κατανοητών σχέσεων-προτύπων σε δεδομένα. Αν και ως όρος είναι σχετικά πρόσφατος, αποτελεί μια σημαντική εφαρμογή σε πραγματικές συνθήκες και σε μεγάλη κλίμακα των ερευνητικών αποτελεσμάτων της μηχανικής μάθησης και στατιστικής.

Η διαδικασία ανακάλυψης γνώσης είναι μία ολοκληρωμένη διαδικασία που περιλαμβάνει την επεξεργασία των δεδομένων, την εφαρμογή των αλγορίθμων ανακάλυψης γνώσης και τέλος την ερμηνεία των αποτελεσμάτων. Χρησιμοποιεί τεχνικές από πολλούς τομείς, όπως στατιστική, μηχανική μάθηση, βάσεις δεδομένων, αναγνώριση προτύπων, πράκτορες, επεξεργασία φυσικής γλώσσας κτλ.

Οι πιο σημαντικές διαφορές μεταξύ ανακάλυψης γνώσης και μηχανικής μάθησης οφείλονται στο γεγονός ότι η πρώτη εφαρμόζεται σε ένα μεγάλο όγκο δεδομένων τα οποία οργανώνονται σε βάσεις δεδομένων που συνήθως έχουν σχεδιαστεί για άλλο σκοπό. Αντίθετα, στη μηχανική μάθηση τα δεδομένα είναι πολύ λιγότερα και προσεκτικά επιλεγμένα, έτσι ώστε να εξυπηρετούν καλύτερα τον εκάστοτε σκοπό.

Σε μία βάση δεδομένων, ένα σύστημα ανακάλυψης γνώσης προσπαθεί να δημιουργήσει μοντέλα ή να ανακαλύψει πρότυπα τα οποία αξιολογούνται συνήθως μέσω κάποιων κριτηρίων ποιότητας. Επειδή όμως η δημιουργία όλων των δυνατών περιγραφών (μοντέλων-προτύπων) είναι υπολογιστικά αδύνατη, το πρόβλημα της ανακάλυψης γνώσης σε βάσεις δεδομένων είναι ουσιαστικά ένα πρόβλημα αναζήτησης ή εύρεσης της καλύτερης περιγραφής από το σύνολο των δυνατών περιγραφών. Η γνώση που προκύπτει υποστηρίζεται από την βάση δεδομένων, χωρίς αυτό όμως να σημαίνει ότι ισχύει οπωσδήποτε και στον πραγματικό κόσμο. Για παράδειγμα, από μία βάση δεδομένων με καταγραμμένα μόνο κόκκινα αυτοκίνητα, θα προκύψει γνώση ότι “όλα τα αυτοκίνητα είναι κόκκινα”. Αυτό, αν και είναι σωστό για την βάση δεδομένων, δεν ισχύει προφανώς στην πραγματικότητα.

Τεχνικές ανακάλυψης γνώσης χρησιμοποιούνται ευρέως σε τομείς όπως οικονομία, ιατρική, marketing, κτλ. Μερικά παραδείγματα εφαρμογής είναι:

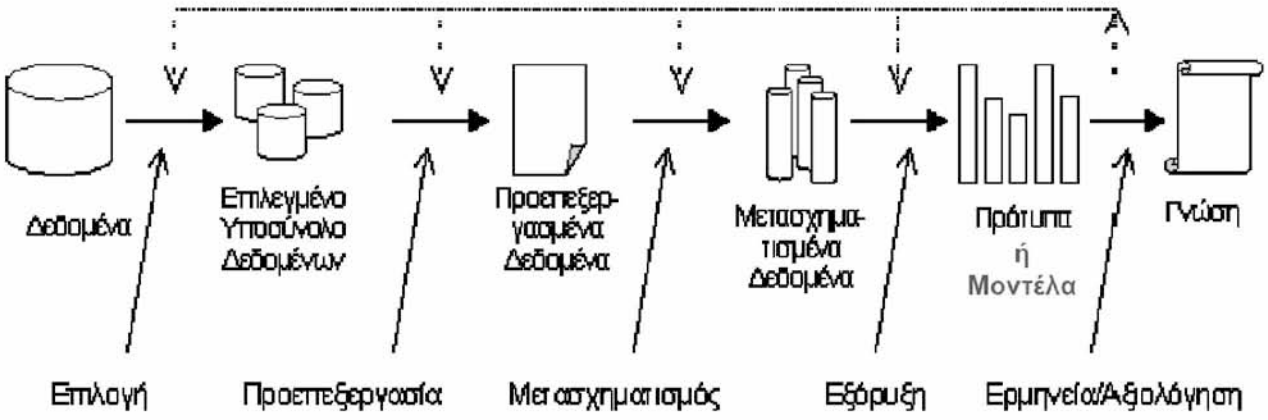
- Εταιρία κινητής τηλεφωνίας θέλει να προβλέψει ποιοι από τους συνδρομητές της δε θα ανανεώσουν τη συνδρομή τους, ώστε να τους κάνει κάποια περισσότερο ελκυστική προσφορά.
- Εταιρία πώλησης ηλεκτρονικών συσκευών θέλει αν μελετήσει τις αγοραστικές συνήθειες των πελατών της, ώστε να προγραμματίσει ανάλογα την επόμενη διαφημιστική της εκστρατεία.
- Ιατρικά εργαστήρια θέλουν να συσχετίσουν ασθένειες με χαρακτηριστικά των ασθενών όπως τόπος διαμονής, διατροφικές συνήθειες, παλιότερες ασθένειες, κτλ, ώστε να οδηγηθούν σε ιατρικές ανακαλύψεις.

Η ανακάλυψη γνώσης σε βάσεις δεδομένων απαιτεί συνήθως τη συνδρομή ενός ειδικού του τομέα, όπου θα εφαρμοστεί η διαδικασία. Πρόκειται για αλληλεπιδραστική και επαναληπτική διαδικασία, στη διάρκεια των επιμέρους σταδίων της οποίας ο ειδικός σε θέματα ανακάλυψης γνώσης καλείται αν πάρει συγκεκριμένες αποφάσεις. Τυπικά μεγέθη του όγκου δεδομένων στα οποία εφαρμόζονται διαδικασίες ανακάλυψης γνώσης είναι εκατομμύρια εγγραφές και δεκάδες Gigabytes.

1.1 Τα στάδια της Ανακάλυψης γνώσης

Η ανακάλυψη γνώσης αρχίζει με την κατανόηση του τομέα στον οποίο θα εφαρμοστεί και τον προσδιορισμό του στόχου της από την σκοπιά του χρήστη των αποτελεσμάτων. Ο ειδικός επί θεμάτων ανακάλυψης γνώσης πρέπει να συνεργαστεί με τον ειδικό του τομέα ώστε το πρόβλημα να καθοριστεί με αρκετή ακρίβεια και να είναι επιλύσιμο, τα αποτελέσματα να είναι μετρήσιμα και να είναι δυνατή η εφαρμογή τους.(για παράδειγμα σε αποδεκτά όρια). Στην συνέχεια περιγράφονται τα επιμέρους στάδια

Η ανακάλυψη γνώσης αρχίζει με την κατανόηση του τομέα στον οποίο θα εφαρμοστεί και τον προσδιορισμό του στόχου της από την σκοπιά του χρήστη των αποτελεσμάτων. Ο ειδικός επί θεμάτων ανακάλυψης γνώσης πρέπει να συνεργαστεί με τον ειδικό του τομέα ώστε το πρόβλημα να καθοριστεί με αρκετή ακρίβεια και να είναι επιλύσιμο, τα αποτελέσματα να είναι μετρήσιμα και να είναι δυνατή η εφαρμογή τους, (για παράδειγμα σε αποδεκτά χρονικά όρια). Στην συνέχεια περιγράφονται τα επιμέρους στάδια της ανακάλυψης γνώσης.



Πρέπει να σημειωθεί ότι τις περισσότερες φορές κάποια από τα επί μέρους βήματα είναι αναγκαίο να επαναληφθούν, καθώς στην πορεία ενδέχεται να προκύψουν προβλήματα που σχετίζονται με τις αρχικές επιλογές και τα οποία δεν είναι δυνατό να εντοπιστούν αρχικά.

1.1.1 Επιλογή

Στο στάδιο της επιλογής δημιουργείται το σύνολο δεδομένων στο οποίο θα εφαρμοστούν οι αλγόριθμοι αποκάλυψης γνώσης. Επειδή τα δεδομένα είναι σχεδόν πάντα οργανωμένα για άλλη χρήση και οι αλγόριθμοι που εκτελούν την ανακάλυψη γνώσης συνήθως δε μπορούν να εφαρμοστούν σε πολλαπλούς πίνακες δεδομένων, απαιτείται η εξαγωγή των δεδομένων από αυτούς και η οργάνωσή τους σε απλούστερες δομές. Συνήθως αυτή η απαίτηση καλύπτεται από συστήματα αποθήκευσης δεδομένων(data warehouse) τα οποία παρέχουν στους αλγορίθμους ανακάλυψης γνώσης μια ευκολότερα προσβάσιμη όψη των δεδομένων.

1.1.2 Προεπεξεργασία

Στο στάδιο της προεπεξεργασίας (preprocessing) των δεδομένων αντιμετωπίζονται περιπτώσεις ελλιπών δεδομένων, για παράδειγμα άδεια πεδία, πεδίων με τιμές που ουσιαστικά τα καθιστούν κενά, για παράδειγμα Οδός= Άγνωστο, πεδίων με τιμές που υπονοούν (κατά σύμβαση) κάτι άλλο, κτλ. Λόγο της φύσης των εργασιών που πραγματοποιούνται το στάδιο αυτό ονομάζεται και στάδιο καθαρισμού δεδομένων (data cleaning).

1.1.3 Μετασχηματισμός

Στο στάδιο του μετασχηματισμού (transformation) τα δεδομένα μετασχηματίζονται ώστε να διευκολύνουν την ανακάλυψη γνώσης. Τέτοιοι μετασχηματισμοί μπορεί να περιλαμβάνουν:

- την ομοιόμορφη κωδικοποίηση της ποιοτικά ίδιας πληροφορίας, για παράδειγμα ενοποίηση ενός πεδίου με τίτλο salary σε ένα πίνακα, με το πεδίο payment σε κάποιον άλλο πίνακα)
- την μείωση του αριθμού των υπό εξέταση χαρακτηριστικών (dimensionality reduction) με επιλογή ορισμένων εξ αυτών (feature selection 'η attribute selection)
- τη διακριτοποίηση (discretization), δηλαδή τη μετατροπή συνεχόμενων αριθμητικών τιμών σε διακριτές, κτλ

Η επιλογή χαρακτηριστικών είναι μία απαραίτητη διαδικασία γιατί συνήθως τα δεδομένα περιλαμβάνουν χαρακτηριστικά τα οποία είναι είτε περιττά στη διαδικασία ανακάλυψης γνώσης (όπως για παράδειγμα τα ονόματα των ασθενών ή των πελατών μιας επιχείρησης) 'η ακατάλληλα. Αν και οι περισσότεροι αλγόριθμοι σχεδιάζονται για να μαθαίνουν ποια είναι τα πιο σημαντικά χαρακτηριστικά για τη λήψη μιας απόφασης, όπως για παράδειγμα στα δέντρα απόφασης, όπου επιλέγεται σε κάθε κόμβο το πιο κατάλληλο χαρακτηριστικό για να γίνει ο διαχωρισμός, εντούτοις η ύπαρξη μεγάλου αριθμών μη σχετικών ή ακατάλληλων χαρακτηριστικών μειώνει σημαντικά την απόδοση των αλγορίθμων. Για παράδειγμα, αποδείχτηκε πειραματικά ότι η απόδοση του αλγορίθμου μάθησης δένδρων ταξινόμησης/απόφασης C4.5, μειώθηκε κατά 5%-10% , όταν προστέθηκε

ένα χαρακτηριστικό με τυχαίες δυαδικές τιμές. Αυτό οφείλεται στο γεγονός ότι σε κάποια φάση της δημιουργίας του δένδρου, αποφασίζεται να γίνει διακλάδωση βάσει του χαρακτηριστικού αυτού, οδηγώντας τελικά σε λανθασμένη ταξινόμηση.

Επιπλέον η μείωση του αριθμού των δεδομένων, αφαιρώντας ορισμένα κατάλληλα χαρακτηριστικά, μπορεί να βελτιώσει την απόδοση των αλγορίθμων όσον αφορά την ταχύτητά τους, αλλά και την ποιότητα της εξαγόμενης γνώσης εστιάζοντας στα πιο σχετικά/σημαντικά χαρακτηριστικά. Η επιλογή χαρακτηριστικών μπορεί να γίνει με αυτόματες μεθόδους, αλλά ο καλύτερος τρόπος είναι να γίνει από τον ειδικό του τομέα, που κατανοεί καλύτερα τη σημασία κάθε χαρακτηριστικού.

Διακριτοποίηση είναι η διαδικασία μετασχηματισμού ενός χαρακτηριστικού που παίρνει (συνεχείς αριθμητικές τιμές, σε ένα πεπερασμένο σύνολο διακριτών διαστημάτων). Η διαδικασία αυτή απαιτείται είτε γιατί ο αλγόριθμος δεν μπορεί να χειριστεί αριθμητικά δεδομένα ή δε τα χειρίζεται σε ικανοποιητικό βαθμό, όπως ορισμένες στατιστικές μέθοδοι ομαδοποίησης και ο απλός ταξινομητής Bayes.

1.1.4 Επιλογή αλγορίθμου και εφαρμογή του

Στο στάδιο αυτό καθορίζεται το είδος της γνώσης που θα αναζητηθεί, κάτι που έμμεσα προσδιορίζει και την κατηγορία αλγορίθμου που θα χρησιμοποιηθεί. Η εφαρμογή του αλγορίθμου είναι ένα καθαρά υπολογιστικό στάδιο, στο οποίο γίνεται η ουσιαστική ανακάλυψη της γνώσης από τα δεδομένα. Περιγράφεται και με τον όρο εξόρυξη σε δεδομένα (data mining), ο οποίος συχνά χρησιμοποιείται καταχρηστικά για να περιγράψει ολόκληρη η διαδικασία ανακάλυψης γνώσης.

Υπάρχουν δύο είδη γνώσης που προκύπτουν ως αποτέλεσμα μιας διαδικασίας ανακάλυψης γνώσης: τα μοντέλα πρόβλεψης (predictive models) και το πρότυπα πληροφόρησης (informative patterns). Πολλές φορές προτιμούνται αλγόριθμοι που έχουν ισχυρά χαρακτηριστικά πληροφόρησης, αν και έχουν χειρότερη δυνατότητα πρόβλεψης, διότι τα αποτελέσματα τους γίνονται πιο εύκολα αντιληπτά από τον ειδικό του τομέα. Οι αλγόριθμοι που χρησιμοποιούνται συχνά προέρχονται από τον τομέα της μηχανικής μάθησης, προσαρμοσμένοι κατάλληλα στις απαιτήσεις της διαδικασίας της ανακάλυψης γνώσης σε βάσεις δεδομένων.

1.1.5 Ερμηνεία και αξιολόγηση

Στο στάδιο αυτό γίνεται ερμηνεία (interpretation) και αξιολόγηση (evaluation) της γνώσης που προκύπτει, πιθανώς με υποβοήθηση γραφικών απεικονίσεων των περιγραφών ή και των δεδομένων που περιγράφονται (visualization). Η γνώση που παράγεται μπορεί να χρησιμοποιηθεί σε ένα σύστημα γνώσης, περίπτωση κατά την οποία ίσως να απαιτείται και η επίλυση πιθανών συγκρούσεων (conflicts) με την υπάρχουσα γνώση.

Αξίζει τέλος να αναφερθεί ότι πολλές φορές, τα αποτελέσματα που προκύπτουν μετά από έναν κύκλο εφαρμογής των παραπάνω βημάτων, δίνουν ερέθισμα για νέου είδους αναζητήσεις, δηλαδή οδηγούν σε επανάληψη ολόκληρης της διαδικασίας με σκοπό την ανακάλυψη νέας γνώσης.

1.2 Προβλήματα στην ανακάλυψη γνώσης

Τα συστήματα ανακάλυψης γνώσης βασίζονται στην παροχή δεδομένων εισόδου από βάσεις δεδομένων οι οποίες τείνουν να είναι δυναμικές, μεγάλου μεγέθους, ελλιπείς και να περιέχουν εσφαλμένα δεδομένα. Επιπλέον προβλήματα προκύπτουν από το πόσο σχετική και επαρκής είναι η αποθηκευμένη πληροφορία. Αναφέρονται στη συνέχεια τα σημαντικότερα προβλήματα που υπεισέρχονται στην ανακάλυψη γνώσης σε βάσεις δεδομένων.

1.2.1 Ακατάλληλα δεδομένα

Οι βάσεις δεδομένων δεν είναι πάντοτε σχεδιασμένες για ανακάλυψη γνώσης και μερικές φορές τα πεδία που θα οδηγούσαν σε σημαντικές ανακαλύψεις όχι μόνο λείπουν, αλλά και δεν είναι δυνατόν να συλλεχθούν από το χρήστη. Για παράδειγμα, δεν είναι δυνατόν να διαγνωστεί ελονοσία σε μία βάση δεδομένων ασθενειών, όταν η βάση δεν περιέχει τον αριθμό των ερυθρών αιμοσφαιρίων τους. Επιπλέον, αν πρόκειται για παλιές καταχωρήσεις, δεν είναι δυνατόν να συλλεχθεί αυτή η πληροφορία.

1.2.2 Ελλιπή δεδομένα

Πολλές φορές η τιμή κάποιων πεδίων απουσιάζει (ελλιπή δεδομένα – missing data). Για παράδειγμα, κάποιο μέγεθος μπορεί να μη μετρήθηκε, κάποια ερώτηση να μην απαντήθηκε, κάποια καταχωρημένη τιμή να διαγράφηκε, κτλ. Ο χειρισμός τέτοιων δεδομένων ποικίλλει και οι μεθοδολογίες που έχουν αναπτυχθεί ενδεικτικά περιλαμβάνουν την απόρριψη των εγγραφών με τα πεδία που δεν έχουν τιμή, τον υπολογισμό των τιμών που λείπουν με βάση άλλες εγγραφές, τον καθορισμό κάποιας ειδικές τιμής που να αντιστοιχεί στις τιμές που δεν υπάρχουν κτλ.

1.2.3 Θόρυβος

Οι βάσεις δεδομένων συχνά περιέχουν λάθη. Πεδία των οποίων η τιμή προέρχεται από υποκειμενικές κρίσεις ή μετρήσεις μπορεί να προκαλέσουν λάθη, όπως για παράδειγμα εσφαλμένη ταξινόμηση ορισμένων περιπτώσεων. Τα λάθη στις τιμές των πεδίων είναι γνωστά ώς θόρυβος (noise – noise data) και είναι σημαντικό να αποβάλλεται, γιατί επηρεάζει τη συνολική ακρίβεια της παραγόμενης γνώσης.

1.2.4 Αραιά δεδομένα

Στην αναζήτηση γνώσης σε βάσεις δεδομένων, ο χώρος αναζήτησης ορίζεται από το δυναμοσύνολο των συνόλων στα οποία ορίζονται τα πεδία. Υπάρχουν περιπτώσεις που για διάφορους λόγους τα διαθέσιμα δεδομένα καλύπτουν μικρό ποσοστό του χώρου αναζήτησης (αραιά δεδομένα – sparse data), με αποτέλεσμα να δημιουργούνται προβλήματα στην αναζήτηση γνώσης. Για παράδειγμα σε ένα πρόβλημα ταξινόμησης τα δεδομένα μπορεί να μην επιτρέπουν τον ακριβή προσδιορισμό των ορίων διαφόρων κατηγοριών.

1.2.5 Μέγεθος βάσης δεδομένων

Η ύπαρξη μεγάλου αριθμού εγγραφών στη βάση δεδομένων κάνει χρονοβόρα την εκτέλεση του αλγορίθμου για την ανακάλυψη της γνώσης και τον έλεγχο της ποιότητας της γνώσης που προκύπτει. Επίσης, η ύπαρξη μεγάλου αριθμού πεδίων ανά εγγραφή, αν και πολλές φορές θεωρείται πλεονέκτημα για την ακρίβεια της παραγόμενης γνώσης, εντούτοις επηρεάζει αρνητικά την απόδοση του αλγορίθμου καθώς και την ποιότητα της παραγόμενης γνώσης, στις περιπτώσεις που περιλαμβάνονται περιττά πεδία. Τέλος , υπάρχει μεγαλύτερη πιθανότητα να δημιουργηθούν πολλές και περίπλοκες περιγραφές, όπως για παράδειγμα ένας τεράστιος αριθμός κανόνων ταξινόμησης, που θα πρέπει να εξεταστούν.

1.2.6 Δείγματα

Η λήψη ενός δείγματος από τη βάση δεδομένων, με χρήση του οποίου θα ανακαλυφθεί γνώση, απαιτεί μεγάλη προσοχή και εφαρμογή στατιστικών τεχνικών, ώστε να αντιπροσωπεύει την αρχική βάση και να περιλαμβάνει χρήσιμα δεδομένα. Η δειγματοληψία αποτελεί συνήθη πρακτική, καθώς και οι πραγματικές βάσεις δεδομένων έχουν τεράστιο αριθμό εγγραφών ενώ ορισμένοι αλγόριθμοι ανακάλυψης γνώσης, για λόγους απόδοσης, απαιτούν τα δεδομένα εκπαίδευσης να είναι όλα στη μνήμη. Αυτό όμως δεν είναι εφικτό λόγω του περιορισμένου μεγέθους της μνήμης.

1.2.7 Πρόσφατα δεδομένα

Επειδή οι βάσεις δεδομένων είναι δυναμικές, προκύπτει το πρόβλημα του κατά πόσο μπορεί να θεωρηθεί ότι η γνώση που παρήχθει κάποτε, ανταποκρίνεται στην πλέον ενημερωμένη και πρόσφατη έκδοση της βάσης δεδομένων. Αυτό απαιτεί έναν περιοδικό έλεγχο για την εγκυρότητα των αποτελεσμάτων. Επιπλέον δημιουργείται η ανάγκη ύπαρξης αλγορίθμων σταδιακής εξόρυξης σε δεδομένα (incremental data mining) που θα λαμβάνουν υπόψη τους την πληροφορία νέων εγγραφών καθώς αυτές θα γίνονται διαθέσιμες, χωρίς να απαιτείται η επανάληψη της διαδικασίας από την αρχή για το σύνολο των εγγραφών.

1.3 Ειδικά Θέματα στην Ανακάλυψη Γνώσης

Υπάρχουν κάποιες ειδικές περιπτώσεις δεδομένων, όπως για παράδειγμα τα δεδομένα που είναι διάσπαρτα σε κατανεμημένες βάσεις δεδομένων σύνθετοι τύποι δεδομένων, που δεν αντιμετωπίζονται με τις κλασικές μεθόδους αποκάλυψης γνώσης, αλλά απαιτούν ειδικές τεχνικές, μερικές από τις οποίες εξετάζονται στη συνέχεια.

1.3.1 Σύνθετοι τύποι δεδομένων

Τις τελευταίες δύο δεκαετίες η πρόοδος των νέων τεχνολογιών οδήγησε στη δημιουργία νέων, σύνθετων τύπων δεδομένων, όπως τα χωρικά, χρονικά και τα δεδομένα πολυμέσων. Στη τελευταία κατηγορία ανήκουν τα δεδομένα εικόνων, ήχου, βίντεο και συνδυασμού αυτών με κείμενο, τα οποία χαρακτηρίζονται ως ημιδομημένα ή αδόμητα δεδομένα. Χαρακτηριστικότερο τέτοιο παράδειγμα αποτελούν οι ιστοσελίδες του παγκόσμιου ιστού. Τέτοιοι τύποι δεδομένων έχουν ξεχωριστές απαιτήσεις κατά την αναπαράσταση, αποθήκευση και επεξεργασία τους και οι παραδοσιακές σχεσιακές βάσεις δεδομένων αδυνατούν να αντεπεξέλθουν. Αυτό είχε ως αποτέλεσμα να δοθεί ώθηση στην έρευνα και στην ανάπτυξη νέων τύπων συστημάτων διαχείρισης βάσεων δεδομένων, όπως πολυμεσικές, οι χρονικές και οι αντικειμενοστραφείς, 'η υβριδικά συστήματα, όπως οι αντικειμενο-σχεσικές βάσεις δεδομένων (object-relational databases). Φυσιολογικά, δημιουργήθηκε και η ανάγκη για εργαλεία ικανά να ανακαλύπτουν γνώση σε τέτοιου είδους δεδομένα, καθώς οι κλασικές στατιστικές μέθοδοι και οι παραδοσιακοί αλγόριθμοι αναζήτησης γνώσης δεν μπορούσαν να αποδώσουν ικανοποιητικά. Σαν αποτέλεσμα, αναπτύχθηκαν νέες τεχνικές καθαρισμού, επιλογής, μετασχηματισμού και μάθησης.

1.3.2 Κατανεμημένα δεδομένα

Τα τελευταία χρόνια παρατηρείται ραγδαία εξέλιξη στην τεχνολογία των δικτύων κάτι που έχει οδηγήσει στην εμφάνιση γεωγραφικά κατανεμημένων δεδομένων. Σε αυτές τις περιπτώσεις, οι υπάρχουσες τεχνικές εξόρυξης σε δεδομένα είναι ανεπαρκείς ή αναποτελεσματικές, γιατί είναι κατασκευασμένες να λειτουργούν με ένα μόνο, τοπικά εντοπισμένο, σύνολο δεδομένων. Υπάρχουν όμως αρκετές εφαρμογές στις οποίες η πληροφορία είναι αποθηκευμένη σε διάφορους υπολογιστές (για παράδειγμα σε πολυεθνικές εταιρίες ή αλυσίδες καταστημάτων), οπότε αυτοί θα πρέπει να “συνεργαστούν” μεταξύ τους για εξάγουν καθολική γνώση.

Η προφανής λύση είναι η συσσώρευση όλων των δεδομένων που βρίσκονται στους υπολογιστές σε ένα κεντρικό σταθμό εργασίας, στον οποίο θα γίνεται η εξόρυξη. Κάτι τέτοιο βέβαια, δεν είναι εφικτό ή επιθυμητό για πολλούς λόγους :

- Το υπολογιστικό σύστημα που θα φιλοξενήσει το σύνολο των δεδομένων πρέπει να έχει πολύ μεγάλη χωρητικότητα και αυξημένη υπολογιστική ισχύ.
- Το κόστος μεταφοράς και αποθήκευσης των δεδομένων ίσως είναι

απαγορευτικά μεγάλο.

- Μπορεί να παρατηρηθεί πρόβλημα συγχρονισμού σε περιπτώσεις κατά τις οποίες μια "τοπική" βάση δεδομένων ενημερώνεται, οπότε καθίσταται απαραίτητη η ταυτόχρονη ενημέρωση και των δεδομένων στο κεντρικό Η/Υ.
- Λόγοι ασφαλείας και αυτονόμησης μπορεί να απαγορεύουν την μετακίνηση των τοπικών δεδομένων, όπως για παράδειγμα συμβαίνει με τα τραπεζικά δεδομένα των οποίων η "αποκάλυψη" δεν είναι επιθυμητή.

Σε περιπτώσεις σαν τις παραπάνω, απαιτείται η χρήση κατανεμημένης εξόρυξης σε δεδομένα (distributed data mining), προκειμένου να γίνει καλύτερη αξιοποίηση των δεδομένων και των διαθέσιμων υπολογιστικών πόρων.

2 Μηχανική Μάθηση

Όπως αναφέρθηκε στο πρώτο κεφάλαιο, η μηχανική μάθηση χρησιμοποιείται κατά την διαδικασία ανακάλυψης γνώσης σε βάσεις δεδομένων. Η ερευνητική περιοχή της μηχανικής μάθησης ασχολείται με υπολογιστικές μεθόδους για την κατασκευή συστημάτων με ικανότητα μάθησης. Τι εννοούμε όμως με τον όρο μάθηση; Έχουν προταθεί διάφοροι ορισμοί, όπως:

- Η μάθηση σηματοδοτεί αλλαγές σε ένα σύστημα που είναι προσαρμοστικές με την έννοια ότι επιτρέπουν στο σύστημα να κάνει την ίδια την εργασία, ή εργασίες της ίδιας κατηγορίας, πιο αποδοτικά και αποτελεσματικά από την επόμενη φορά.
- Μάθηση είναι η δημιουργία ή αλλαγή αναπαραστάσεων των εμπειριών που βιώνουμε.

Θα μπορούσαμε να πούμε ότι η παραπάνω ορισμοί είναι συμπληρωματικοί, γιατί μαζί καλύπτουν δύο βασικές ικανότητες του ανθρώπου που σχετίζονται με τη μάθηση, όπως αυτές γίνονται αντιληπτές στην καθημερινή ζωή :

1. βελτίωση της αποτελεσματικότητας στην εκτέλεση μιας εργασίας με την εμπειρία
2. την απόκτηση νέας γνώσης ή την οργάνωση της ήδη υπάρχουσας.

Υπάρχει πολύ και πολυποίκιλη ερευνητική δουλειά στη περιοχή της μηχανικής μάθησης. Έχουν γίνει διάφορες προσπάθειες ταξινόμησης αυτής της δουλειάς βάσει διαφορετικών κριτηρίων. Εδώ η παρουσίαση των διαφορετικών μεθόδων μηχανικής μάθησης οργανώνεται με βάση το πρόβλημα που αντιμετωπίζουν. Τα βασικά προβλήματα είναι :

- Η προσέγγιση μιας συνάρτησης από τα δεδομένα με παρατηρήσεις τιμών εισόδου και εξόδου της. Αν η συνάρτηση έχει έξοδο ως διακριτές τιμές, τότε το πρόβλημα ονομάζεται ταξινόμηση ή κατηγοριοποίηση (classification), ενώ αν έχει συνεχείς τιμές ονομάζεται παρεμβολή (regression).
- Η εύρεση φυσικών οργανώσεων των δεδομένων σε ομάδες, έτσι ώστε δεδομένα της ίδιας ομάδας να μοιάζουν όσο το δυνατόν περισσότερο και δεδομένα διαφορετικών ομάδων να διαφέρουν όσο το δυνατό περισσότερο. Το πρόβλημα αυτό ονομάζεται ομαδοποίηση (clustering).
- Η εύρεση κανόνων συσχέτισης μεταξύ αντικειμένων σε συναλλακτικές (transactional) βάσεις δεδομένων. Το πρόβλημα αυτό ονομάζεται εξόρυξη κανόνων συσχέτισης (association rule mining) και προέκυψε τη δεκαετία του '90 από τον τομέα της ανάλυσης καλαθιών αγορών.
- Η εύρεση της βέλτιστης συμπεριφοράς ενός πράκτορα με βάση την ανταμοιβή που παίρνει σε μια τελική κατάσταση σε κάποιο περιβάλλον έχοντας ξεκινήσει από μια αρχική κατάσταση στο ίδιο περιβάλλον και ακολουθώντας μια σειρά από ενέργειες και ενδιάμεσες καταστάσεις. Το πρόβλημα αυτό

ονομάζεται ενισχυτική μάθηση (reinforcement learning).

Στη μηχανική μάθηση υπήρχαν παραδοσιακά οι όροι μάθηση με επίβλεψη (supervised learning) και μάθηση χωρίς επίβλεψη (unsupervised learning). Η μάθηση με επίβλεψη ταυτίζεται με την πρώτη κατηγορία προβλημάτων , δηλαδή της ταξινόμησης και παρεμβολής. Το όνομα προέρχεται από το γεγονός ότι σε αυτά τα προβλήματα υπάρχει κάποιος "επιβλέπων", ο οποίος μας παρέχει την τιμή εξόδου της συνάρτησης για τα δεδομένα που εξετάζουμε. Η μάθηση χωρίς επίβλεψη ταυτιζόταν παραδοσιακά με το πρόβλημα της ομαδοποίησης. Ο λόγος είναι ότι στην ομαδοποίηση δεν υπάρχει κάποιος "επιβλέπων", αφού δεν γνωρίζουμε πόσες, ποιες και αν υπάρχουν ομάδες.

Η εξόρυξη κανόνων συσχέτισης εμφανίστηκε αρκετά αργότερα από τη μηχανική μάθηση, και είχε περισσότερες επιρροές από την ερευνητική περιοχή των βάσεων δεδομένων. Ωστόσο θα μπορούσαμε να την εντάξουμε στη μάθηση χωρίς επίβλεψη, αφού και πάλι δεν γνωρίζουμε εκ των προτέρων αν υπάρχουν κάποιες συσχετίσεις στα δεδομένα και ποιες είναι αυτές.

Τέλος, η ενισχυτική μάθηση έχει επιρροές και από τα δύο είδη μάθησης. Όπως στη μάθηση με επίβλεψη, υπάρχει κάποιος εξωτερικός παράγων (το περιβάλλον), που δίνει μια αριθμητική ανταμοιβή στον πράκτορα για κάθε ενέργεια του. Ωστόσο, η συμπεριφορά του περιβάλλοντος είναι άγνωστη στον πράκτορα, και πρέπει να την ανακαλύψει μέσω δοκιμής και αποτυχίας, κάτι που θυμίζει την μάθηση χωρίς επίβλεψη/

Ένας άλλος διαχωρισμός των μεθόδων μάθησης που συναντάται στη διεθνή βιβλιογραφία είναι αυτός σε περιγραφική μοντελοποίηση (descriptive modeling) και μοντελοποίηση πρόβλεψης (predictive modeling). Η περιγραφική μοντελοποίηση περιλαμβάνει την εξόρυξη κανόνων συσχέτισης και την ομαδοποίηση επειδή η γνώση που προκύπτει έχει την ικανότητα της περιγραφής των δεδομένων. Στη μοντελοποίηση πρόβλεψης ανήκουν η ταξινόμηση και η παρεμβολή, επειδή η γνώση που προκύπτει μπορεί να χρησιμοποιηθεί για πρόβλεψη. Ωστόσο ορισμένες μέθοδοι ταξινόμησης (π.χ. Κανόνες ταξινόμησης, δένδρα απόφασης) έχουν και περιγραφικό χαρακτήρα, οπότε αυτός ο διαχωρισμός δεν είναι κάθετος. Το ίδιο ισχύει και για την ενισχυτική μάθηση, όπου η γνώση που αποκτάται μπορεί να χρησιμοποιηθεί τόσο για την πρόβλεψη της πιο κατάλληλης ενέργειας του πράκτορα για μια κατάσταση όσο και την περιγραφή του περιβάλλοντος στο οποίο δραστηριοποιείται ο πράκτορας.

Κάποιοι άλλοι όροι που εμφανίζονται στη διεθνή βιβλιογραφία είναι το πρότυπο (pattern) και το μοντέλο(model). Ο όρος πρότυπο έχει την έννοια της γνώσης που αφορά ένα τοπικό κομμάτι των δεδομένων. Για παράδειγμα, πρότυπο αποτελεί ένας κανόνας συσχέτισης, ένας κανόνας ταξινόμησης ή μια ομάδα. Αντίθετα το μοντέλο έχει την έννοια της γνώσης που αφορά σε όλα τα δεδομένα. Για παράδειγμα, μοντέλο αποτελεί ένα σύνολο κανόνων ταξινόμησης, ένα δένδρο απόφασης, ένα νευρωνικό δίκτυο, ή μία γραμμική συνάρτηση.

2.1 Ταξινόμηση και παρεμβολή

Στη μάθηση με επίβλεψη το πρόβλημα που αντιμετωπίζεται είναι η εύρεση μιας συνάρτησης μέσα από ένα σύνολο δεδομένων με ζευγάρια τιμών της εισόδου και εξόδου της. Η συνάρτηση αυτή ονομάζεται συνάρτηση στόχος (target function) και συμβολίζεται συνήθως με "c". Η συνάρτηση στόχος χρησιμοποιείται για την πρόβλεψη της τιμής μιας εξαρτημένης μεταβλητής που ονομάζεται μεταβλητή στόχος με βάση τις τιμές ενός συνόλου από ανεξάρτητες μεταβλητές που ονομάζονται χαρακτηριστικά (attributes ή features). Το σύνολο των διαφορετικών δυνατών τιμών εισόδου της συνάρτησης στόχος, δηλαδή το πεδίο ορισμού της, ονομάζεται σύνολο των περιπτώσεων ή στιγμιοτύπων (instances), και συμβολίζεται με X.

Ένα σύστημα μάθησης με επίβλεψη, κατά την εκπαίδευσή του, έχει ως είσοδο ένα υποσύνολο του συνόλου των περιπτώσεων, για το οποίο είναι γνωστή η τιμή της μεταβλητής εξόδου και το οποίο ονομάζεται σύνολο εκπαίδευσης και συμβολίζεται με D. Ένα στοιχείο του συνόλου εκπαίδευσης ονομάζεται παράδειγμα (example). Σκοπός του συστήματος είναι να προσεγγίσει όσο καλύτερα μπορεί τη συνάρτηση στόχος δοθέντος ενός συνόλου εκπαίδευσης. Για να το κάνει αυτό εξετάζει διαφορετικές συναρτήσεις, οι οποίες ονομάζονται υποθέσεις (hypotheses), και συμβολίζονται με h. Το σύνολο όλων των δυνατών διαφορετικών υποθέσεων που το σύστημα μάθησης ενδέχεται να εξετάσει ονομάζεται σύνολο των υποθέσεων, και συμβολίζεται με H.

Η πιο απλή λύση στο πρόβλημα της μάθησης με επίβλεψη είναι η μάθηση με απομνημόνευση(learning by memorization, rote learning). Το σύστημα μάθησης απλά αποθηκεύει το σύνολο των δεδομένων και μπορεί να δώσει την τιμή της μεταβλητής εξόδου για μια νέα περίπτωση μόνο αν αυτή η ίδια είναι ήδη αποθηκευμένη. Είναι ευνόητο ότι η μάθηση με απομνημόνευση δεν είναι αποτελεσματική όταν το σύνολο των δεδομένων περιλαμβάνει ένα μικρό υποσύνολο του συνόλου των περιπτώσεων, όπως συμβαίνει στη πράξη.

Στη πράξη απαιτείται η προσέγγιση της συνάρτησης στόχος μέσω μιας διαδικασίας επαγωγής. Το σύστημα μάθησης εξετάζοντας μόνο ένα μέρος του συνόλου των περιπτώσεων (το σύνολο εκπαίδευσης) καλείται να επάγει μια συνάρτηση που θα ισχύει για όλο το σύνολο των περιπτώσεων. Αυτή η λύση στο πρόβλημα της μάθησης στηρίζεται σε μία υπόθεση που είναι γνωστή ως *υπόθεση της επαγωγικής μάθησης:*

- Κάθε συνάρτηση που έχει βρεθεί να προσεγγίζει τη συνάρτηση στόχο καλά για ένα αρκετά μεγάλο σύνολο εκπαίδευσης, θα προσεγγίζει το ίδιο καλά την συνάρτηση στόχο και για άλλες περιπτώσεις που δεν έχει εξετάσει.

Για να μπορέσει να γενικεύσει ένα σύστημα μάθησης πέρα από τα δεδομένα εκπαίδευσης θα πρέπει να κάνει κάποιες παραδοχές σε σχέση με την συνάρτηση στόχο. Αυτές οι παραδοχές αποτελούν την επαγωγική μεροληψία (inductive bias) του συστήματος μάθησης. Υπάρχουν δύο είδη μεροληψίας. Στη *μεροληψία περιορισμού* το σύστημα μάθησης κάνει κάποια παραδοχή για το χώρο υποθέσεων που θα ψάξει, δηλαδή για την μορφή της συνάρτησης στόχος, έτσι ώστε να περιορίσει αυτό τον χώρο. Στη *μεροληψία προτίμησης* το σύστημα μάθησης κάνει κάποια παραδοχή σχετικά με τις υποθέσεις έτσι ώστε να μπορεί να προτιμά κάποιες από αυτές έναντι κάποιων άλλων, έτσι ώστε να μην κάνει εξαντλητική ανάλυση όλων των υποθέσεων.

Όπως αναφέρθηκε στην αρχή του κεφαλαίου, τα προβλήματα μάθησης με επίβλεψη διακρίνονται σε δύο είδη ανάλογα με την τιμή της μεταβλητής εξόδου, στα προβλήματα ταξινόμησης (classification) και στα προβλήματα παρεμβολής (regression). Στα προβλήματα ταξινόμησης ή κατηγοριοποίησης η τιμή εξόδου παίρνει διακριτές τιμές (π.χ. Γράμματα της αλφαβήτου). Τα προβλήματα παρεμβολής αφορούν τη μάθηση μιας συνάρτησης στόχου με πεδίο τιμών τους πραγματικούς αριθμούς, δηλαδή προβλήματα όπου η μεταβλητή εξόδου παίρνει συνεχόμενες τιμές (π.χ. Τιμή μετοχής).

Οι κυριότερες τεχνικές μηχανικής μάθησης με επίβλεψη είναι:

- Δένδρα Απόφασης (Decision Trees)
- Μάθηση κατά Bayes (Bayesian Learning)
- Μάθηση Κανόνων (Rule Learning)
- Μάθηση με Βάση τις Περιπτώσεις (Instance-Based Learning)
- Νευρωνικά δίκτυα (Neural Networks)
- Μηχανές διανυσμάτων υποστήριξης ()Support Vector Machines)

Στην συνέχεια θα περιγράψουμε αναλυτικά τις μεθόδους Δένδρα απόφασης και Μάθηση κατά Bayes οι οποίες χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής.

2.1.1 Δένδρα Απόφασης

Η μάθηση δένδρων απόφασης είναι ένα από τα πιο δημοφιλή είδη μάθησης και έχει εφαρμοστεί αποτελεσματικά σε διάφορους τομείς, όπως διάγνωση ιατρικών περιστατικών, αξιολόγηση ρίσκου αποδοχής αίτησης για πιστωτική κάρτα και εύρεση αιτίας δυσλειτουργίας μηχανημάτων.

Το χαρακτηριστικό σε αυτό το είδος μάθησης είναι ότι το μοντέλο που προκύπτει αναπαριστάται ως δένδρο, που ονομάζεται δένδρο απόφασης (decision tree). Κάθε κόμβος σε ένα δένδρο απόφασης ορίζει μια συνθήκη ελέγχου της τιμής κάποιου χαρακτηριστικού αυτού αν είναι διακριτό (πχ. Φύλο = άνδρας), ή σε ένα περιορισμό του εύρους τιμών από τη ρίζα και ακολουθώντας τα κλαδιά του δένδρου προς κάποιο φύλλο, το οποίο περιέχει μια διακριτή τιμή της μεταβλητής εξόδου. Σε κάθε κόμβο ελέγχεται η τιμή της περίπτωσης για το χαρακτηριστικό του κόμβου και ακολουθείται το αντίστοιχο κλαδί.

Στο σχήμα 2.1 βλέπουμε ένα παράδειγμα δένδρου απόφασης για την συνάρτηση στόχο “Καλή μέρα για τένις”

Σχήμα 2.1: Δένδρο απόφασης για τη συνάρτηση στόχο "καλή μέρα για τένις"

Ουσιαστικά η αναπαράσταση που χρησιμοποιείται από τα δένδρα απόφασης είναι διάζευξη που αποτελείται από συζεύξεις περιορισμών στις τιμές των χαρακτηριστικών. Κάθε μονοπάτι από τη ρίζα προς ένα φύλλο αντιστοιχεί σε συζεύξεις περιορισμών στις τιμές των χαρακτηριστικών. Το δένδρο συνολικά εκφράζει τη διάζευξη των συζεύξεων, αφού αποτελείται από όλα τα εναλλακτικά μονοπάτια.

Γενικά η μάθηση δένδρων απόφασης είναι κατάλληλη για προβλήματα όπου:

- Η συνάρτηση στόχος παίρνει διακριτές τιμές εξόδου, δηλαδή προβλήματα ταξινόμησης. Με κάποιες σημαντικές επεκτάσεις στη βασική μεθοδολογία, μπορεί να χρησιμοποιηθεί και για προβλήματα παρεμβολής, αλλά δεν συνηθίζεται.
- Οι περιπτώσεις αποτελούνται από ζευγάρια χαρακτηριστικών και τιμών. Η απλούστερη περίπτωση είναι οι τιμές να είναι διακριτές. Ωστόσο υπάρχουν επεκτάσεις της βασικής μεθοδολογίας και για πραγματικές τιμές.
- Η έννοια στόχος είναι πιθανό να αποτελείται από διαζεύξεις συζεύξεων από περιορισμούς σε τιμές των χαρακτηριστικών.
- Τα δεδομένα εκπαίδευσης είναι πιθανό να περιέχουν σφάλματα.
- Είναι πιθανό κάποιες τιμές χαρακτηριστικών στα δεδομένα εκπαίδευσης να λείπουν.

Κατασκευή δένδρων απόφασης

Η πλειονότητα των αλγορίθμων μάθησης δένδρων απόφασης κατασκευάζουν το δένδρο άπληστα από πάνω προς τα κάτω. Αρχικά πρέπει να επιλεγεί το πιο κατάλληλο χαρακτηριστικό για έλεγχο στη ρίζα. Η επιλογή στηρίζεται σε κάποιο στατιστικό μέτρο που υπολογίζεται από τα δεδομένα. Θα αναφερθούν διάφορα τέτοια μέτρα στη συνέχεια. Έπειτα, για κάθε δυνατή τιμή του χαρακτηριστικού δημιουργούνται οι αντίστοιχοι απόγονοι της ρίζας και τα δεδομένα μοιράζονται στους νέους κόμβους ανάλογα με την τιμή που έχουν για το χαρακτηριστικό που ελέγχεται στη ρίζα.

Η όλη διαδικασία επαναλαμβάνεται για κάθε νέο κόμβο. Το μέτρο επιλογής του πιο κατάλληλου χαρακτηριστικού για έλεγχο σε ένα νέο κόμβο θα υπολογιστεί και πάλι, αυτή τη φορά όμως χρησιμοποιώντας μόνο τα δεδομένα που ανήκουν σε αυτόν το κόμβο.

Ένας κόμβος γίνεται φύλλο όταν όλα τα δεδομένα που ανήκουν σε αυτόν ανήκουν στην ίδια κατηγορία. Η κατηγορία αυτή γίνεται και η τιμή του φύλλου. Αν σε κάποιο βάθος τελειώσουν τα χαρακτηριστικά προς έλεγχο, τότε ο κόμβος γίνεται φύλλο και σαν τιμή παίρνει εκείνη που έχει την πλειοψηφία με βάση τα δεδομένα του κόμβου αυτού.

Μέτρα επιλογής χαρακτηριστικού

Το σημαντικό σημείο ενός αλγορίθμου μάθησης δένδρων απόφασης είναι η επιλογή του χαρακτηριστικού που θα ελέγχεται σε κάθε κόμβο για το σκοπό αυτό έχουν προταθεί διάφορα στατιστικά μέτρα, με βάση τα οποία αξιολογούνται τα χαρακτηριστικά και επιλέγεται το καλύτερο.

Ένας από τους πρώτους αλγορίθμους, ο ID3, χρησιμοποιεί το μέτρο κέρδος πληροφορίας (information gain), το οποίο στηρίζεται στην έννοια της εντροπίας (entropy).

Η εντροπία χαρακτηρίζει την ανομοιογένεια ενός συνόλου παραδειγμάτων και εκφράζει το αναμενόμενο μήκος κωδικοποίησης πληροφορίας μετρημένο σε bit. Έστω ένα σύνολο παραδειγμάτων S. Αν Pc είναι το ποσοστό των παραδειγμάτων του S που ανήκουν

$$E(S) = \sum_{c \in C} -\mathrm{p}_c \log 2 \mathrm{p}_c$$

στην κατηγορία c ε C, τότε η εντροπία E(S) του συνόλου είναι :

Το κέρδος πληροφορίας μετράει τη μείωση της εντροπίας που θα προκληθεί αν χωριστούν τα παραδείγματα ενός συνόλου με βάση κάποιο χαρακτηριστικό. Έστω λοιπόν ένα σύνολο παραδειγμάτων S και ένα χαρακτηριστικό A με σύνολο τιμών V(A). Το κέρδος πληροφορίας IG(S,A) σε σχέση με αυτό το χαρακτηριστικό είναι :

$$\mathrm{IG(S,A)=E(S)} - \sum_{u \in V(A)} \frac{|S_u|}{|S|} \mathrm{E}(S_u)$$

Ο πρώτος όρος της διαφοράς είναι η εντροπία του συνόλου των παραδειγμάτων. Ο δεύτερος όρος είναι η εντροπία των παραδειγμάτων μετά το διαχωρισμό τους ανάλογα με την τιμή του χαρακτηριστικού A. Αποτελείται από το άθροισμα της εντροπίας για το κάθε σύνολο Su που προκύπτει μετά το διαχωρισμό του S, ανάλογα με την τιμή u του χαρακτηριστικού A. Το χαρακτηριστικό με το μεγαλύτερο κέρδος πληροφορίας είναι αυτό που επιλέγεται από τον αλγόριθμο ID3 για έλεγχο σε κάθε κόμβο.

Ένα πρόβλημα του κέρδους πληροφορίας είναι ότι επιλέγει με μεγαλύτερη

πιθανότητα χαρακτηριστικά που έχουν πολλές διακριτές τιμές. Μια λύση σε αυτό το πρόβλημα είναι η χρήση ενός διαφορετικού μέτρου που θα μειώνει τη πιθανότητα επιλογής ενός χαρακτηριστικού ανάλογα με τον αριθμό των τιμών του. Ένα τέτοιο μέτρο είναι ο λόγος κέρδους (gain ratio), το οποίο στηρίζεται στην έννοια της πληροφορίας διαχωρισμού (split information).

Η πληροφορία διαχωρισμού είναι ένα μέτρο ευαίσθητο στο εύρος και την ομοιομορφία διαχωρισμού των δεδομένων από ένα χαρακτηριστικό. Έστω ένα σύνολο παραδειγμάτων S και ένα χαρακτηριστικό A με ένα σύνολο τιμών V(A). Η πληροφορία διαχωρισμού SI(S,A) είναι :

$$SI(S,A) = - \sum_{u \in V(A)} \frac{|S_u|}{|S|} \log_2 \frac{|S_u|}{|S|}$$

Όπου Su είναι το σύνολο που προκύπτει από το διαχωρισμό του S ανάλογα με την τιμή u του χαρακτηριστικού A. Ας σημειωθεί ότι η πληροφορία διαχωρισμού ενός συνόλου με βάση κάποιο χαρακτηριστικό A ταυτίζεται με την εντροπία του αν αυτό το χαρακτηριστικό ήταν η εξαρτημένη μεταβλητή. Ο λόγος κέρδους GR(S,A) ορίζεται με βάση το κέρδος πληροφορίας IG(S,A) και την πληροφορία διαχωρισμού SI(S,A) ως εξής :

$$GR(S,A) = \frac{IG(S,A)}{SI(S,A)}$$

Ο αλγόριθμος C4.5 χρησιμοποιεί εξ' ορισμού το λόγο κέρδους, αλλά υποστηρίζει και το κέρδος πληροφορίας. Έχουν προταθεί κι άλλα στατιστικά μέτρα στη βιβλιογραφία της μάθησης δένδρων απόφασης.

Κατάλληλο μέγεθος δένδρου

Οι αλγόριθμοι κατασκευής δένδρων απόφασης στηρίζουν την επέκταση του κάθε κόμβου αποκλειστικά και μόνο στα δεδομένα εκπαίδευσης που υπάρχουν σε αυτόν το κόμβο. Αυτό όμως μπορεί να οδηγήσει σε δένδρα που έχουν μεγάλη ακρίβεια πρόβλεψης στα δεδομένα εκπαίδευσης αλλά μικρότερη σε νέα άγνωστα δεδομένα.

Αυτό το πρακτικό πρόβλημα εμφανίζεται τόσο στη μάθηση δένδρων απόφασης όσο και σε άλλες μεθόδους μάθησης και ονομάζεται υπερμοντελοποίηση (overfitting): Σε ένα χώρο υποθέσεων H , μια υπόθεση h ε H υπερμοντελοποιεί τα δεδομένα αν υπάρχει μια άλλη υπόθεση h' ε H με μεγαλύτερο σφάλμα από την h στα δεδομένα εκπαίδευσης, αλλά μικρότερο σε όλο το σύνολο των περιπτώσεων.

Στους αλγόριθμους μάθησης δένδρων απόφασης, ο κίνδυνος της υπερμοντελοποίησης ελλοχεύει όταν :

- Υπάρχει θόρυβος στα δεδομένα εκπαίδευσης, δηλαδή λάθη είτε στις τιμές των χαρακτηριστικών είτε ακόμα χειρότερα στην τιμή του χαρακτηριστικού πρόβλεψης.
- Τα δεδομένα αυτά δεν αποτελούν ένα αντιπροσωπευτικό δείγμα της έννοιας στόχου. Αν πολύ λίγα δεδομένα σχετίζονται με ένα κόμβο του δένδρου τότε μπορεί κατά τύχη ένα χαρακτηριστικό που είναι άσχετο με την έννοια στόχο να διαχωρίζει πολύ καλά τα δεδομένα αυτά.

Η πιο συνηθισμένη λύση στο πρόβλημα της εύρεσης του πιο κατάλληλου μεγέθους για τα δένδρα απόφασης και την αποφυγή με αυτόν τον τρόπο της υπερμοντελοποίησης είναι το κλάδεμα (pruning) : Το δένδρο μεγαλώνει πλήρως και μετά κλαδεύεται με διάφορους τρόπους. Το κλάδεμα ενός κόμβου του δένδρου συνιστάται στην αφαίρεση του υποδένδρου που κρέμεται από αυτόν και στη μετατροπή του σε τερματικό κόμβο με τιμή για την εξαρτημένη μεταβλητή την πιο κοινή στα δεδομένα που ανήκουν στον κόμβο.

Μια κατηγορία μεθόδων κλαδέματος στηρίζεται στη δέσμευση ενός υποσυνόλου των δεδομένων εκπαίδευσης, το σύνολο κλαδέματος (pruning set), για το σκοπό της αξιολόγησης των εναλλακτικών δένδρων. Μία γνωστή μέθοδος αυτής της κατηγορίας ονομάζεται cost-complexity pruning. Αυτή η μέθοδος λειτουργεί σε 2 φάσεις. Στην, αναπτύσσεται ένα σύνολο από δένδρα T0,T1,...,Tk, ξεκινώντας από το πλήρως δένδρο T0 και προχωρώντας σε όλο και μικρότερα κλαδεύοντας κάθε φορά έναν ή παραπάνω κόμβους. Το τελευταίο δένδρο Tκ αποτελείται μόνο από τη ρίζα. Στη δεύτερη φάση ένα από τα δένδρα αυτά επιλέγεται με βάση την ακρίβεια πρόβλεψης τους στο σύνολο κλαδέματος.

Μια άλλη γνωστή μέθοδος κλαδέματος της ίδιας κατηγορίας ονομάζεται reduce error pruning. Αυτή θεωρεί κάθε μη τερματικό κόμβο του δένδρου ως υποψήφιο για κλάδεμα. Κάθε φορά επιλέγεται για κλάδεμα ο κόμβος που αυξάνει περισσότερο την ακρίβεια του δένδρου με βάση το σύνολο κλαδέματος. Όπως και η προηγουμένη μέθοδος κατασκευάζει μια σειρά από δένδρα, αλλά το τελευταίο δένδρο της σειράς είναι και μικρότερο δυνατό με την μεγαλύτερη ακρίβεια στο σύνολο κλαδέματος. Για το λόγο αυτό θεωρείται πιο άμεση και ξεκάθαρη μέθοδος από την προηγούμενη.

Η χρήση ενός ξεχωριστού συνόλου κλαδέματος έχει το μειονέκτημα της μείωσης των διαθέσιμων δεδομένων εκπαίδευσης, κάτι ιδιαίτερα αρνητικό ειδικά σε περιπτώσει όπου τα διαθέσιμα δεδομένα είναι λίγα. Για να αντιμετωπιστεί αυτό το πρόβλημα έχει προταθεί μια διαδικασία που αποφεύγει τη δέσμευση δεδομένων για το κλάδεμα αλλά έχει υψηλή υπολογιστική πολυπλοκότητα. Μια άλλη λύση είναι ο υπολογισμός μιας πεισματικής εκδοχής του μέτρου της ακρίβειας στα δεδομένα. Για παράδειγμα ο αλγόριθμος C4.5 υπολογίζει την ακρίβεια στα δεδομένα εκπαίδευσης acc και μαζί με αυτή και την τυπική απόκλιση της std θεωρώντας διωνυμική κατανομή. Για κάποιο διάστημα εμπιστοσύνης η ακρίβεια υπολογίζεται ως το κάτω άκρο της εκτίμησης της ακρίβειας στα δεδομένα εκπαίδευσης(π.χ. Για διάστημα εμπιστοσύνης 95% η ακρίβεια υπολογίζεται ως acc-1.96std).

Μια διαφορετική προσέγγιση στο κλάδεμα ονομάζεται rule post pruning και εκτελείται σε 3 φάσεις :

1. κατασκευή του πλήρους δένδρου με υπερμοντελοποίηση των δεδομένων
2. μετατροπή του δένδρου σε ένα ισοδύναμο σύνολο κανόνων, δημιουργώντας

ένα κανόνα για κάθε διαδρομή από τη ρίζα προς κάποιο φύλλο (είτε με βάση κάποιο σύνολο κλαδέματος, είτε χρησιμοποιώντας μια πεισματική εκδοχή της ακρίβειας στο σύνολο εκπαίδευσης)

3. αν για ένα νέο δεδομένο ικανοποιούνται οι συνθήκες για παραπάνω από ένα κανόνα, τότε εκτελείται αυτός με τη μεγαλύτερη ακρίβεια

Τιμές που λείπουν

Σε πολλές περιπτώσεις (πχ ιατρικά δεδομένα) είναι πιθανό να λείπουν οι τιμές για κάποια χαρακτηριστικά. Το πρόβλημα σε μια τέτοια περίπτωση είναι ο υπολογισμός του μέτρου επιλογής χαρακτηριστικού σε τέτοιου είδους δεδομένα.

Έστω ότι σε ένα κόμβο του δένδρου θέλουμε να υπολογίσουμε την καταλληλότητα του χαρακτηριστικού A για έλεγχο. Ανάμεσα στα δεδομένα του κόμβου είναι το <x, c(x)>, για το οποίο η τιμή του A είναι άγνωστη. Μια λύση είναι να του αναθέσουμε την πιο κοινή τιμή που έχουν τα δεδομένα του τρέχοντος κόμβου. Εναλλακτικά θα μπορούσαμε να του αναθέσουμε την πιο κοινή τιμή που έχουν τα δεδομένα του κόμβου με την ίδια τιμή c(x) για την εξαρτημένη μεταβλητή.

Στον αλγόριθμο C4.5 ακολουθείται μια πιο σύνθετη στρατηγική, Αναθέτουμε πιθανότητες για την κάθε ξεχωριστή τιμή του χαρακτηριστικού με βάση το ποσοστό εμφάνισης των τιμών αυτών στα δεδομένα του κόμβου. Οι πιθανότητες συνεισφέρουν στον υπολογισμό του στατιστικού μέτρου, ενώ το δεδομένο μοιράζεται σε όλους τους κόμβους που ακολουθούν μαζί με την αντίστοιχη πιθανότητα. Η ίδια στρατηγική μπορεί τμηματικά να κατέβει από διάφορα κλαδιά του δένδρου, ενώ στο τέλος αθροίζουμε τις πιθανότητας για την κάθε απόφαση.

Cart

Τα δένδρα κατηγοριοποίησης και παλινδρόμησης (classification and regression trees – CART) είναι μία τεχνική που δημιουργεί ένα δυαδικό δένδρο απόφασης. Όπως και με τον ID3, η εντροπία χρησιμοποιείται σαν μέτρο για την επιλογή του καλύτερου γνωρίσματος διάσπασης και του κριτηρίου τερματισμού. Σε αντιδιαστολή με τον ID3, ωστόσο, όπου ένα παιδί δημιουργείται για κάθε υποκατηγορία, εδώ δημιουργούνται μόνο δύο παιδιά. Η διάσπαση εκτελείται γύρο από εκείνο το σημείο που θεωρείται σαν το καλύτερο σημείο διάσπασης. Σε κάθε βήμα, γίνεται μια πλήρης αναζήτηση για να καθορίσει την καλύτερη διάσπαση, όπου η “καλύτερη” ορίζεται από :

Αυτός ο τύπος υπολογίζεται στον τρέχοντα κόμβο t και για κάθε πιθανό γνώρισμα διάσπασης και κριτήριο s. Τα L και R χρησιμοποιούνται για να υποδηλώσουν το αριστερό και το δεξιό υποδένδρο του τρέχοντα κόμβου του δένδρου. Οι PL και PR εκφράζουν την πιθανότητα μία πλειάδα του συνόλου εκπαίδευσης να ανήκει στην αριστερή ή στην δεξιά πλευρά του δένδρου. Αυτή ορίζεται σαν : |πλειάδες υποδένδρου| / |πλειάδες του συνόλου εκπαίδευσης| .

Κάνουμε την παραδοχή ότι, σε περίπτωση ισοπαλίας, ακολουθείται η δεξιά

διακλάδωση. Η P(Cj| tL) ή η P(Cj| tR) είναι πιθανότητα μία πλειάδα να ανήκει σε αυτήν την κατηγορία Cj και στο αριστερό ή στο δεξιό υποδένδρο. Αυτή ορίζεται ως :
|πλειάδες της κατηγορίας j στο υποδένδρο| / |πλειάδες στον κόμβο εξέτασης|.

Σε κάθε βήμα, μόνο ένα κριτήριο επιλέγεται ως το καλύτερο απ' όλα τα πιθανά κριτήρια.Η ανάπτυξη του δένδρου σταματά όταν δεν υπάρχει καμία διάσπαση που να μπορεί να βελτιώσει την απόδοση. Τέλος αξίζει να αναφέρουμε ότι ο CART διαχειρίζεται ελλιπή δεδομένα με το να αγνοήσει απλά εκείνη την εγγραφή στον υπολογισμό της ορθότητας της διάσπασης ενός γνωρίσματος.

2.1.2 Μάθηση κατά Bayes

Η συλλογιστική κατά Bayes προσεγγίζει το θέμα της εξαγωγής συμπερασμάτων με βάση τις πιθανότητες. Στηρίζεται στην υπόθεση ότι οι ποσότητες που μας ενδιαφέρουν διέπονται από κατανομές πιθανοτήτων και ότι η εξαγωγή συμπερασμάτων με αυτές τις πιθανότητες σε συνδυασμό με δεδομένα που παρατηρούνται μπορούν να οδηγήσουν στη λήψη βέλτιστων αποφάσεων.

Η συλλογιστική κατά Bayes είναι πολύ σημαντική στη μηχανική μάθηση για δύο λόγους :

- Παρέχει τη βάση για την ανάπτυξη πρακτικών και αποτελεσματικών αλγορίθμων μάθησης που χειρίζονται πιθανότητες. Οι αλγόριθμοι μάθησης κατά Bayes έχουν σε ορισμένα προβλήματα την ίδια απόδοση με άλλες γνωστές μεθόδους, όπως τα δένδρα απόφασης και τα νευρωνικά δίκτυα. Επιπλέον είναι από τους πιο αποτελεσματικούς αλγορίθμους για την ταξινόμηση κειμένων.
- Παρέχει ένα πλαίσια ανάλυσης άλλων αλγορίθμων που δεν χειρίζονται πιθανότητες. Μπορεί να δώσει απαντήσεις σε ερωτήματα όπως “Πότε οι αλγόριθμοι μάθησης εννοιών καταλήγουν στην πιο πιθανή υπόθεση;” και “Πως μπορεί να εξηγηθεί με πιθανότητες η μεροληψία (bias) των αλγορίθμων δένδρων απόφασης;”.

Στη μάθηση κατά Bayes κάθε παράδειγμα μπορεί προοδευτικά να αυξήσει ή να μειώσει την πιθανότητα να είναι σωστή μια υπόθεση. Το γεγονός αυτό παρέχει έναν πιο ευέλικτο τρόπο μάθησης σε σχέση με άλλους αλγορίθμους που διαγράφουν τελείως μια υπόθεση αν είναι ασύμβατη έστω και με ένα παράδειγμα.

Ένα άλλο χαρακτηριστικό της μάθησης κατά Bayes είναι ότι τα δεδομένα μπορούν να συνδυαστούν με προϋπάρχουσα γνώση για την εξαγωγή της τελικής υπόθεσης. Η προϋπάρχουσα γνώση παρέχεται είτε ως αρχικές τιμές πιθανότητας για την κάθε υποψήφια υπόθεση είτε ως κατανομή πιθανοτήτων των υποψήφιων υποθέσεων στα δεδομένα εκπαίδευσης.

Εκτός από το στάδιο της εκπαίδευσης, η μάθηση κατά Bayes δίνει ευελιξία και στο

στάδιο της εφαρμογής της γνώσης. Τα μοντέλα που προκύπτουν από μια διαδικασία μάθησης κατά Bayes μπορούν να κάνουν προβλέψεις με πιθανότητα. Για παράδειγμα μια πρόβλεψη τέτοιου τύπου είναι "αυτός ο ασθενής με πνευμονία έχει 93% πιθανότητα να θεραπευθεί τελείως".

Ένα μειονέκτημα της μάθησης κατά Bayes είναι ότι απαιτεί τη γνώση αρκετών πιθανοτήτων. Αν οι τιμές αυτών των πιθανοτήτων δεν είναι γνωστές, τότε πρέπει να υπολογιστούν με βάση εμπειρική γνώση σχετικά με το πεδίο εφαρμογής, διαθέσιμα δεδομένα και υποθέσεις για τη μορφή της κατανομής αυτών των πιθανοτήτων.

Ένα άλλο μειονέκτημα είναι ότι παρουσιάζουν πολύ υψηλή πολυπλοκότητα. Στη γενικότερη περίπτωση η πολυπλοκότητα είναι γραμμική σε σχέση με τον αριθμό των υποψηφίων υποθέσεων. Αλλά ακόμα και σε περιπτώσεις που η υπολογιστική πολυπλοκότητα της μάθησης κατά Bayes καθιστά απαγορευτική τη χρήση της, μπορεί να χρησιμοποιηθεί ως κριτήριο για τον έλεγχο της απόδοσης άλλων μεθόδων.

Θεώρημα Bayes

Το θεώρημα του Bayes ορίζεται ως εξής :

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

όπου:

- P(h) είναι η προϋπάρχουσα πιθανότητα να ισχύει η υπόθεση h, χωρίς να έχουμε παρατηρήσει δεδομένα. Εκφράζει τη γνώση που μπορεί να έχουμε για τη συγκεκριμένη εφαρμογή υπό μορφή πιθανοτήτων για τις εναλλακτικές υποθέσεις. Αν δεν έχουμε καθόλου γνώση, τότε δίνουμε την ίδια πιθανότητα σε όλες τις υποθέσεις (πχ η πιθανότητα να έχει κάποιος γρίπη).
- P(D) είναι η προϋπάρχουσα πιθανότητα να παρατηρηθούν τα δεδομένα D, χωρίς να γνωρίζουμε ποια υπόθεση ισχύει.(πχ η πιθανότητα να έχει κάποιος πυρετό)
- P(D/h) είναι η υπό συνθήκη πιθανότητα να παρατηρηθούν τα δεδομένα D, όταν ισχύει η υπόθεση h (πχ η πιθανότητα να έχει κάποιος πυρετό, όταν έχει γρίπη).
- P(h/D) είναι η υπό συνθήκη πιθανότητα να ισχύει η υπόθεση h, όταν έχουν παρατηρηθεί τα δεδομένα D. Αυτή είναι η πιθανότητα που μας ενδιαφέρει στη Μηχανική Μάθηση. Εκφράζει τη βεβαιότητα ότι ισχύει η υπόθεση h έπειτα από την παρατήρηση των δεδομένων εκπαίδευσης D. (πχ η πιθανότητα να έχει κάποιος γρίπη, όταν έχει πυρετό).

Αφελής ταξινόμηση Bayes

Έστω μία περίπτωση x που περιγράφεται από τις εξής τιμές για τα χαρακτηριστικά <α1,α2,...,αn>. Σύμφωνα με τη μάθηση κατά Bayes η κατηγορία της περίπτωσης αυτής θα είναι η πιο πιθανή c ε C με βάση τα δεδομένα εκπαίδευσης :

$$c_B = \arg_{c \in C} \max \{P(c \mid a_1, a_2, ..., a_n)\}$$

Με τη βοήθεια του θεωρήματος Bayes η παραπάνω έκφραση ξαναγράφεται :

$$c_B = \arg_{c \in C} \max \{\frac{P(a_1, a_2, ..., a_n \mid c)P(c)}{P(c \mid a_1, a_2, ..., a_n)}\}$$

Επειδή ο παρανομαστής είναι σταθερός, αρκεί να υπολογιστούν οι όροι του αριθμητή από τα δεδομένα εκπαίδευσης, για να αποφασιστεί η πιο πιθανή κατηγορία.

Ο όρος P(c) μπορεί να υπολογιστεί εύκολα, μετρώντας τη συχνότητα εμφάνισης της κάθε κατηγορίας c στα δεδομένα εκπαίδευσης. Αντίθετα ο όρος P(a1,a2,...,an|c), πρακτικά δεν μπορεί να υπολογιστεί, εκτός και αν είναι διαθέσιμο ένα πάρα πολύ μεγάλο σύνολο δεδομένων εκπαίδευσης, Ο λόγος είναι ότι ο αριθμός όλων των δυνατών όρων P(a1,a2,...,an|c) που μπορεί να προκύψουν είναι ίσος με τον αριθμό των δυνατών περιπτώσεων επί τον αριθμό των κατηγοριών. Επομένων, για να υπολογιστεί μια καλή προσέγγιση του κάθε όρου θα πρέπει αυτός να εμφανίζεται πολλές φορές στα δεδομένα εκπαίδευσης.

Για να αντιμετωπιστεί αυτό το πρόβλημα ο αφελής ταξινομητής Bayes (naïve Bayes classifier – NB) κάνει τη θεώρηση ότι οι τιμές των χαρακτηριστικών δεδομένης μιας κατηγορίας, είναι ανεξάρτητες μεταξύ τους. Η θεώρηση αυτή θεωρείται "αφελής" επειδή στην πράξη τα χαρακτηριστικά δεν είναι ανεξάρτητα μεταξύ τους. Παρόλα αυτά, ο αφελής ταξινομητής Bayes έχει σε ορισμένες εφαρμογές απόδοση παρόμοια με τα δένδρα απόφασης και τα νευρωνικά δίκτυα.

Δεδομένης μιας κατηγορίας c, η πιθανότητα να παρατηρηθούν οι τιμές <a1,a2,...,an> είναι ίση με το γινόμενο των πιθανοτήτων να παρατηρηθεί η κάθε μια από τις τιμές αυτές

$$P(a_1, a_2, ..., a_n \mid c) = \prod_{i=1} nP(a_i \mid c)$$

ξεχωριστά :

Επομένως, η πρόβλεψη του αφελούς ταξινομητή Bayes είναι :

$$c_{NB} = \arg_{c \in C} \max \{P(c) \prod_{i=1}^{n} P(a_i \mid c)\}$$

Παρατηρούμε ότι πλέον το πλήθος των όρων P(ai|c) που πρέπει να υπολογιστούν από τα δεδομένα εκπαίδευσης είναι το γινόμενο των διαφορετικών τιμών όλων των χαρακτηριστικών επί τον αριθμό των κατηγοριών. Ο αριθμός αυτός είναι πολύ μικρότερος σε σχέση με τον προηγούμενο αριθμό των όρων P(a1,a2,...,an|c). Όταν ισχύει η θεώρηση της ανεξαρτησίας των τιμών των χαρακτηριστικών, τότε η απόφαση του NB, ισούται με την ποιο πιθανή κατηγορία CNB.

Η μάθηση σύμφωνα με τον αλγόριθμο NB συνίσταται στον υπολογισμό των πιθανοτήτων P(ai|c) και P(c) με βάση τη συχνότητα εμφάνισης των τιμών στα δεδομένα εκπαίδευσης, Αν το αi είναι αριθμητικό χαρακτηριστικό, τότε αρχικά υπολογίζεται ο μέσος όρος μc και η τυπική απόκλιση σc των τιμών του χαρακτηριστικού για τα παραδείγματα με τιμή c για την εξαρτημένη μεταβλητή. Η ποσότητα P(ai|c) g(ai,μc,σc), όπου η g είναι η συνάρτηση πυκνότητας πιθανότητας για την κατανομή πιθανότητας (πχ κανονική ή Gauss), την οποία θεωρούμε ότι ακολουθούν οι τιμές του χαρακτηριστικού. Οι πιθανότητες που υπολογίζονται αποτελούν την υπόθεση που "έμαθε" ο αλγόριθμος. Στη συνέχεια, κάθε νέα περίπτωση ταξινομείται σύμφωνα με τον παραπάνω τύπο.

Μια άλλη λύση στο πρόβλημα των αριθμητικών χαρακτηριστικών είναι η διακριτοποίηση τους. Σε μια μελέτη σχετικά με την διακριτοποίηση αριθμητικών χαρακτηριστικών, έγινε σύγκριση της απόδοσης του NB σε 16 σύνολα δεδομένων, τόσο υποθέτοντας κατανομή Gauss για τα αριθμητικά χαρακτηριστικά όσο και διακριτοποιώντας τα με διάφορες μεθόδους. Τα αποτελέσματα έδειξαν ότι η χρήση της διακριτοποίησης αύξησε κατά πολύ τη μέση ακρίβεια του NB.

Ο αφελής ταξινομητής Bayes απαιτεί απλά ένα πέρασμα σε μία βάση δεδομένων για τον υπολογισμό των πιθανοτήτων και επομένως για τη φάση της εκπαίδευσης είναι σχετικά γρήγορος αλγόριθμος. Ωστόσο το κύριο μειονέκτημά του είναι ότι η αποδοτικότητα πρόβλεψης των μοντέλων που παράγει είναι μειωμένη όταν υπάρχουν αλληλεπιδράσεις μεταξύ δύο ή περισσότερων χαρακτηριστικών (πεδίων), διότι βασίζεται στην ακριβώς αντίθετη παραδοχή.

3 Η εφαρμογή

Εισαγωγή

Στο κεφάλαιο αυτό θα ασχοληθούμε με την εφαρμογή που υλοποιήθηκε στα πλαίσια της πτυχιακής εργασίας. Θα γίνει μια σύντομη περιγραφή του προβλήματος που ζητήθηκε να εξεταστεί και του τρόπου που αυτό αντιμετωπίστηκε. Ακόμα θα γίνει λεπτομερής περιγραφή της εφαρμογή, του τρόπου υλοποίησής της, αλλά και σχολιασμός των αποτελεσμάτων που αυτή παρέχει.

3.1 Περιγραφή του προβλήματος

Σκοπός της παρούσας πτυχιακής εργασίας είναι η υλοποίηση μιας εφαρμογής, που θα υποστηρίζει την πρόγνωση του αποτελέσματος αγώνων ποδοσφαίρου κάποιου σημαντικού πρωταθλήματος, με χρήση τεχνικών εξόρυξης δεδομένων και μηχανικής μάθησης. Απώτερος σκοπός της εργασίας θα είναι τόσο η επίτευξη ενός ποσοστού επιτυχίας της τάξης του 60%, αλλά και η καταγραφή των επιδόσεων γνωστών αλγορίθμων μηχανικής μάθησης στο πρόβλημα της πρόγνωσης αποτελέσματος.

3.2 Η προσέγγιση του προβλήματος

3.2.1 Επιλογή Εργαλείων – Γλώσσας Προγραμματισμού

Ο τρόπος λειτουργίας της εφαρμογής βασίζεται σε τρία κύρια συστατικά:

1. Μία βάση δεδομένων όπου αποθηκεύονται πληροφορίες σχετικές με το πρωτάθλημα που είναι υπό εξέταση. Πληροφορίες όπως ποιες είναι οι ομάδες που αγωνίζονται στο πρωτάθλημα, αποτελέσματα αγώνων, ημερομηνίες διεξαγωγής των αγώνων,πληροφορίες για την αγωνιστική κατάσταση των ομάδων που συμμετέχουν σε ένα αγώνα κ.α. Η βάση θα υλοποιηθεί με την χρήση του εργαλείου ανοιχτού κώδικα MySQL.
2. Η προγραμματιστική γλώσσα Python. Η Python είναι μια εύκολη στην εκμάθηση, ισχυρή γλώσσα προγραμματισμού . Έχει αποδοτικές δομές δεδομένων υψηλού επιπέδου και μια απλή αλλά αποτελεσματική προσέγγιση στον αντικειμενοστραφή προγραμματισμό. Η κομψή σύνταξη της Python και οι δυναμικοί τύποι της, μαζί με τη λειτουργία της ως διερμηνευόμενης(αντί μεταγλωττιζόμενης) γλώσσας, την καθιστούν την ιδανική γλώσσα για δημιουργία σεναρίων εντολών και για ταχεία ανάπτυξη εφαρμογών σε πολλούς τομείς και στις περισσότερες πλατφόρμες.
3. Η γλώσσα Django. Η Django είναι μία high-level Python Web framework που ενθαρρύνει την ταχεία ανάπτυξη Web εφαρμογών. Έχει αρκετά αυτοματοποιημένες διαδικασίες που διευκολύνουν το έργο του προγραμματιστή.

Ο χρήστης έχει την δυνατότητα να ζητήσει πρόγνωση για ένα πρωτάθλημα με τους

διαθέσιμους αλγορίθμους και η εφαρμογή θα του δείξει ποια από τα τρία αποτελέσματα (νίκη γηπεδούχου, ισοπαλία, νίκη φιλοξενούμενου), θεωρεί ότι θα είναι το αποτέλεσμα του κάθε αγώνα για την τρέχουσα αγωνιστική. Η ανανέωση της βάσης γίνεται από τον administrator χειροκίνητα, για σίγουρη/εξασφαλισμένη εγκυρότητα της.

3.2.2 Η επιλογή του πρωταθλήματος

Η πρώτη επιλογή που έπρεπε να γίνει κατά τη διάρκεια του σχεδιασμού της εφαρμογής, ήταν αυτή του πρωταθλήματος που θα χρησιμοποιούνταν. Θα έπρεπε πρώτα από όλα να είναι ένα πρωτάθλημα όπου τα αποτελέσματα των αγώνων να είναι ανεξάρτητα από εξωτερικούς παράγοντες, που θα ήταν πιθανόν δύσκολο να μοντελοποιηθούν(όπως διαιτησία, φίλαθλοι, εξωαγωνιστικοί παράγοντες κ.α.). Θα έπρεπε ακόμα να είναι ένα πρωτάθλημα με όσο το δυνατόν περισσότερους αγώνες ανά αγωνιστική περίοδο, ούτως ώστε να είναι και μεγαλύτερα τα ποσά πληροφορίας που θα ήταν διαθέσιμα προς ανάλυση.

Τα παραπάνω, σε συνδυασμό με τις προσωπικές προτιμήσεις, οδήγησαν στην επιλογή του πρωταθλήματος της πρώτης κατηγορίας της Αγγλίας (Premier League). Στην Premier League συμμετέχουν 20 ομάδες, που αγωνίζονται μεταξύ τους σε διπλούς αγώνες, που μεταφράζεται σε 380 αγώνες ανά έτος. Δημιουργήθηκε μία βάση δεδομένων 5 χρόνων έτσι ώστε να θεωρηθεί αρκετά ικανοποιητικό σε μέγεθος το σύνολο εκπαίδευσης.

3.2.3 Η επιλογή των χαρακτηριστικών

Η επόμενη βασική επιλογή που έπρεπε να γίνει ήταν αυτή των χαρακτηριστικών που επηρεάζουν την έκβαση ενός αγώνα ποδοσφαίρου. Μετά από πολλή σκέψη και πειραματισμό επιλέχθηκαν τα παρακάτω χαρακτηριστικά πάνω στα οποία βασίστηκε η αναπαράσταση του προβλήματος:

1. Η κατάταξη της γηπεδούχου ομάδας στον πίνακα βαθμολογίας του πρωταθλήματος.
2. Η κατάταξη της φιλοξενούμενης ομάδας στον πίνακα βαθμολογίας του πρωταθλήματος.
3. Η φόρμα της γηπεδούχου ομάδας.
4. Η φόρμα της φιλοξενούμενης ομάδας.
5. Η φόρμα της γηπεδούχου ομάδας στα εντός έδρας παιχνίδια.
6. Η φόρμα της φιλοξενούμενης ομάδας στα εκτός έδρας παιχνίδια.
7. Η προϊστορία των αγώνων ανάμεσα στις δύο εμπλεκόμενες ομάδες.

Στο σημείο αυτό πρέπει να αναφερθεί ότι οι απόψεις επί του συγκεκριμένου ζητήματος ποικίλουν σε μεγάλο βαθμό. Το σύνολο των παραγόντων που καθορίζουν την έκβαση ενός αγώνα, αλλάζει ανάλογα με τον παρατηρητή. Η παραπάνω προσέγγιση των επτά παραγόντων επιλέχθηκε με σκοπό να συμπεριληφθούν κάποια από τα πιο σημαντικά χαρακτηριστικά, ούτως ώστε να επιτευχθεί μία ευέλικτη, μικρή σε όγκο, αλλά ουσιαστική και περιεκτική αναπαράσταση.

Από την άλλη όμως, δεν έχουν συμπεριληφθεί άλλοι, κατά γενική ομολογία σημαντικοί παράγοντες. Τέτοιοι είναι οι απουσίες παικτών από την ενδεκάδα μίας ομάδας,

το αν μία ομάδα έχει να δώσει ευρωπαϊκό αγώνα μέσα στην εβδομάδα, ή ακόμα το ενδιαφέρον(ή την αδιαφορία) που μπορεί να έχει μία ομάδα για έναν συγκεκριμένο αγώνα. Αν για παράδειγμα σε έναν αγώνα μεταξύ ομάδων Α και Β, κρίνεται η παραμονή ή όχι της ομάδας Α στην κατηγορία, ενώ η ομάδα Β δεν έχει βαθμολογικό ενδιαφέρον, τότε η ομάδα Α έει μεγαλύτερη πιθανότητα να κερδίσει το συγκεκριμένο αγώνα. Οι παραλήψεις αυτές έγιναν είτε λόγο δυσκολίας αναπαράστασης, είτε λόγο δυσκολίας εύρεσης τέτοιον πληροφοριών για αγώνες που έγιναν πέντε χρόνια πριν.

Με το τρόπο αυτό θα δημιουργηθεί ένας πίνακας όπου θα είναι αποθηκευμένοι οι διεξαχθέντες αγώνες μαζί με τα επτά είδη πληροφορίας που έχουν κριθεί σημαντικά, ανοίγοντας έτσι τον δρόμο στην μηχανική μάθηση να αναλύσει τον πίνακα, που θα αποτελεί τα δεδομένα εκπαίδευσης, και να μπορεί στη συνέχεια να κατηγοριοποιήσει έναν καινούριο αγώνα που ζητήθηκε να προγνωστεί το αποτέλεσμά του.

3.3 Περιγραφή της βάσης δεδομένων

Η βάση δεδομένων του συστήματος αποτελεί το μοναδικό μέσο αποθήκευσης, και υλοποιήθηκε με το εργαλείο ανοιχτού κώδικα MySQL. Παρακάτω παρουσιάζονται αναλυτικά ποια τα χαρακτηριστικά του πίνακα και ποιος ο ρόλος τους στην αναπαράσταση του προβλήματος. Θα ήθελα να τονίσω ότι επιλέχθηκε η υλοποίηση της βάσης δεδομένων σε έναν πίνακα, για να αποφευχθούν θέματα πολυπλοκότητας και δημιουργία παράλληλων συνδέσεων με διαφορετικούς πίνακες για άντληση πληροφοριών. Αλλά και για μελλοντική χρήση άλλον πρωταθλημάτων, ώστε να γίνει πιο εύκολη η οργάνωση των αποθηκευμένων πληροφοριών

3.3.1 Ο πίνακας Stats – England

Είναι ο πίνακας όπου αποθηκεύονται τα αποτελέσματα των αγώνων που έχουν διεξαχθεί. Κάθε πλειάδα του πίνακα υλοποιεί έναν αγώνα που έχει διεξαχθεί. Ο πίνακας αποτελείται από τα παρακάτω χαρακτηριστικά:

1. date : Είναι τύπου date και υλοποιεί την ημερομηνία διεξαγωγής του αγώνα(σύνθετο κλειδί)
2. hometeam : Είναι τύπου varchar και υλοποιεί το όνομα της γηπεδούχου ομάδας(σύνθετο κλειδί).
3. awayteam : Είναι τύπου varchar και υλοποιεί το όνομα της φιλοξενούμενης ομάδας(σύνθετο κλειδί).
4. fthgoals : Είναι τύπου int και υλοποιεί το υλοποιεί το πλήθος των τερμάτων που πέτυχε η γηπεδούχος ομάδα στον αγώνα.
5. ftagoals : Είναι τύπου int και υλοποιεί το υλοποιεί το πλήθος των τερμάτων που πέτυχε η φιλοξενούμενη ομάδα στον αγώνα.

6. ftresult : Είναι τύπου varchar και περιέχει το τελικό αποτέλεσμα του αγώνα. Μπορεί να έχει τιμή 'H' για νίκη γηπεδούχου ,"A" για νίκη του φιλοξενούμενου και "D" για ισοπολία.
7. htrating : Είναι τύπου float που κυμαίνεται στο διάστημα [0,3], και αναπαριστά τους βαθμούς που έχει συλλέξει η γηπεδούχος ομάδα κατά την διάρκεια του πρωταθλήματος σε κανονικοποιημένη μορφή. Στην ουσία πρόκειται για το λόγο των βαθμών που έχει αποσπάσει η ομάδα προς τους αγώνες που έχει δώσει την συγκεκριμένη αγωνιστική:

$$\textbf{home\_team\_rating=} \frac{\text{βαθμοί γηπεδούχου ομάδας}}{\text{αγώνες που έχει δώσει}}$$

Αυτό γίνεται ούτως ώστε τα δεδομένα να είναι ανεξάρτητα από το πόσοι αγώνες έχουν γίνει στο πρωτάθλημα, και να υπάρχει κοινό μέτρο σύγκρισης μεταξύ των πλειάδων του πίνακα. Για παράδειγμα στην αρχή του πρωταθλήματος μια ομάδα μπορεί να έχει συλλέξει 7 βαθμούς και στον επόμενο αγώνα να κερδίσει, ενώ προς το τέλος του πρωταθλήματος να έχει 67 και στον επόμενο αγώνα να χάσει. Με την παραπάνω προσέγγιση αυτό το πρόβλημα λύνεται αφού ανά πάσα στιγμή αυτό που θα τίθεται προς σύγκριση θα είναι ο μέσος όρος των βαθμών ανά αγώνα που η ομάδα αποσπά

8. atrating : Είναι τύπου float που κυμαίνεται στο διάστημα [0,3] και αντίστοιχα με την περίπτωση του htrating αναπαριστά τους βαθμούς που έχει συλλέξει η φιλοξενούμενη ομάδα κατά τη διάρκεια του πρωταθλήματος σε κανονικοποιημένη μορφή.
9. rating_diff : Είναι τύπου float που κυμαίνεται στο διάστημα [-3,3] και η τιμή της ισούται με τη διαφορά του μέσου όρου των βαθμών της γηπεδούχου ομάδας από αυτόν της φιλοξενούμενης:

   rating_diff = htrating – atrating

Η διαφορά ανάμεσα στους δύο μέσους όρους είναι το χαρακτηριστικό που θα δοθεί στην συνέχεια προς ανάλυση από τους αλγορίθμους τεχνητής νοημοσύνης και όχι οι μέσοι όροι από μόνοι τους. Αυτό γίνεται με σκοπό να καθοδηγηθούν οι αλγόριθμοι ως προς το ποια από τις δύο ομάδες είναι μεγαλύτερης δυναμικότητας. Βάζοντας το rating_diff και όχι το μέσο όρο κάθε ομάδας ξεχωριστά, κάνουμε τους αλγορίθμους πιο γρήγορους, αλλά και πιο αποδοτικούς.

10. home_team_form : Είναι τύπου int που κυμαίνεται στο διάστημα [0,9] και αναπαριστά τη φόρμα της γηπεδούχου ομάδας. Πρόκειται στην ουσία για το άθροισμα των βαθμών που έχει αποσπάσει η γηπεδούχος ομάδα στους τρεις τελευταίους αγώνες. Οπότε μπορεί να πάρει τιμές από το 0 έως 9, εκτός από την τιμή 8 (δεδομένου ότι η νίκη δίνει 3 βαθμούς, η ισοπαλία 1 και η ήττα κανένα). Οι τρεις αγώνες κρίθηκαν ιδανικό χρονικό διάστημα για τον υπολογισμό της φόρμας μιας ομάδας καθώς ένα μεγαλύτερο ή μικρότερο διάστημα θα ήταν αποπροσανατολιστικό.

11. away_team_form : Είναι τύπου int που κυμαίνεται στο διάστημα [0,9] και αντίστοιχα με την home_team_form αναπαριστά την φόρμα της φιλοξενούμενης ομάδας.
12. form_diff : Είναι τύπου int που κυμαίνεται στο διάστημα [-9,9] και η τιμή της ισούται με την διαφορά της φόρμας της γηπεδούχου ομάδας από αυτή της φιλοξενούμενης

    form_diff= home_team_form – away_team_form

Όπως και στην περίπτωση της rating_diff στόχος είναι η απλούστευση της αναπαράστασης και η υποβοήθηση των αλγορίθμων, μειώνοντας τις μεταβλητές από δύο σε μία.

13. htform_at_home : Είναι τύπου int που κυμαίνεται στο διάστημα [0,15] και αναπαριστά τη φόρμα της γηπεδούχου ομάδας στην έδρα της. Πρόκειται για το σύνολο των βαθμών που έχει αποσπάσει η ομάδα στους τελευταίους 5 αγώνες στην έδρα της. Αυτή τη φορά το χρονικό διάστημα αυξήθηκε από τρεις αγώνες σε πέντε αγώνες με το σκεπτικό ότι το πώς αγωνίζεται μία ομάδα εντός έδρας (και αντίστοιχα εκτός) δεν είναι μόνο θέμα φόρμας αλλά και άλλων παραγόντων. Για παράδειγμα μια ομάδα μπορεί να είναι πολύ ισχυρή στην έδρα της λόγω των φιλάθλων της, μια άλλη να είναι δυνατή στους εκτός έδρας αγώνες λόγο της αμυντικής στρατηγικής που ακολουθεί κ.τ.λ. Τέτοιου είδους πληροφορίες ανακαλύπτονται ευκολότερα σε ένα μεγαλύτερο χρονικό διάστημα.
14. atform_at_away : Είναι τύπου int που κυμαίνεται στο διάστημα [0,15] και αντίστοιχα με την htform_at_home αναπαριστά την φόρμα της φιλοξενούμενης ομάδας στους τελευταίους 5 εκτός έδρας αγώνες που έχει δώσει. Και πάλι η τιμή της ισούται με το άθροισμα των βαθμών που απέσπασε στα τελευταία 5 εκτός έδρας παιχνίδια.
15. home_away_form_diff : Είναι τύπου int και κυμαίνεται στο διάστημα [-15,15]. Παρόμοια με τις προηγούμενες περιπτώσεις διαφορών, η τιμή της ισούται με την διαφορά τω βαθμών που απέσπασε η γηπεδούχος ομάδα στους τελευταίους 5 αγώνες στην έδρα της, από αυτούς που απέσπασε η φιλοξενούμενη ομάδα στους τελευταίους 5 αγώνες που έδωσε εκτός έδρας:

    home_away_form_diff= htform_at_home - atform_at_away

16. past_games : Είναι τύπου varchar και αναπαριστά την προϊστορία των αγώνων ανάμεσα στις δύο εμπλεκόμενες ομάδες τα 5 τελευταία χρόνια. Οι τιμές που μπορεί να πάρει το συγκεκριμένο χαρακτηριστικό είναι οι εξής:
    - Clear_Home_Win : Όταν το πλήθος των νικηφόρων αγώνων που έχει σημειώσει η γηπεδούχος ομάδα ξεπερνά σε ποσοστό το 60% των αγώνων μεταξύ των δύο ομάδων τα τελευταία 5 χρόνια. Σημειώστε ότι στην πενταετία αυτή ο αριθμός των αγώνων ανάμεσα σε δύο ομάδες δεν είναι αναγκαστικά πέντε, αφού κάποια ομάδα μπορεί να μην συμμετείχε και τα πέντε χρόνια στην πρώτη κατηγορία.
    - Clear_Away_Win : Όταν το πλήθος των νικηφόρων αγώνων που έχει σημειώσει η φιλοξενούμενη ομάδα ξεπερνά σε ποσοστό το 60% των γώνων μετξύ των δύο ομάδων τα τελευταία 5 χρόνια.

- Clear_Draw : Όταν το πλήθος των ισοπαλιών ανάμεσα στις δύο ομάδες τα τελευταία 5 χρόνια ξεπερνάει σε ποσοστό το 60% των αγώνων μεταξύ των δύο αυτών ομάδων μέσα στη πενταετία.
- Home_Win : Όταν οι νίκες της γηπεδούχου τα τελευταία πέντε χρόνια είναι περισσότερες και από τις νίκες της φιλοξενούμενης και από τις ισοπαλίες , χωρίς να ξεπερνά το παραπάνω 60%
- Away_Win : Όταν οι νίκες της φιλοξενούμενης ομάδας τα τελευταία πέντε χρόνια είναι περισσότερες και από τις νίκες της γηπεδούχου και από τις ισοπαλίες, χωρίς να ξεπερνά το παραπάνω 60%.
- Draw : Όταν οι ισοπαλίες τα τελευταία 5 χρόνια είναι περισσότερες και από τις νίκες της γηπεδούχου και από τις νίκες της φιλοξενούμενης, αλλά δεν ξεπερνάνε το παραπάνω 60%.
- mixed_result : Όταν δεν συμβαίνει καμία από τις παραπάνω περιπτώσεις.

Παρατηρούμε ότι το κλειδί του πίνακα είναι σύνθετο και μάλιστα τριπλό. Αποτελείται από τα χαρακτηριστικά hometeam, awayteam,date. Αυτό σημαίνει ότι δεν μπορεί να υπάρξει δεύτερος αγώνας ανάμεσα σε δύο συγκεκριμένες ομάδες με την ίδια ημερομηνία. Οπότε για να γίνει αναζήτηση στην βάση δεδομένων για έναν αγώνα πρέπει να γνωρίζουμε τις δύο ομάδες που συμμετείχαν σε αυτόν καθώς και την ημερομηνία διεξαγωγής του.

Ακόμα για μελλοντική χρήση έχουν βρεθεί και περιληφθεί χαρακτηριστικά όπως:

1. hth_goals : Είναι τύπου int και υλοποιεί το πλήθος των τερμάτων που πέτυχε η γηπεδούχος ομάδα στο ημίχρονο.
2. hta_goals : Είναι τύπου int και υλοποιεί το πλήθος των τερμάτων που πέτυχε η φιλοξενούμενη ομάδα στο ημίχρονο.
3. htresult : Είνι τύπου varchar και είναι το αποτέλεσμα ημιχρόνου. Παίρνει τιμές 'H' για νίκη γηπεδούχου, 'A' για νίκη φιλοξενούμενης και 'D' για ισοπαλία.
4. over_under : Είναι τύπου varchar και μπορεί να πάρει τιμή Over αν στον αγώνα οι ομάδες πέτυχαν από 3 τέρματα και πάνω και Under αν στον αγώνα οι ομάδες πέτυχαν κάτω από 3 τέρματα.
5. hredcards : Είναι τύπου int και περιέχει τον αριθμό των κόκκινων καρτών που δέχθηκε η γηπεδούχος ομάδα κατά τη διάρκεια του αγώνα.
6. aredcards : Είναι τύπου int και περιέχει τον αριθμό των κόκκινων καρτών που δέχθηκε η φιλοξενούμενη ομάδα κατά τη διάρκεια του αγώνα.

3.4 Τεχνική ανάλυση κώδικα

Για την υλοποίηση της εφαρμογής επιλέχθηκαν δύο αλγόριθμοι που ξεχωρίζουν για την απλότητά τους, αλλά και την αποτελεσματικότητά τους. Ο λόγος για τους CART και Bayes (naïve). Στην συνέχεια θα παρουσιάσουμε πως ο καθένας από αυτούς υλοποιήθηκε στην εφαρμογή. Ο κώδικας υλοποιήθηκε στη γλώσσα Python.

3.4.1 CART

Αρχικά θα πρέπει να δώσουμε μορφή στο δένδρο και να καθορίσουμε τη δομή του. Το δένδρο θα έχει σαν χαρακτηριστικά :

- col : Το χαρακτηριστικό που εξετάζεται
- value : Η τιμή που πρέπει να έχει ένα χαρακτηριστικό για να είναι Αληθής
- results : Κρατάει ένα κατάλογο με τα αποτελέσματα(H/D/A για την περίπτωσή μας) για κάθε φύλλο του δένδρου. Για κάθε κόμβο πέρνει τιμή None
- tb : Η διακλάδωση για Αληθές (True Branch)
- fb : Η διακλάδωση για Ψευδές (False Branch)

Έτσι δημιουργούμε μία κλάση και την αρχικοποιούμε όπως φαίνεται παρακάτω:

```
class decisionnode:
  def __init__(self,col=-1,value=None,results=None,tb=None,fb=None):
    self.col=col
    self.value=value
    self.results=results
    self.tb=tb
    self.fb=fb
```

divideset(self,rows,column,value)

Στην συνέχεια θα πρέπει να δημιουργήσουμε μία συνάρτηση η οποία θα χωρίζει ένα σύνολο από πλειάδες σύμφωνα με μία τιμή (value), σε δύο υποσύνολα ανάλογα με το αν είναι αληθές ή ψευδές η στήλη(col) της κάθε εξεταζόμενη πλειάδα. Αν το κριτήριο για τον διαχωρισμό είναι αριθμητικό(για παράδειγμα αν value=10) τότε θα είναι True αν η τιμή της συγκεκριμένης στήλης από τη πλειάδα υπό εξέταση είναι >=10, αλλιώς θα είναι False. Αν πάλι το κριτήριο διαχωρισμού είναι αλφαριθμητικό(για παράδειγμα hometeam='Arsenal') τότε είναι True αν η στήλη της πλειάδας που εξετάζεται έχει την ίδια τιμή, αλλιώς είναι false. Έτσι έχουμε την παρακάτω συνάρτηση:

```
def divideset(self,rows,column,value):
  split_function=None
  if isinstance(value,int) or isinstance(value,float):
    split_function=lambda row:row[column]>=value
  else:
    split_function=lambda row:row[column]==value
  set1=[row for row in rows if split_function(row)]          # To set για τα  True
  set2=[row for row in rows if not split_function(row)]      #Για τα  False
  return (set1,set2)
```

uniquecounts(self,rows)

Ακόμα χρειαζόμαστε μία συνάρτηση που θα παίρνει σαν όρισμα ένα σύνολο από πλειάδες και θα δημιουργεί τη λίστα με τα αποτελέσματα(results). Για παράδειγμα στα φύλλα του δένδρου θέλουμε να γνωρίζουμε τα αποτελέσματα που περιέχουν(π.χ. 5 'H' και 1 'A') για να μπορέσουμε μετά να κάνουμε πρόγνωση. Η λειτουργία αυτή υλοποιείται από τον παρακάτω κώδικα:

```
def uniquecounts(self,rows):
  results={}
  for row in rows:
    r=row[len(row)-1]                    #Δείχνει στο ftresult,η τελευταία στήλη
    if r not in results: results[r]=0    #Αν εμφανίστηκε το αποτέλεσμα r για πρώτη φορά
    results[r]+=1                        #Μετρητής για το αποτέλεσμα  r
  return results
```

Τώρα θα πρέπει να υλοποιήσουμε τη συνάρτηση σύμφωνα με την οποία θα αποφασίζουμε ποιος είναι ο καλύτερος διαχωρισμός(σύμφωνα με ποιο χαρακτηριστικό). Δηλαδή με κάποιο τρόπο θα αποφασίζει με ποιο διαχωρισμό(split) θα έχουμε μεγαλύτερο κέρδος πληροφορίας άρα και καλύτερη αποτελεσματικότητα. Για αυτή τη λειτουργικότητα υπάρχουν δύο διαδικασίες για τις οποίες θα μιλήσουμε παρακάτω για την αποτελεσματικότητά τους.

gini_impurity(self,rows)

Η πρώτη είναι η **Gini Impurity.** Η λειτουργικότητά της και τα αποτελέσματα που βγάζει βασίζονται στην λογική "αν τοποθετήσω τυχαία ένα αντικείμενο ποια η πιθανότητα να είναι λάθος". Για παράδειγμα αν έχουμε ένα σύνολο από πλειάδες από τις οποίες οι 9 έχουν result H και η μία έχει result A, τότε το αποτέλεσμα που θα επιστρέψει θα είναι 1/10=0.1. Αντίθετα αν έχουμε 5 με αποτέλεσμα H και 5 με αποτέλεσμα A θα επιστρέψει 0.5.Από τον τρόπο λειτουργίας της Gini Impurity καταλαβαίνουμε ότι όσο μικρότερος είναι ο αριθμός που επιστρέφει τόσο καλύτερο διαχωρισμό των πλειάδων έχουμε κάνει. Δηλαδή σαν καλύτερο διαχωρισμό (best split) κρατάμε αυτόν με το μικρότερο αποτέλεσμα. Ο κώδικας είναι ο ακόλουθος:

```
def gini_impurity(self,rows):
  total=len(rows)
  counts=uniquecounts(rows)
  imp=0
  for k1 in counts:
    p1=float(counts[k1])/total
    for k2 in counts:
        if k1==k2:continue
        p2=float(counts[k2])/total
        imp+=p1*p2
  return imp
```

entropy(self,rows)

Ο άλλος τρόπος υλοποίησης του προβλήματος της εύρεσης του best split, που είναι και αυτός που προτιμήθηκε, είναι αυτός που χρησιμοποιεί τον τύπο της εντροπίας. Η εντροπία υπολογίζεται με το άθροισμα των p(x)*log2(p(x)) για όλα τα δυνατά αποτελέσματα. Ουσιαστικά η εντροπία μπορεί να καταλάβει πόσο διαφορετικά είναι τα αποτελέσματα από μία πλειάδα αποτελεσμάτων. Αν για παράδειγμα όλα τα αποτελέσματα είναι ίδια, τότε η μέθοδος της εντροπίας θα επιστρέψει 0. Όσο μεγαλύτερη ανομοιογένεια έχουμε στα αποτελέσματα τόσο μεγαλύτερο θα είναι το αποτέλεσμα της εντροπίας. Σκοπός μας είναι να χωρίσουμε τα δεδομένα με τέτοιο τρόπο ώστε να μειώσουμε όσο το δυνατόν

$$E(S) = \sum_{c \in C} -p_c \log 2 p_c$$

περισσότερο γίνεται την εντροπία.

Ο κώδικας για την μέθοδο της εντροπίας είναι ο ακόλουθος:

```
def entropy(self,rows):
  from math import log
  log2=lambda x:log(x)/log(2)                  #Κατασκευάζω την συνάρτηση για log2
  results=self.uniquecounts(rows)              #Παίρνω τη λίστα με το πως μοιράζονται τα αποτελέσματα
  entr=0.0
  for r in results.keys():                     #Για κάθε result υπολογίζω την εντροπία
    p=float(results[r])/len(rows)
    entr=entr-p*log2(p)
  return entr
```

buildtree(self,rows)

Και μετά από όλες αυτές τις συναρτήσεις μπορούμε να περάσουμε στη συνάρτηση για την κατασκευή του δένδρου απόφασης. Όπως αναφέραμε παραπάνω θα χρησιμοποιήσουμε τη μέθοδο τις εντροπίας για λόγους ταχύτητας κυρίως, αλλά και αποτελεσματικότητας σε κάποιες των περιπτώσεων. Ο κώδικας είναι ο ακόλουθος:

```
def buildtree(self,rows):
  if len(rows)==0: return decisionnode()
  current_score=self.entropy(rows)
```

```
  best_gain=0.0                                   #Για να κρατάει το καλύτερο κέρδος
  best_criteria=None                              #Για το χαρακτηριστικό με το καλύτερο κέρδος
  best_sets=None                                  #Κρατάει τα sets με το καλύτερο split

  column_count=len(rows[0])-2
  for col in range(0,column_count):               #Εξετάζω για κάθε col και κάθε τιμή το κέρδος
   column_values={}
   for row in rows:
       if col==2:
         column_values[round(row[col],1)]=1
       else:
         column_values[row[col]]=1
   for value in column_values.keys():
       (set1,set2)=self.divideset(rows,col,value)
       p=float(len(set1))/len(rows)
       gain=current_score-p*self.entropy(set1)-(1-p)*self.entropy(set2)
       if gain>best_gain and len(set1)>0 and len(set2)>0:
         best_gain=gain
         best_criteria=(col,value)
         best_sets=(set1,set2)

  if best_gain>0:                          #Αν βρήκα διαχωρισμό με κέρδος>0 φτιάχνω τους κλάδους
   trueBranch=self.buildtree(best_sets[0])
   falseBranch=self.buildtree(best_sets[1])
   return decisionnode(col=best_criteria[0],value=best_criteria[1],tb=trueBranch,fb=falseBranch)
  else:
   return decisionnode(results=self.uniquecounts(rows))   #Αν κέρδος=0 τότε κάνε τη πλειάδα φύλλο
```

prune(self,tree,mingain)

Για να αντιμετωπίσουμε το πρόβλημα της υπερμοντελοποίησης (overfitting) προσθέτουμε άλλη μία συνάρτηση ο σκοπός της οποίας είναι να κλαδέψει το δένδρο. Αυτό το κάνει σύμφωνα με το κέρδος ενός διαχωρισμού. Ουσιαστικά ορίζουμε κατώφλια κέρδους, ότι για παράδειγμα διαχωρισμοί με κέρδος κάτω από 0.2 να μην γίνονται. Έτσι έχουμε τον παρακάτω κώδικα.

```
 def prune(self,tree,mingain):
  if tree.tb.results==None:
   self.prune(tree.tb,mingain)
  if tree.fb.results==None:
   self.prune(tree.fb,mingain)
  if tree.tb.results!=None and tree.fb.results!=None:
   tb,fb=[],[]
   for v,c in tree.tb.results.items():
       tb+=[[v]]*c
   for v,c in tree.fb.results.items():
       fb+=[[v]]*c
   delta=float(self.entropy(tb+fb))-float((self.entropy(tb)+self.entropy(fb)/2))
   if abs(delta)<mingain:
       print 1
       tree.tb,tree.fb=None,None
       tree.results=self.uniquecounts(tb+fb)
```

```
    else:
        tree.results=None
```

classify(self,observation,tree)

Τέλος χρειαζόμαστε μία συνάρτηση η οποία θα παίρνει μία νέα παρατήρηση και σύμφωνα με το δένδρο απόφασης που δημιουργήσαμε θα κάνει πρόγνωση για το αποτέλεσμα. Ο κώδικας για την λειτουργία αυτή είναι ο ακόλουθος:

```
def classify(self,observation,tree):
  if tree.results!=None:
    return tree.results
  else:
    v=observation[tree.col]
    branch=None
    if isinstance(v,int) or isinstance(v,float):
        if v>=tree.value:
          branch=tree.tb
        else:
          branch=tree.fb
    else:
        if v==tree.value:
          branch=tree.tb
        else:
          branch=tree.fb
    return self.classify(observation,branch)
```

Τέλος για την υλοποίηση των παραπάνω κώδικα απαραίτητα είναι τα αρχεία:

- listfortree.py : Σκοπός του είναι να δημιουργεί τη σύνδεση με τη βάση δεδομένων και να παίρνει τις πλειάδες με τα επιθυμητά χαρακτηριστικά, έτσι ώστε να γίνουν οι παραπάνω ενέργειες.
- predict.py : Σκοπός του είναι να υπολογίσει τα χαρακτηριστικά για μία νέα πρόβλεψη. Για νέα πρόβλεψη τα μόνα στοιχεία που δίνονται είναι τα ονόματα των ομάδων. Τα υπόλοιπα (ratting_diff, form_diff, home_away_form_diff, history) υπολογίζονται με την βοήθεια το predict.py.

3.4.2 Bayes Naive

Αρχικά χρειαζόμαστε μία κλάση για να υλοποιήσει τον classifier. Αυτή η κλάση θα περιέχει ότι έχει μάθει ο αλγόριθμος έως τώρα. Έτσι έχουμε τον παρακάτω κώδικα:

```
class classifier:
  def __init__(self,getfeatures):
    self.fc={}                          #Συνδυασμός (στήλη,τιμή) χαρακτηριστικού | ftresult
    self.cc={}                          #Πόσα από κάθε αποτέλεσμα(H/D/A)
    self.getfeatures=getfeatures
```

Στην συνέχεια θα υλοποιήσουμε συναρτήσεις απαραίτητες για τον υπολογισμό των πιθανοτήτων με εύκολο και γρήγορο τρόπο.

```
def incf(self,f,cat):                    #Αυξάνει το (result|(sthlh,timh))
  results={}
  self.fc.setdefault(f,{})
  self.fc[f].setdefault(cat,0)
  self.fc[f][cat]+=1

def incc(self,cat):                      #Αυξάνει τον αριθμό των result(H/D/A)
  self.cc.setdefault(cat,0)
  self.cc[cat]+=1

def fcount(self,f,cat):                  #Βρίσκει και επιστρέφει το (result|(sthlh,timh))
  if f in self.fc and cat in self.fc[f]:
    return float(self.fc[f][cat])
  return 0.0

def catcount(self,cat):                  #Επιστρέφει το πόσες φορές εμφανίζεται το ζητούμενο result
  if cat in self.cc:
    return float(self.cc[cat])
  return 0

def totalcount(self):                    #Επιστρέφει το μέγεθος του συνόλου δεδομένων
  return sum(self.cc.values())

def categories(self):                    #Επιστρέφει τις κατηγορίες
  return self.cc.keys()
```

train(self,rows)

Η εκπαίδευση της μεθόδου γίνεται με τον παρακάτω κώδικα παίρνοντας τις πλειάδες μία μία, διαχωρίζοντας την κάθε πλειάδα στις στήλες της και καλώντας την incf αυξάνει τον μετρητή για το συγκεκριμένο χαρακτηριστικό.

```
def train(self,rows):                    #Εκπαίδευση
   for row in rows:
     features=self.getfeatures(row)
     for f in features.keys():
         self.incf(f,row[len(row)-1])    #To row[len(row)-1] περιέχει το result
     self.incc(row[len(row)-1])          #Πόσα  Home/Draw/Away έχει συνολικά
```

Στη συνέχεια θα υλοποιήσουμε μία συνάρτηση με σκοπό να υπολογίζει τις πιθανότητες για κάθε ζητούμενο χαρακτηριστικό μίας πλειάδας. Ο κώδικας είναι ο παρακάτω:

```
def fprob(self,f,cat):
  if self.catcount(cat)==0:
    return 0
  return self.fcount(f,cat)/self.catcount(cat)
```

weightedprob(self,f,cat,prf,weight=1.0,ap=0.5)

Το fprob μπορεί να βγάζει τις πιθανότητες με επιτυχία, έχει όμως ένα μικρό πρόβλημα. Το πρόβλημα εμφανίζεται όταν η μέθοδος εκτελείτε σε μικρή βάση δεδομένων, ή όταν μιλάμε για τιμές χαρακτηριστικών που σπάνια εμφανίζονται. Στις περιπτώσεις αυτές εμφανίζεται σαν αποτέλεσμα η απόλυτη πιθανότητα 1, πράγμα το οποίο μπορεί να δημιουργήσει σημαντικό πρόβλημα στις προβλέψεις μας. Για να το αποφύγουμε αυτό θα πρέπει να χρησιμοποιούμε στις περιπτώσεις αυτές μία υποθετική πιθανότητα. Έτσι με το ap(assumed probability) δημιουργούμε έναν μέσο όρο για τα δεδομένα που εμφανίστηκαν λίγες φορές και το weight δίνει το βάρος τους. Δηλαδή για weight=1 σημαίνει ότι θα πιάσει για μία λέξη.

Παράδειγμα: Αν έχω το (1,'Arsenal') 2 φορές με result 'H' και 'H' τότε από το fprob θα παίρναμε P(H|(1,'Arsenal')=1. Με τις αλλαγές που θα κάνουμε και με weight=1.0 και ap=0.5 θα έχουμε ((1*0.5)+(1*1))/(1+1)=0.75

```
def weightedprob(self,f,cat,weight=1.0,ap=0.5):
  basicprob=fprob(f,cat)                                  #Υπολογίζει το P(result|(sthlh,timh))
  totals=sum([self.fcount(f,c) for c in self.categories()]) #Πόσες φορές εμφανίστηκε το
(sthlh,timh)
  bp=((weight*ap)+(totals*basicprob))/(weight+totals)
  return bp
```

naivebayes(classifier)

Τώρα αφού τελειώσαμε τον classifier ήρθε η ώρα να αρχίσουμε την υλοποίηση του Naïve Classifier. Έως τώρα αυτό που κάναμε ήταν να παίρνουμε ένα ένα τα χαρακτηριστικά μιας πλειάδας και να υπολογίζουμε τις πιθανότητες. Τώρα θα πρέπει να υλοποιήσουμε κώδικα ο οποίος θα παίρνει σαν όρισμα ολόκληρη την πλειάδα και να μπορεί να συνδυάσει τις πιθανότητες για κάθε χαρακτηριστικό της και να βγάζει πιθανότητες για κάθε δυνατό αποτέλεσμα για ολόκληρη την πλειάδα.

Για παράδειγμα, έστω ότι έως τώρα ξέρουμε ότι το (1,'Arsenal') έχει 20% πιθανότητα να βγει 'H', δηλαδή το P((1,'Arsenal')|H)=0.2 και το (2,'Wolves') έχει 80% πιθανότητα να βγει 'H', δηλαδή το P((2,'Wolves')|H)=0.8. Τότε η πιθανότητα και για τα δύο μαζί θα ήταν 0.2*0.8=0.16, δηλαδή P((1,'Arsenal'),(2,'Wolves')|H)=0.16.

Έτσι για τον σκοπό αυτό δημιουργούμε άλλη μία κλάση, συνδεδεμένη με τη προηγούμενη(classifier) για να υλοποιήσει τις ενέργειες που αναφέραμε παραπάνω. Ο κώδικας είναι ο ακόλουθος:

```
class naivebayes(classifier):
  def __init__(self,getfeatures):
    classifier.__init__(self,getfeatures)
    self.thresholds={}
```

```
def docprob(self,item,cat):                    #Υπολογίζει την πιθανότητα για μία ολόκληρη πλειάδα
  features=self.getfeatures(item)
  p=1
  for f in features:
    p*=self.weightedprob(f,cat,self.fprob)
  return p
```

Με το docprob υπολογίζουμε την πιθανότητα P(πλειάδα|result).

Τώρα ήρθε η ώρα να υπολογίσουμε την πιθανότητα σύμφωνα με τον τύπο του Bayes Naïve classifier, P(result|πλειάδα)= P(πλειάδα|result) * P(result) / P(πλειάδα). Έτσι καταλήγουμε στον παρακάτω κώδικα που υπολογίζει την τελική πιθανότητα:

```
def prob(self,item,cat):
  catprob=float(self.catcount(cat))/self.totalcount()
  docprob=self.docprob(item,cat)
  return docprob*catprob
```

Το τελικό στάδιο είναι να υλοποιήσουμε έναν naïve Bayes classifier ο οποίος να μπορεί να αποφασίσει ποιο result να προτείνει για μία νέα πλειάδα χαρακτηριστικών. Η πιο εύκολη προσέγγιση του προβλήματος θα ήταν να υπολογίσουμε την πιθανότητα για κάθε δυνατό result και να διαλέξουμε το μεγαλύτερο. Όμως σε πολλές εφαρμογές, όπως και στην δικιά μας δεν θέλουμε το κάθε αποτέλεσμα να έχει την ίδια πιθανότητα να προταθεί. Για παράδειγμα για να προταθεί το 'A' μπορεί να θέλουμε να έχει μεγαλύτερη πιθανοτική διαφορά από τα άλλα δύο αποτελέσματα, από ότι για παράδειγμα το 'H'. Τέλος πολλές φορές *είναι προτιμότερο να λες στον classifier ότι δεν ξέρεις το αποτέλεσμα, από το να του ζητάς να σου βρει το αποτέλεσμα με την μεγαλύτερη πιθανότητα.*

thresholds

Για να αντιμετωπίσουμε όλα τα παραπάνω θέματα/προβλήματα θα δημιουργήσουμε μία ακόμα συνάρτηση η οποία θα τοποθετεί κατώφλια σε κάθε πιθανό αποτέλεσμα(H/D/A). Για κάθε νέα πλειάδα που ζητάει πρόβλεψη, για να προταθεί ένα αποτέλεσμα θα πρέπει η πιθανότητα του συγκεκριμένου αποτελέσματος να είναι μεγαλύτερη(τόσο όσο το κατώφλι που ορίσαμε επιβάλει) από τις πιθανότητες για κάθε άλλο δυνατό αποτέλεσμα. Έτσι έχουμε τον παρακάτω κώδικα:

```
def setthreshold(self,cat,t):
  self.thresholds[cat]=t

def getthresholds(self,cat):
  if cat not in self.thresholds:
    return 1.0
  return self.thresholds[cat]
```

classify

Έτσι τελικά μπορούμε να υλοποιήσουμε και την τελική συνάρτηση μέσο της οποίας μπορούμε να πάρουμε την πρόγνωση που θέλουμε. Θα υπολογίζει την πιθανότητα για κάθε δυνατό αποτέλεσμα, θα ελέγχει ποια ικανοποιεί το κατώφλι και θα την προτείνει. Πρέπει να τονίσουμε ότι όταν καλούμε την συνάρτηση ορίζουμε σαν default result το Unknown. Αυτό γιατί όπως αναφέραμε και πιο πάνω "*είναι προτιμότερο να λες στον classifier ότι δεν ξέρεις το αποτέλεσμα, από το να του ζητάς να σου βρει το αποτέλεσμα με την μεγαλύτερη πιθανότητα*". Έτσι καταλήγουμε στον παρακάτω κώδικα:

```
def classify(self,item,default=Unknown):
  probs={}
  max=0.0
  for cat in self.categories():
    probs[cat]=self.prob(item,cat)
    if probs[cat]>max:
        max=probs[cat]
        best=cat
  for cat in probs:
    if cat==best:continue
    if probs[cat]*self.getthresholds(best)>probs[best]:
        return default
  return best
```

Τέλος σημαντική είναι παρουσία των παρακάτω αρχείων/συναρτήσεων για την υλοποίηση των κωδίκων που παρουσιάστηκαν παραπάνω:

listfortree : Δημιουργεί την σύνδεση με τη βάση δεδομένων και δημιουργεί μία λίστα με πλειάδες από τους αγώνες που είναι αποθηκευμένοι στην βάση δεδομένων, με τα χαρακτηριστικά που επιθυμούμε.

NewPredict : Για τους νέου αγώνες που ζητάνε πρόβλεψη, προϋπολογίζει όλα τα απαραίτητα χαρακτηριστικά (ratting_diff, from_diff, home_away_form_diff, history).

algorithm_manual : Δημιουργεί την σύνδεση και τραβάει από την βάση δεδομένων τα απαραίτητα δεδομένα, σύμφωνα με τις ομάδες που επέλεξε ο χρήστης.

Views.py : Αποτελεί ένα από τα πιο σημαντικά αρχεία του project. Σκοπός του είναι να υπολογίζει και να τροφοδοτεί με όλα τα απαραίτητα δεδομένα το κάθε URL. Θα μπορούσαμε να πούμε ότι είναι ο εγκέφαλος του project, κι αυτό γιατί οργανώνει και συνδυάζει όλα τα αρχεία και τις συναρτήσεις μαζί, φορτώνοντας παράλληλα και όλες τις απαραίτητες βιβλιοθήκες, έτσι ώστε να έχουμε το τελικό αποτέλεσμα που θα παρουσιάσουμε στο κεφάλαιο 4. Ο κώδικας του συγκεκριμένου αρχείου είναι ο ακόλουθος.

```
from django.shortcuts import render_to_response, get_object_or_404
from project.stats.models import england
from django.http import HttpResponse
from django import forms
from django.forms.models import ModelForm
from bayespredict import *
from treepredict import *
from algorithm_manual import *
from NewEntry import *
from bayesprobs import *

class ContactForm(forms.Form):
   hometeam = forms.CharField()
   awayteam = forms.CharField()
teams_query=[]
def index(request):
   teams = england.objects.all().order_by('hometeam')
   for team in teams:
     if team.hometeam in teams_query:
         continue
     else:
         teams_query.append(str(team.hometeam))
   return render_to_response('stats/index.html', {'teams_query': teams_query})

def algorithm_manual(request):
   if request.method == 'POST':
     form = ContactForm(request.POST)
     if form.is_valid():
         hometeam = form.cleaned_data['hometeam']
         awayteam = form.cleaned_data['awayteam']
         temp=for_algorithm_manual()
         list_for_history=temp.show_history(hometeam,awayteam)
         list_for_home_at_home=temp.show_hthata(hometeam,1)
         list_for_away_at_away=temp.show_hthata(awayteam,2)
         temp.close()
         p=NewData()
         hometeam_rating=p.change_rating(hometeam)
         awayteam_rating=p.change_rating(awayteam)
         rating_diff=hometeam_rating-awayteam_rating
         rating_diff=rating_diff*(100/3)
         hometeam_form=p.change_team_form(hometeam)
         awayteam_form=p.change_team_form(awayteam)
         form_diff=hometeam_form-awayteam_form
         form_diff=form_diff*(100/9)
         htform_at_home=p.change_htform_at_home(hometeam)
         atform_at_away=p.change_atform_at_away(awayteam)
         home_away_form_diff=htform_at_home-atform_at_away
         home_away_form_diff=home_away_form_diff*(100/15)
         history=p.change_past_games(hometeam,awayteam)
         p.close()
         p1=run_for_probs()
         probs=p1.run_bayes(hometeam,awayteam)
```

```
        prob_home=round((probs['H']*100)/(probs['H']+probs['D']+probs['A']),2)
        prob_draw=round((probs['D']*100)/(probs['H']+probs['D']+probs['A']),2)
        prob_away=round((probs['A']*100)/(probs['H']+probs['D']+probs['A']),2)
        return render_to_response('stats/algorithm_manual.html',{'teams_query': teams_query,
                                                                 'hometeam':hometeam,
                                                                 'awayteam':awayteam,
                                                                 'list_for_history':list_for_history,

'list_for_home_at_home':list_for_home_at_home,

'list_for_away_at_away':list_for_away_at_away,
                                                                 'rating_diff':rating_diff,
                                                                 'form_diff':form_diff,

'home_away_form_diff':home_away_form_diff,
                                                                 'history':history,
                                                                 'prob_home':prob_home,
                                                                 'prob_draw':prob_draw,
                                                                 'prob_away':prob_away,})
def  algorithm_bayes(request):
    temp=run_them()
    results=temp.run_bayes()
    unknown="Unknown"
    return render_to_response('stats/algorithm.html', {'teams_query': teams_query,
                                                       'results' : results ,
                                                       'unknown' : unknown})

def algorithm_cart(request):
    temp=run_for_cart()
    results=temp.run_cart()
    return render_to_response('stats/algorithm_cart.html', {'teams_query': teams_query,
                                                            'results' : results})
```

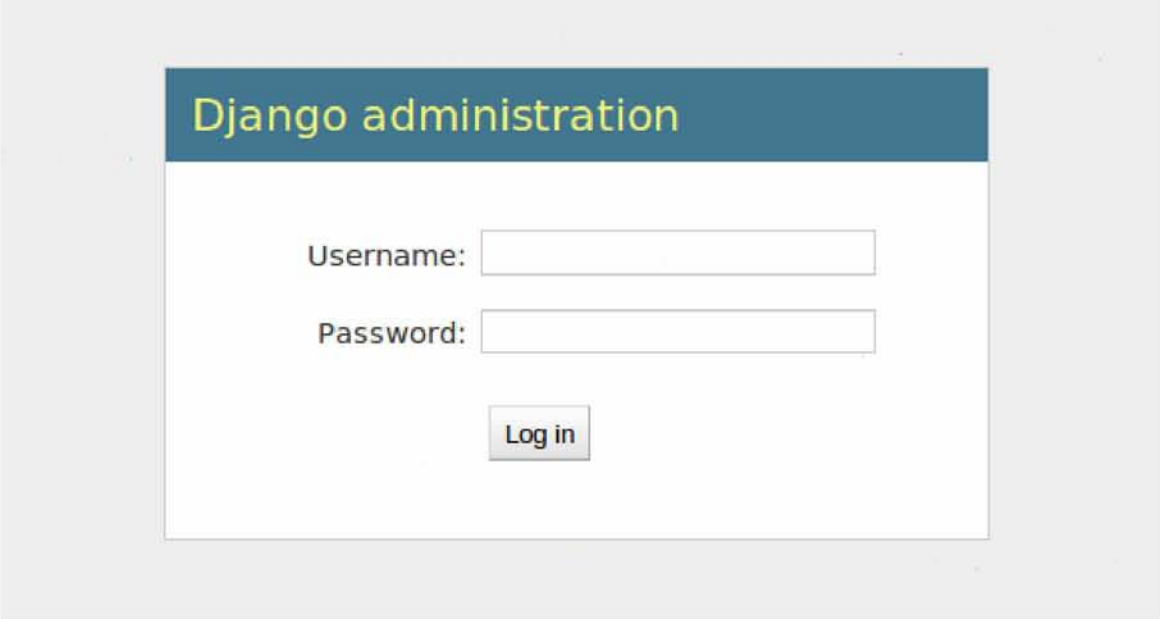
4. Η εφαρμογή στην πράξη

Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει λεπτομερής παρουσίαση της εφαρμογής που υλοποιήθηκε στα πλαίσια της πτυχιακής εργασίας. Πιο συγκεκριμένα θα παρουσιαστούν οι λειτουργίες που η εφαρμογή προσφέρει στο χρήστη και πως αυτές υλοποιούνται, αλλά και τις λειτουργίες που προσφέρονται στους διαχειριστές της εφαρμογής(administrators).

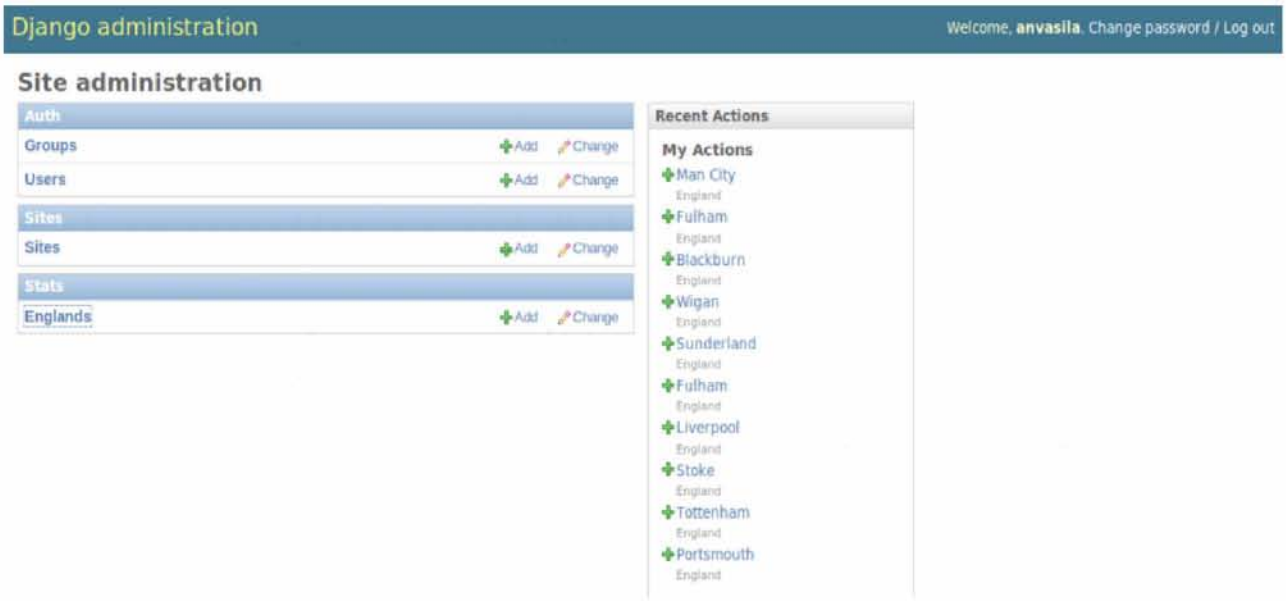
4.1. Η εφαρμογή από τη σκοπιά του administrator

Ο administrator για να μπορέσει να συνδεθεί θα πρέπει, όπως είναι αναμενόμενο, να κάνει log in με το username και το password του. Η καρτέλα σύνδεσης είναι η ακόλουθη:



Εικόνα 4.1 Είσοδος Διαχειριστή

Από την στιγμή που η σύνδεση γίνει με επιτυχία ακολουθεί η κεντρική σελίδα της εφαρμογής για τον διαχειριστή απ' όπου μπορεί να δει τα διαθέσιμα πρωταθλήματα που υπάρχουν στην βάση δεδομένων, να διαλέξει όποιο επιθυμεί από αυτά και να εκτελέσει ότι αλλαγές θέλει. Ακόμα έχει την δυνατότητα να δει τους χρήστες που έχουν εγγραφεί στην εφαρμογή με επιτυχία και να κάνει αλλαγές στον λογαριασμό τους, να σβήσει ή να δημιουργήσει καινούριο χρήστη. Τέλος, όπως φαίνεται και από την εικόνα που ακολουθεί υπάρχει και μία στήλη η οποία παρουσιάζει τις τελευταίες αλλαγές που έχουν γίνει στην εφαρμογή Η κεντρική σελίδα του διαχειριστή της εφαρμογής είναι η ακόλουθη:



Εικόνα 4.2 Αρχική καρτέλα μετά την είσοδο

Όταν ο διαχειριστής επιλέξει το πρωτάθλημα που επιθυμεί του ανοίγει νέα καρτέλα που του παρουσιάζει όλους τους αγώνες του συγκεκριμένου πρωταθλήματος με όλα τα χαρακτηριστικά του αγώνα διαθέσιμα για ανάγνωση, μελέτη, ή και αλλαγή. Θα πρέπει να τονιστεί η αυξημένη προσοχή που δείξαμε στο ευαίσθητο κομμάτι της αναζήτησης ενός αγώνα. Έτσι προσφέρονται οι παρακάτω αναζητήσεις:

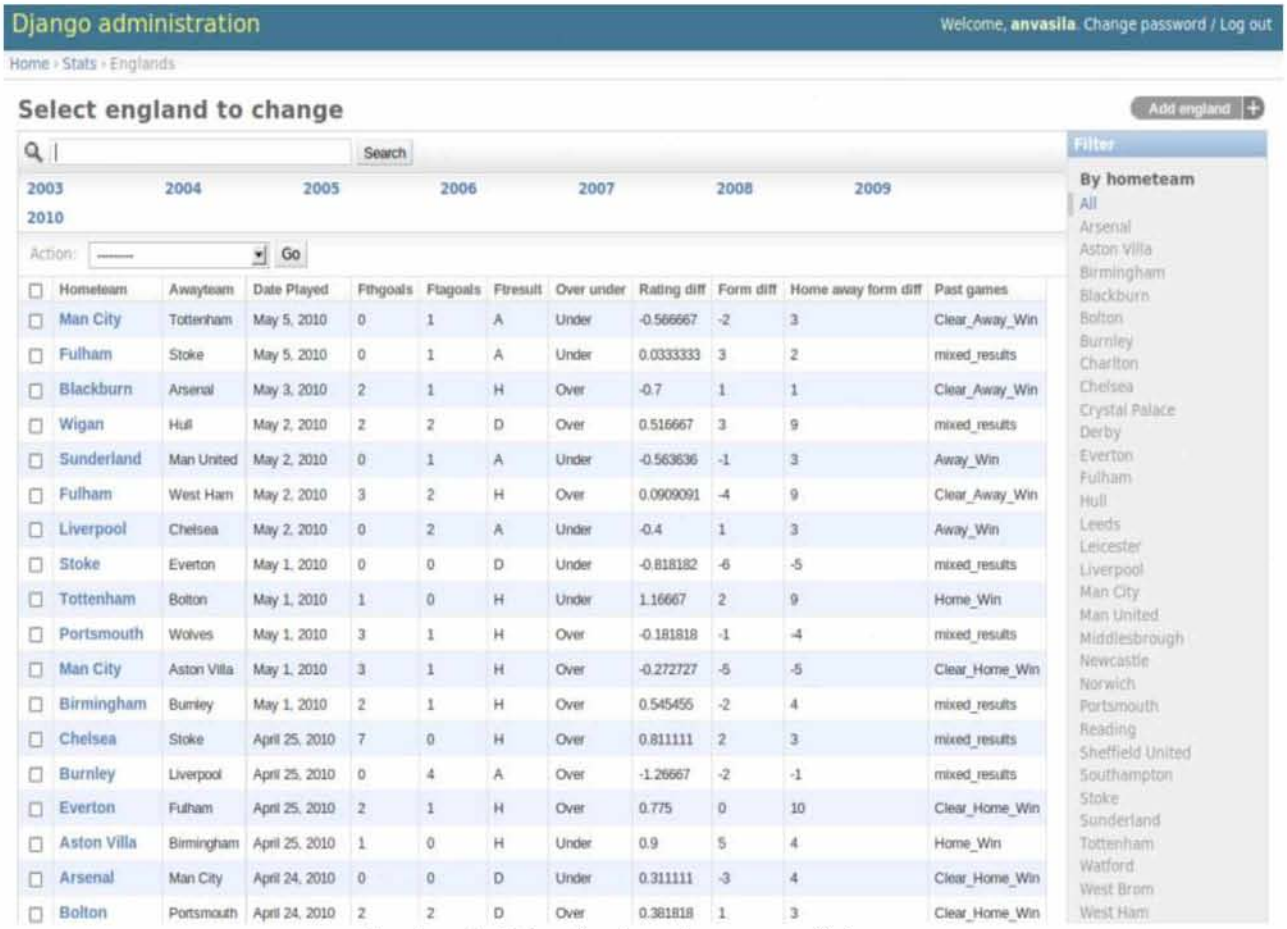
1. Αναζήτηση σύμφωνα με την ημερομηνία διεξαγωγής του αγώνα. Ο διαχειριστής επιλέγει αρχικά το έτος διεξαγωγής του αγώνα. Μόλις το επιλέξει του εμφανίζονται όλοι οι αγώνες που πραγματοποιήθηκαν το συγκεκριμένο έτος, και στο πεδίο της αναζήτησης με βάση την ημερομηνία διεξαγωγής του εμφανίζονται οι αντίστοιχοι μήνες στους οποίους έχουν καταχωρηθεί αγώνες, σε περίπτωση που θέλει να κάνει μία πιο εστιασμένη αναζήτηση. Με τον ίδιο τρόπο, αφού διαλέξει μήνα, μπορεί να συνεχίσει σε συγκεκριμένη μέρα.

Εικόνα 4.3 Αναζήτηση με βάση την ημερομηνία διεξαγωγής του αγώνα

2. Αναζήτηση σύμφωνα με την γηπεδούχο ομάδα. Στην δεξιά πλευρά της καρτέλας του πρωταθλήματος εμφανίζεται μία λίστα με όλες τις ομάδες που υπάρχουν στην βάση δεδομένων για το συγκεκριμένο πρωτάθλημα. Ο χρήστης μπορεί να επιλέξει όποια ομάδα θέλει και να δει τους αγώνες που έχει πραγματοποίηση στην έδρα της.
3. Αναζήτηση σύμφωνα με την φιλοξενούμενη ομάδα. Κάτω από την λίστα που

αναλύσαμε προηγουμένως υπάρχει μία πανομοιότυπη λίστα με τις ομάδα του πρωταθλήματος. Με την επιλογή μίας ομάδα, μας εμφανίζονται όλοι οι αγώνες που η συγκεκριμένη ομάδα πραγματοποίηση σαν φιλοξενούμενη.

4. Συνδυασμένη αναζήτηση. Μπορούν οι παραπάνω αναζητήσεις να συνδυαστούναι με γρήγορο και εύκολο τρόπο, έτσι ώστε να μπορέσει η εφαρμογή να εξυπηρετήσει τον διαχειριστή με τον καλύτερο και γρηγορότερο τρόπο στις αλλαγές που επιθυμεί να εκτελέσει.



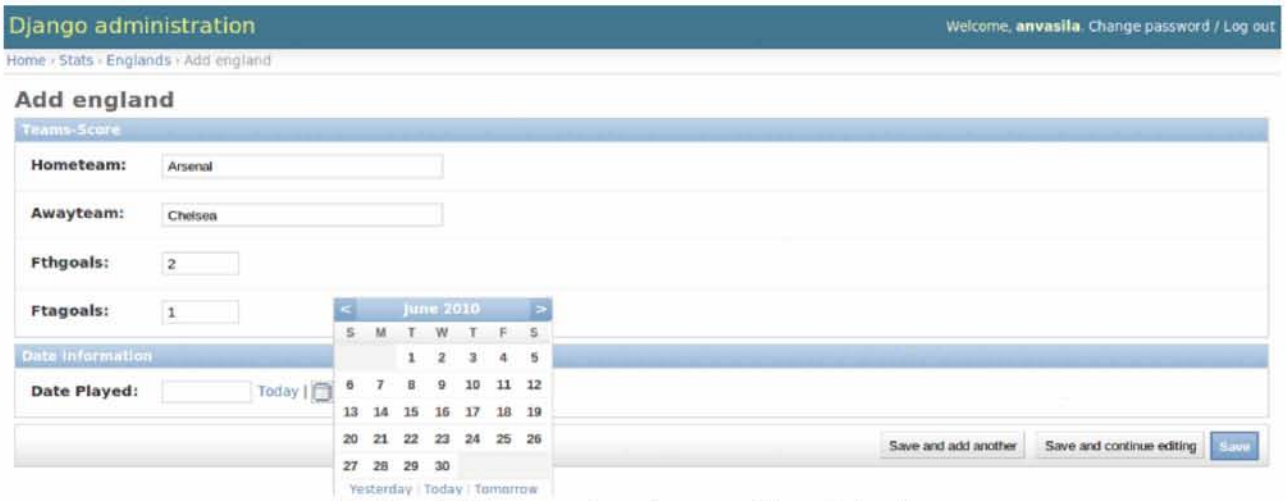
Εικόνα 4.4 Συνολική εικόνα πρωταθλήματος

Τέλος όπως φαίνεται και από την εικόνα 4.4 σε κάθε πρωτάθλημα υπάρχει ένα κουμπί για την προσθήκη νέου αγώνα στο συγκεκριμένο πρωτάθλημα. Για την ασφάλεια και την εγκυρότητα της βάσης δεδομένων η δυνατότητα αυτή δίνατε μόνο στους διαχειριστές της εφαρμογής. Έτσι μόλις ο διαχειριστής πατήσει το κουμπί "Add England" θα βρεθεί στην καρτέλα εισαγωγής νέου αγώνα στη βάση δεδομένων όπως φαίνεται στην εικόνα 4.5.

Θα πρέπει να τονιστεί ότι ο διαχειριστής αρκεί να προσθέσει τα ονόματα των ομάδων που πραγματοποιούν τον συγκεκριμένο αγώνα, το τελικό αποτέλεσμα του αγώνα και την ημερομηνία. Τα υπόλοιπα χαρακτηριστικά υπολογίζονται και τοποθετούνται στην βάση δεδομένων αυτόματα, μετά από την επιλογή Save.

Τέλος για την επιλογή της ημερομηνίας, πέρα από το κουμπί "Today" που τοποθετεί αυτόματα την τρέχουσα ημερομηνία, υπάρχει και ημερολόγιο για την ευκολότερη και

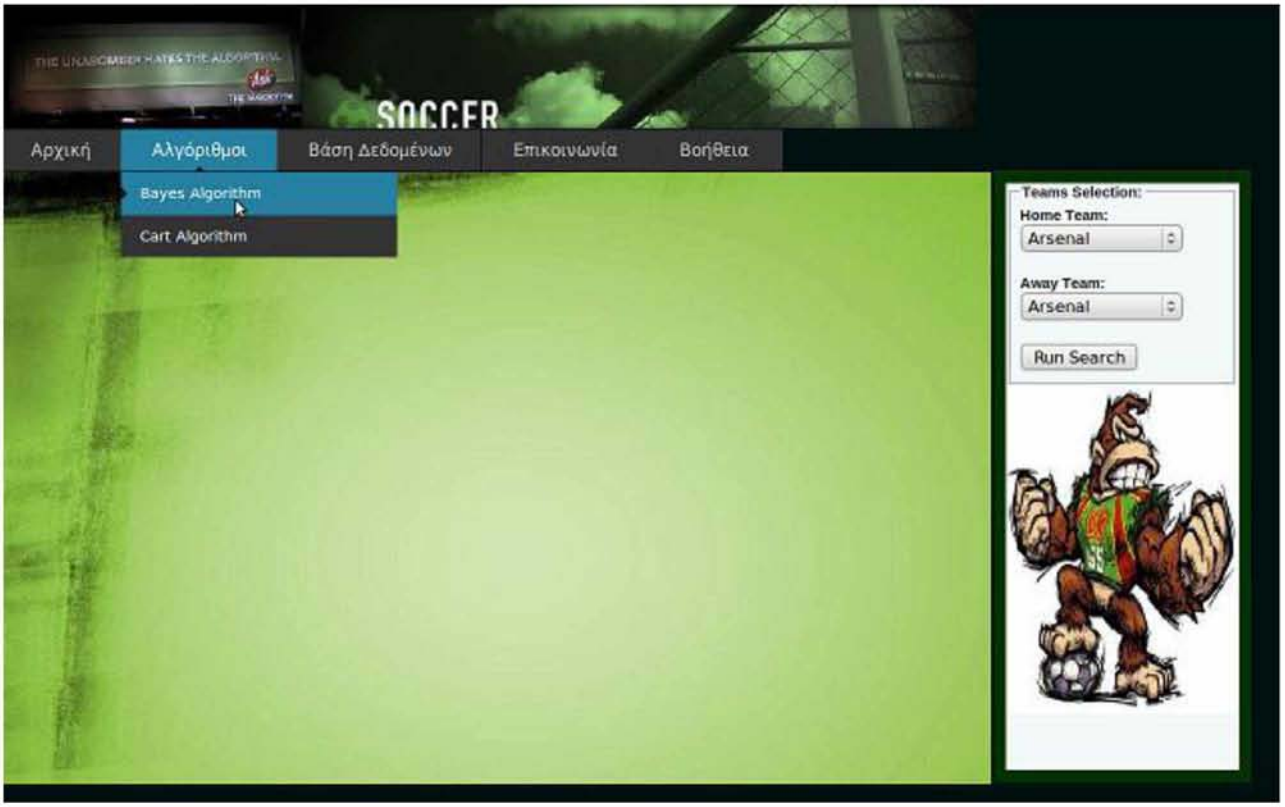
γρηγορότερη επιλογή ημερομηνίας, αλλά και για την αποφυγή λαθών κατά την διαδικασία της καταχώρησης(πχ να περαστεί η ημερομηνία με διαφορετική σειρά από την επιθυμητή, D/M/Y αντί για Y/M/D)



Εικόνα 4.5 Εισαγωγή αγώνα στη βάση δεδομένων

4.2 Η εφαρμογή για τον χρήστη

Η εφαρμογή υλοποιήθηκε, όπως αναφέραμε και στο κεφάλαιο 3 με τη βοήθεια της Django. Έχει υλοποιηθεί με τη μορφή ιστοσελίδας και όλες οι ενέργειες πραγματοποιούνται στον server. Η μορφή του ιστότοπου είναι αυτή που φαίνεται στην εικόνα 4.6.



Εικόνα 4.6 Αρχική σελίδα της εφαρμογής

Η εφαρμογή προσφέρει στον χρήστη τρεις διαφορετικές εναλλακτικές λειτουργίες προσπαθώντας με αυτό το τρόπο να καλύψει όσο το δυνατόν περισσότερες από τις ανάγκες του χρήστη. Οι λειτουργίες αυτές είναι οι ακόλουθες:

1. Πρόβλεψη με τη μέθοδο CART
2. Πρόβλεψη με τη μέθοδο του Bayes
3. Προβολή πιθανοτήτων για κάθε δυνατό αποτέλεσμα, χρησιμοποιώντας τον αλγόριθμο του Bayes, αντιπαραβάλλοντας όλα τα στατιστικά στοιχεία, βάση των οποίων καταλήξαμε στις συγκεκριμένες πιθανότητες.

4.2.1 Πρόβλεψη CART

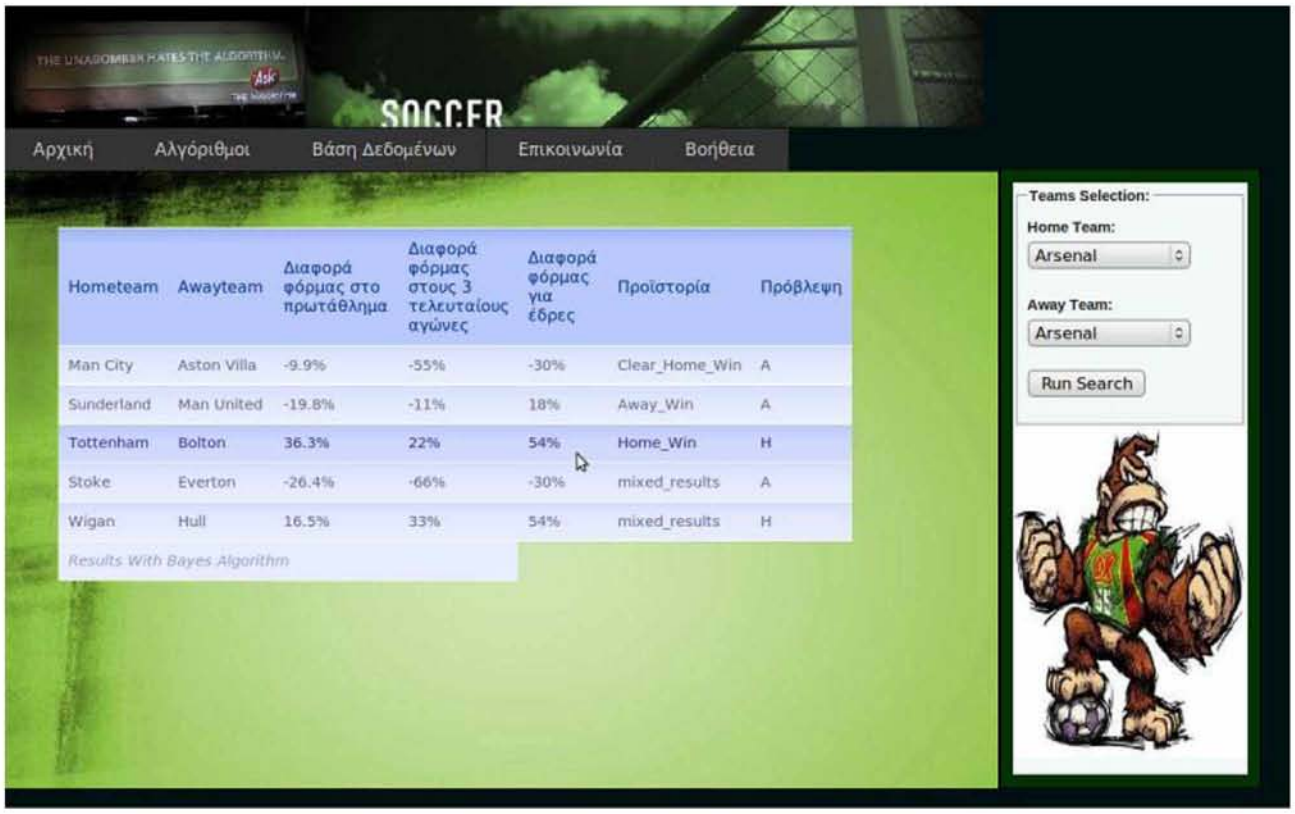
Από το βασικό μενού ο χρήστης μπορεί να δει τις προβλέψεις που κάνει ο αλγόριθμος CART για κάθε αγώνα τις ερχόμενης αγωνιστικής. Αξίζει να τονίσουμε ότι ο CART δίνει προβλέψεις για όλους τους αγώνες, αφού η λειτουργικότητα του δεν βασίζεται σε πιθανότητες όπως ο Bayes για να μπορέσουμε να κόψουμε προβλέψεις που δεν είναι τόσο σίγουρες πιθανοτικά. Έτσι όταν ο χρήστης επιλέξει τον αλγόριθμο CART θα πάρει σαν αποτέλεσμα τον πίνακα που φαίνεται στην εικόνα 4.7.

Εικόνα 4.7 Ο αλγόριθμος CART

Λόγου του ότι τα διάφορα χαρακτηριστικά δεν έχουν κοινό πεδίο τιμών, και αν παρουσιάζονταν όπως είναι στην βάση δεδομένων δεν θα έδιναν με σαφήνεια τις απαραίτητες πληροφορίες, κανονικοποιήθικαν με βάση το 100. Έτσι καταλήγουμε στα αποτελέσματα που φαίνονται στην εικόνα 4.7

4.2.2 Πρόβλεψη με Bayes Naive

Από το βασικό μενού ο χρήστης μπορεί να δει τις προβλέψεις που κάνει ο αλγόριθμος Bayes για κάθε αγώνα τις ερχόμενης αγωνιστικής. Αξίζει να τονίσουμε ότι ο Bayes λόγο του ότι η λειτουργικότητα του βασίζεται στον υπολογισμό πιθανοτήτων μπορεί και αποφασίζει ποια απ' όλες τις προβλέψεις θα προτείνει. Έτσι οι προβλέψεις με τον αλγόριθμο του Bayes μπορεί να είναι μικρότερες σε αριθμό, αλλά θεωρητικά είναι πιο σίγουρες σε σχέση με αυτές του CART. Έτσι όταν ο χρήστης επιλέξει τον αλγόριθμο του Bayes θα πάρει σαν αποτέλεσμα τον πίνακα που φαίνεται στην εικόνα 4.8.



Εικόνα 4.8 Ο αλγόριθμος του Bayes Naive

4.2.3. Η manual έκδοση του Bayes σε συνδυασμό με Search

Η λειτουργία αυτή παρέχεται από το box που βρίσκεται στην δεξιά πλευρά την εφαρμογής(εικόνα 4.9). Ο χρήστης μπορεί να επιλέξει από μία λίστα με όλες τις διαθέσιμες ομάδες την γηπεδούχο ομάδα που τον ενδιαφέρει και στην συνέχεια την φιλοξενούμενη. Από γίνεται αντιληπτό ότι η λειτουργία αυτή εξετάζει ένα αγώνα την φορά παρέχοντας πληροφορίες για τον αγώνα αυτό και μόνο. Αφού γίνει η επιλογή των ομάδων, ο χρήστης πατάει στο button "Run Search" και του παρουσιάζονται οι πληροφορίες που θα αναλύσουμε παρακάτω.

Εικόνα 4.9 Εκτέλεση της manual έκδοσης του Bayes

Μόλις γίνει η επιλογή ομάδων και γίνει submit από το κουμπί «Run Search» η εφαρμογή παρουσιάζει στον χρήστη τις παρακάτω πληροφορίες.

Ένας πίνακας που παρέχει τις συνολικές πληροφορίες για τον συγκεκριμένο συνδυασμό ομάδων που επιλέχτηκε, όπως φαίνεται στην εικόνα 4.10.

Hometeam	Awayteam	Διαφορά στο πρωτάθλημα	Διαφορά φόρμας	Διαφορά έδρας	Προϊστορία
Arsenal	Bolton	27.5%	-33%	42%	Clear_Home_Win

Στατιστικά στοιχεία για Arsenal VS Bolton

Εικόνα 4.10 Χαρακτηριστικά του αγώνα που επιλέχθηκε

Ένας πίνακας που με τη βοήθεια του αλγόριθμου Bayes δίνει τις πιθανότητες για κάθε δυνατό αποτέλεσμα (Home / Draw / Away) όπως φαίνεται στην εικόνα 4.11.

Hometeam	Awayteam	Πιθανότητα για Home	Πιθανότητα για Draw	Πιθανότητα για Away
Arsenal	Bolton	81.0%	15.37%	3.64%

Πιθανότητες για H/D/A για Arsenal VS Bolton

Εικόνα 4.11 Προβολή πιθανοτήτων για κάθε δυνατό αποτέλεσμα

Ακολουθούν 2 πίνακες που παρουσιάζουν τους 3 τελευταίους αγώνες από την κάθε ομάδα δείχνοντας έτσι το πώς παράχθηκε το αποτέλεσμα που περιγράφεται στην μεταβλητή form_diff. Οι πίνακες έχουν τη μορφή που παρουσιάζεται στην εικόνα 4.12.

Hometeam	Awayteam	Date	Home Goals	Away Goals	Result	Over Under
Tottenham	Arsenal	2010-04-14	2	1	H	Over
Wigan	Arsenal	2010-04-18	3	2	H	Over
Arsenal	Man City	2010-04-24	0	0	D	Under

Last 3 Match for Arsenal

Εικόνα 4.12 Προβολή των 3 τελευταίων αγώνων της μίας ομάδος

Στην συνέχεια παρουσιάζονται ένας πίνακας για την πρώτη ομάδα που επιλέχθηκε με τους 5 τελευταίους αγώνες που έδωσε σαν γηπεδούχος, όπως φαίνεται στην εικόνα 4.13 και ένας πίνακας με τους 5 τελευταίους αγώνες σαν φιλοξενούμενης της δεύτερη ομάδα που επιλέχθηκε όπως φαίνεται στην εικόνα 4.14. Στους πίνακες παρουσιάζεται το τελικό αποτέλεσμα, η ημερομηνία διεξαγωγής του αγώνα, τα τέρματα που πέτυχαν οι κάθε ομάδα και το αν ο αγώνας έληξε σαν Over ή Under.

Date	Hometeam	Awayteam	Home Goals	Away Goals	Result	Over / Under
2010-02-20	Arsenal	Sunderland	2	0	H	Under
2010-03-06	Arsenal	Burnley	3	1	H	Over
2010-03-20	**Arsenal**	**West Ham**	**2**	**0**	**H**	**Under**
2010-04-03	Arsenal	Wolves	1	0	H	Under
2010-04-24	Arsenal	Man City	0	0	D	Under
Last 5 match for Arsenal In Home						

Εικόνα 4.13 Οι προηγουμένοι αγώνες της 1ης ομάδας στα εντός έδρας

Date	Hometeam	Awayteam	Home Goals	Away Goals	Result	Over / Under
2010-03-06	West Ham	Bolton	1	2	A	Over
2010-03-09	Sunderland	Bolton	4	0	H	Over
2010-03-20	**Everton**	**Bolton**	**2**	**0**	**H**	**Under**
2010-04-13	Chelsea	Bolton	1	0	H	Under
2010-04-17	Stoke	Bolton	1	2	A	Over
Last 5 match for Bolton In Away						

Εικόνα 4.14 Οι αγώνες της 2ης ομάδας που επιλέχθηκε στα εκτός έδρας

Και τέλος παρουσιάζεται σε πίνακα η προϊστορία των δύο ομάδων στα προηγούμενα πέντε χρόνια, αν υπάρχουν αγώνες φυσικά. Η πληροφορίες παρουσιάζονται όπως φαίνεται στην εικόνα 4.15.

Date	Arsenal	Bolton	Result	Over / Under
2006-02-11	1	1	D	Under
2007-04-14	2	1	H	Over
2007-10-20	**2**	**0**	**H**	**Under**
2009-01-10	1	0	H	Under
2010-01-20	4	2	H	Over
History For Arsenal - Bolton				

Εικόνα 4.15 Η προϊστορία των δύο ομάδων που επιλέχθηκαν.

5. Αποτελέσματα - Παρατηρήσεις

Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν κάποιοι από τους πειραματισμούς που έγιναν μετά την υλοποίηση της εφαρμογής, συνοδευόμενοι από σχόλια και παρατηρήσεις. Ακόμα θα γίνει συνολική αποτίμηση της δουλειάς που έγινε στα πλαίσια της πτυχιακής εργασίας.

5.1 Ανασκόπηση στόχων και αναπαράστασης

Όπως έχει αναφερθεί και στα προηγούμενα κεφάλαια, στόχος της πτυχιακής ήταν η δημιουργία μίας εφαρμογής που θα υποστήριζε την πρόγνωση αποτελεσμάτων αγώνων ποδοσφαίρου. Αρχικός στόχος ήταν η επίτευξη ενός ποσοστού επιτυχίας της τάξης του 60%. Όπως θα δούμε στην συνέχεια το ποσοστό αυτό σε κάποιες τον περιπτώσεων επιτεύχθηκε.

Να θυμηθούμε σ' αυτό το σημείο ποιες ήταν οι αρχικές μας επιλογές όσο αφορά τους παράγοντες που επηρεάζουν την έκβαση ενός ποδοσφαιρικού αγώνα:

- Η θέση της γηπεδούχου ομάδας στον πίνακα της βαθμολογίας.
- Η θέση της φιλοξενούμενης ομάδας στον πίνακα της βαθμολογίας.
- Η φόρμα της γηπεδούχου ομάδας.
- Η φόρμα της φιλοξενούμενης ομάδας.
- Η φόρμα της γηπεδούχου ομάδας στην έδρα της.
- Η φόρμα της φιλοξενούμενης ομάδας στα εκτός έδρας παιχνίδια.
- Η προϊστορία ανάμεσα στις δύο ομάδες.

Που αναπαριστώνται από τις αντίστοιχες μεταβλητές του πίνακα stats_England:

- htrating
- atrating
- home_team_form
- away_team_form

- htform_at_home
- atform_at_away
- past_games

Να θυμίσουμε ότι οι παραπάνω στήλες, εκτός από την past_games αντικαθίστανται ανά δύο από τις διαφορές τους, για εξοικονόμηση χρόνου και πολυπλοκότητας κατά τη διάρκεια της εκτέλεσης του αλγορίθμου. Έτσι τελικά καταλήξαμε στις ακόλουθες στήλες, πάνω στις οποίες υλοποιήθηκε όλη η εφαρμογή:

- hometeam
- awayteam
- rating_diff
- form_diff
- home_away_form_diff
- past_games

Ακόμα να τονιστεί ότι η εξαρτημένη μεταβλητή του προβλήματος, που είναι το αποτέλεσμα ενός αγώνα αναπαρίσταται από την στήλη ***result***.

Τέλος να θυμίσουμε ότι οι αλγόριθμοι που εξετάστηκαν και προσφέρονται από την εφαρμογή είναι οι:

- CART(Decision Trees)
- Bayes Naive(Bayesian Learning)

Η ανάλυση που θα ακολουθήσει θα αφορά τόσο σε ποσοστά επιτυχίας του κάθε αλγορίθμου, με όλες τις δυνατές παραλλαγές και τροποποιήσεις του(κατώφλια, τρόπος υλοποίησης κτλ), αλλά και σε χρονικές απαιτήσεις, τόσο για την διαδικασία της μάθησης(training), αλλά και για την διαδικασία της πρόγνωσης.

Τρόπος μέτρησης αποτελεσματικότητας

Η μέτρηση των αποτελεσμάτων έγινε με κώδικα, ο οποίος χωρίζει τυχαία το σύνολο δεδομένων σε δύο μέρη, στο σύνολο εκπαίδευσης(training set) και στο σύνολο επαλήθευσης(test set). Ο διαχωρισμός έγινε σε 66% training set και 33% test set επί του συνόλου της βάσης δεδομένων. Η διαδικασία αυτή εκτελέστηκε για την κάθε εξεταζόμενη περίπτωση 10 φορές για τον αλγόριθμο CART και 20 φορές για τον Bayes. Η λόγος των επαναλήψεων είναι για να μπορέσει να βγει ένας μέσος όρος, αλλά και μία μέγιστη τιμή επιτυχίας. Η αιτία των διαφορετικών επαναλήψεων(10 και 20) ήταν ο εκτελέσιμος χρόνος που απαιτεί ο CART σε σχέση με τον Bayes. Μεγαλύτερη ανάλυση θα υπάρξει στις ενότητες που ακολουθούν.

5.2 Ανάλυση του αλγόριθμου CART

Όπως αναφέραμε και στα προηγούμενα κεφάλαια ο αλγόριθμος CART βασίζεται στην δημιουργία ενός δυαδικού δένδρου και η λειτουργικότητά του(εύρεση αποτελέσματος) υλοποιείται με τη λογική True/False προσπελάζοντας το δένδρο που δημιουργήθηκε προηγουμένως, μέχρι να καταλήξουμε σε κάποιο φύλλο, απ' όπου θα πάρουμε και το αποτέλεσμα που αναζητούμε.

Από τα παραπάνω μπορούμε να καταλάβουμε ότι δεν υπάρχουν περιθώρια να παραμετροποιήσουμε την διαδικασία αναζήτησης της λύσης. Με το μόνο που μπορούμε να πειραματιστούμε είναι η διαδικασία δημιουργίας του δένδρου.

Ζητήματα κλειδί για την δημιουργία του δένδρου είναι τα ακόλουθα:

1. Τρόπος επιλογή του χαρακτηριστικού για τον διαχωρισμό. Οι εξεταζόμενες μέθοδοι είναι:
    - Gini Impurity
    - Entropy
2. Η διαδικασία του κλαδέματος του δένδρου(prune tree). Οι αλλαγές που μπορούν να γίνουν είναι στην επιλογή της τιμής για το κατώφλι του κέρδους που έχουμε από τον εκάστοτε διαχωρισμό των δεδομένων. Το πεδίο τιμών είναι [0,1].

5.2.1 Split method : Entropy

Όπως αναφέραμε και στο δεύτερο κεφάλαιο η εντροπία χαρακτηρίζει την ανομοιογένεια ενός συνόλου παραδειγμάτων και εκφράζει το αναμενόμενο μήκος κωδικοποίησης πληροφορίας μετρημένο σε bit. Έστω ένα σύνολο παραδειγμάτων S. Αν Pc είναι το ποσοστό των παραδειγμάτων του S που ανήκουν στην κατηγορία c ε C, τότε η εντροπία E(S) του συνόλου είναι :

$$E(S) = \sum_{c \in C} -\mathrm{p}_c \log 2 \mathrm{p}_c$$

Το κέρδος πληροφορίας μετράει τη μείωση της εντροπίας που θα προκληθεί αν χωριστούν τα παραδείγματα ενός συνόλου με βάση κάποιο χαρακτηριστικό, και μας δίνει τη δυνατότητα να πειραματιστούμε με τα κατώφλια. Έτσι με την διαδικασία του κλαδέματος μπορούμε να ορίσουμε το ελάχιστο κέρδος που θέλουμε να έχουμε, για να γίνει ένας

διαχωρισμός δεδομένων κατά τη διαδικασία δημιουργίας του δένδρου. Όπως αναφέραμε και πριν το πεδίο τιμών του κέρδους είναι [0,1].

Τα αποτελέσματα του πειραματισμού παρουσιάζονται στον πίνακα 5.1

Κατώφλι Κέρδους από διαχωρισμό δεδομένων	Ποσοστό επιτυχίας (Μέσος Όρος)	Ποσοστό επιτυχίας (max)	Χρόνος Πρόβλεψης (για έναν αγώνα)	Συνολικός Χρόνος (Δημιουργία Δένδρου, Κλάδεμα, Πρόβλεψη)
0.0	48,00%	50,00%	$6.5*10^{-5}$	13,3
0.2	48,00%	51,00%	$6.5*10^{-5}$	13,3
0.3	48,00%	51,00%	$6.3*10^{-5}$	13,2
0.4	48,00%	53,00%	$6.2*10^{-5}$	13,15
0.5	49,00%	53,00%	$5.8*10^{-5}$	13
0.6	49,00%	52,00%	$5.7*10^{-5}$	12,6
0.7	49,00%	52,00%	$5.7*10^{-5}$	12,25
0.8	48,00%	52,00%	$5.5*10^{-5}$	11,5
0.9	48,00%	51,00%	$5.4*10^{-5}$	10,5
1	47,00%	50,00%	$4.0*10^{-5}$	9,5

Πίνακας 5.1 Αποτελέσματα πειραματισμού με τα κατώφλια με Entropy

Ανάλυση ευστοχίας

Από τον πίνακα 5.1 βλέπουμε ότι με τη διαδικασία του κλαδέματος του δένδρου δεν έχουμε κάποια σημαντική βελτίωση στην αποτελεσματικότητα του αλγορίθμου. Ωστόσο παρατηρούμε ότι με κατώφλι κέρδους 0.5 έχουμε μία αύξηση τόσο στον μέσο όρο ευστοχίας του αλγορίθμου, όσο και στην μέγιστο ποσοστό ευστοχίας που μπορεί να πετύχει. Αυτό γίνεται διότι με τη διαδικασία του κλαδέματος του δένδρου αντιμετωπίζεται το πρόβλημα της υπερμοντελοποίησης(overfitting), δηλαδή το πρόβλημα της υπερεκπαίδευσης του δένδρου που έχει σαν αποτέλεσμα τη μεγάλη ακρίβεια πρόβλεψης στα δεδομένα εκπαίδευσης, αλλά μικρή σε νέα άγνωστα δεδομένα.

Ο λόγος της μικρής αύξησης της ευστοχίας του αλγορίθμου, ίσως να οφείλεται και στον τρόπο που αποθηκεύονται τα αποτελέσματα στα φύλλα του δένδρου και στον τρόπο της επιλογής του προτεινόμενου αποτελέσματος. Τα αποτελέσματα στα φύλλα του δένδρου αποθηκεύονται με τη μορφή {'D':8 , 'A':1} και επιλέγεται αυτό με τη μεγαλύτερη τιμή. Έτσι και μετά της υλοποίηση του κλαδέματος και τις συγχώνευσης δύο φύλλων, αφού το κατώφλι κέρδους είναι μικρό, τα φύλλα που θα ενωθούν δεν θα επιφέρουν κάποια σημαντική αλλαγή στο προτεινόμενο αποτέλεσμα.

Ανάλυση χρόνου εκτέλεσης

Τέλος η διαδικασία κατωφλίου επηρεάζει τον χρόνο που απαιτείτε για μία νέα πρόβλεψη αποτελέσματος. Η διαφορά του χρόνου για μία πρόγνωση μπορεί να είναι ιδιαίτερα μικρή, της τάξης του $2\text{-}4*10^{-6}$, αλλά από ότι βλέπουμε στην τελευταία στήλη, για όλη της διαδικασία και για συνεχόμενες προβλέψεις μπορεί να επιφέρει κέρδος χρόνου μέχρι και 1 second.

Χρόνος δημιουργίας δένδρου με Entropy

Ο χρόνος που απαιτείτε για την δημιουργία του δένδρου με την μέθοδο της εντροπία είναι κατά μέσο όρο 7.2 second. Ο χρόνος αυτός είναι ανεξάρτητος από την διαδικασία κλαδέματος, αφού εκτελείται μετά τη δημιουργία του δένδρου.

5.2.2 Split Method : Gini Impurity

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο η λειτουργικότητά της Gini Impurity και τα αποτελέσματα που βγάζει βασίζονται στην λογική "αν τοποθετήσω τυχαία ένα αντικείμενο ποια η πιθανότητα να είναι λάθος". Για παράδειγμα αν έχουμε ένα σύνολο από πλειάδες από τις οποίες οι 9 έχουν result Η και η μία έχει result Α, τότε το αποτέλεσμα που θα επιστρέψει θα είναι 1/10=0.1. Αντίθετα αν έχουμε 5 με αποτέλεσμα Η και 5 με αποτέλεσμα Α θα επιστρέψει 0.5.Από τον τρόπο λειτουργίας της Gini Impurity καταλαβαίνουμε ότι όσο μικρότερος είναι ο αριθμός που επιστρέφει τόσο καλύτερο διαχωρισμό των πλειάδων έχουμε κάνει. Δηλαδή σαν καλύτερο διαχωρισμό (best split) κρατάμε αυτόν με το μικρότερο αποτέλεσμα. Ο τρόπος λειτουργίας του είναι ιδιαίτερα απλός και αυτό φαίνεται και στους χρόνους εκτέλεσης που απαιτούνται για την δημιουργία ενός δένδρου. Με τη μέθοδο αυτή το πεδίο τιμών για το κέρδος πληροφορίας είναι το [0,0.5].

Έτσι καταλήγουμε στον πίνακα 5.2 που μας παρουσιάζει τα αποτελέσματα των μετρήσεων για κάθε κατώφλι στο κέρδος πληροφορίας.

Κατώφλι Κέρδους από διαχωρισμό δεδομένων	Ποσοστό επιτυχίας (Μέσος Όρος)	Ποσοστό επιτυχίας (max)
0.0	48	51
0.2	48	51
0.3	50	52
0.4	49	51
0.5	47	50

Πίνακας 5.2 Αποτελέσματα πειραματισμού με τα κατώφλια, με Gini Impurity

Οι χρόνοι εκτέλεσης δεν παρουσιάστηκαν στον πίνακα 5.2, γιατί οι τιμές δεν διαφέρουν ιδιαίτερα από αυτές του πίνακα 5.1, έτσι ώστε να θεωρηθεί απαραίτητη η παρουσίασή τους.

Ανάλυση ευστοχίας

Από τον πίνακα 5.2 βλέπουμε και πάλι ότι με τη διαδικασία του κλαδέματος του δένδρου δεν έχουμε κάποια σημαντική βελτίωση στην αποτελεσματικότητα του αλγορίθμου. Ωστόσο παρατηρούμε ότι με κατώφλι κέρδους 0.3 έχουμε μία αύξηση τόσο στον μέσο όρο ευστοχίας του αλγορίθμου, όσο και στον μέγιστο ποσοστό ευστοχίας που αυτός μπορεί να πετύχει.

Χρόνος δημιουργίας δένδρου με Gini Impurity

Ο χρόνος που απαιτείτε για την δημιουργία του δένδρου με την μέθοδο Gini Impurity είναι κατά μέσο όρο 6.8 second. Ο χρόνος αυτός είναι ανεξάρτητος από την διαδικασία κλαδέματος, αφού εκτελείται μετά τη δημιουργία του δένδρου. Παρατηρούμε ότι η μέθοδος αυτή πέρα από την επίτευξη ελαφρός καλύτερου ποσοστού επιτυχίας, είναι και γρηγορότερη με αποτέλεσμα να μας κάνει να την επιλέξουμε για την εφαρμογή μας.

5.2.3 Σύνοψη για CART

Από τα παραπάνω διαπιστώνουμε ότι ο αλγόριθμος CART δεν μπορεί να ικανοποιήσει τον αρχικό στόχο ευστοχίας(60%). Μπορεί να φτάσει στην καλύτερη περίπτωση σε ευστοχία της τάξης του 53%, ενώ παράλληλα οι χρόνοι εκτέλεσής τους είναι ιδιαίτερα υψηλοί, δημιουργώντας πρόβλημα σε εφαρμογές όπως η παρούσα που υλοποιούνται μέσο διαδικτύου.

Τα παραπάνω προβλήματα περιορίζονται μερικός με τον αλγόριθμο Gini Impurity, ο οποίος απαιτεί και λιγότερο χρόνο εκτέλεσης, αλλά πετυχαίνει και μεγαλύτερο ποσοστό επιτυχίας πρόβλεψης.

Έτσι στην εφαρμογή μας ο αλγόριθμος CART υλοποιείται με την διαδικασία Gini Impurity και με κατώφλι κέρδους πληροφορίας το 0.3.

Στρογγυλοποίηση χαρακτηριστικών τύπου float

Σε όλους τους αλγορίθμους υπάρχει το θέμα της στρογγυλοποίησης των δεκαδικών χαρακτηριστικών. Στην περίπτωση του CART αυτό δε μπορεί να επηρεάσει την

αποτελεσματικότητα του αλγορίθμου, αφού όπως είπαμε η λειτουργικότητα του παρέχεται από μία διαδικασία True/False. Δηλαδή κατά την διαδικασία της πρόβλεψης ο αλγόριθμος προσπελάσει το δένδρο συγκρίνοντας τις αριθμητικές τιμές βάση ανισότητας. Έτσι δεν επηρεάζεται η αποτελεσματικότητα του αλγορίθμου από τον αριθμός των δεκαδικών ψηφίων των χαρακτηριστικών τύπου float.

Αντίθετα επηρεάζεται ο απαιτούμενος χρόνος εκτέλεσης του κώδικα για την δημιουργία του δένδρου. Αυτό γίνεται γιατί χωρίς στρογγυλοποίηση υπάρχουν πολλές περισσότερες τιμές για έλεγχο, κατά τη διάρκεια της εύρεσης καλύτερης τιμής χαρακτηριστικού για το διαχωρισμό των δεδομένων. Μετά από μετρήσεις που έγιναν καταλήξαμε στον πίνακα 5.3 και αποφασίστηκε η στρογγυλοποίηση των δεκαδικών χαρακτηριστικών στο πρώτο δεκαδικό ψηφίο.

Στρογγυλοποίηση	Χρόνος δημιουργίας δένδρου απόφασης με Entropy	Χρόνος δημιουργίας δένδρου απόφασης με Gini Impurity
Χωρίς στρογγυλοποίηση	20 seconds	17.8 seconds
Στρογγυλοποίηση στο 1ο δεκαδικό	7.2 seconds	6.8 seconds

Πίνακας 5.3 Σύγκριση χρόνου δημιουργίας δένδρου σε σχέση με την στρογγυλοποίηση δεκαδικών χαρακτηριστικών

5.3 Ανάλυση του αλγόριθμου Bayes Naive

Σε αντίθεση με τον αλγόριθμο CART ο αλγόριθμος του Bayes προσεγγίζει το θέμα της εξαγωγής συμπερασμάτων με βάση τις πιθανότητες. Στηρίζεται στην υπόθεση ότι οι ποσότητες που μας ενδιαφέρουν διέπονται από κατανομές πιθανοτήτων και ότι η εξαγωγή συμπερασμάτων με αυτές τις πιθανότητες σε συνδυασμό με δεδομένα που παρατηρούνται μπορούν να οδηγήσουν στη λήψη βέλτιστων αποφάσεων. Στη μάθηση κατά Bayes κάθε παράδειγμα μπορεί προοδευτικά να αυξήσει ή να μειώσει την πιθανότητα να είναι σωστή μια υπόθεση. Το γεγονός αυτό παρέχει έναν πιο ευέλικτο τρόπο μάθησης σε σχέση με τον αλγόριθμο CART που διαγράφει τελείως μια υπόθεση αν είναι ασύμβατη έστω και με ένα παράδειγμα.

Εκτός από το στάδιο της εκπαίδευσης, η μάθηση κατά Bayes δίνει ευελιξία και στο στάδιο της εφαρμογής της γνώσης. Τα μοντέλα που προκύπτουν από μια διαδικασία μάθησης κατά Bayes μπορούν να κάνουν προβλέψεις με πιθανότητα. Για παράδειγμα μια πρόβλεψη τέτοιου τύπου είναι “ο αγώνας Arsenal-Wolves θα έχει αποτέλεσμα Home με πιθανότητα 75% ”.

Από τα παραπάνω καταλαβαίνουμε ότι τα πεδία πειραματισμού είναι τα ακόλουθα:

- Στρογγυλοποίηση δεκαδικών χαρακτηριστικών. Αυτό γίνεται γιατί ο αλγόριθμος του Bayes υπολογίζει για κάθε τιμή ενός χαρακτηριστικού τις

πιθανότητες για κάθε δυνατή τιμή της εξαρτημένης μεταβλητής(result). Έτσι αν δεν στρογγυλοποιήσουμε κατάλληλα τα δεκαδικά χαρακτηριστικά, θα έχουμε πρόβλημα στην αποτελεσματικότητα του αλγορίθμου, λόγο μεγάλης διασποράς των τιμών του συγκεκριμένου χαρακτηριστικού.

- Τα κατώφλια(thresholds) για τις πιθανότητες των προτεινόμενων αποτελεσμάτων. Η λογική της τοποθέτησης κατωφλίου στη διαδικασία της πρόβλεψης, όπως εξηγήσαμε και στο 4ο κεφάλαιο βασίζεται στο ότι δεν θέλουμε ο αλγόριθμος να προτείνει ένα αποτέλεσμα, το οποίο να είναι σχεδόν ισοπίθανο με κάποιο άλλο. Αν δεν υπάρχει κάποιο δυνατό αποτέλεσμα με πιθανότητα μεγαλύτερη από το κατώφλι που ορίσαμε, είναι προτιμότερο ο αλγόριθμος να μην κάνει πρόβλεψη. Γι' αυτό και υποστηρίζεται ότι *είναι προτιμότερο να λες στον classifier ότι δεν ξέρεις το αποτέλεσμα, από το να του ζητάς να σου βρει το αποτέλεσμα με την μεγαλύτερη πιθανότητα.*

Όπως αναφέραμε και προηγουμένως, η εξαγωγή τον αποτελεσμάτων που θα ακολουθήσουν, έγιναν με κώδικα ο οποίος χωρίζει τυχαία το σύνολο της βάσης δεδομένων σε δύο σύνολα, 66% σε training set και 33% σε test set. Η διαδικασία υλοποιήθηκε με 20 επαναλήψεις για κάθε περίπτωση, για τον υπολογισμό του μέσου όρου και του μέγιστου ποσοστού ευστοχίας.

5.3.1 Χωρίς στρογγυλοποίηση δεκαδικών χαρακτηριστικών

Το πρόβλημα με την στρογγυλοποίηση όπως αναφέραμε και προηγουμένως έγκειται στο ότι ο Bayes βγάζει πιθανότητες για κάθε δυνατή τιμή ενός χαρακτηριστικού. Έτσι θα υπολογιστή διαφορετική πιθανότητα για κάθε δυνατή τιμή της εξαρτημένης μεταβλητής για το 1.05 και για το 1.050000001. Αυτό έχει σαν αποτέλεσμα τη μη σωστή εκπαίδευση του αλγορίθμου. Ακόμα είναι πολύ εύκολο να υπάρξει τιμή για την οποία θέλουμε να κάνουμε πρόβλεψη αλλά να μην έχει συναντηθεί ακόμα και άρα να μην έχει υπολογιστεί πιθανότητα για την συγκεκριμένη τιμή. Έτσι παρουσιάζεται πρόβλημα και στην διαδικασία της πρόβλεψης.

Από τα παραπάνω καταλαβαίνουμε την σημαντικότητα της στρογγυλοποίησης στους αλγορίθμους του Bayes.

Κατώφλια(thresholds)

Στην συνέχεια θα παρουσιαστούν αποτελέσματα, που δεν έχει επέλθει στρογγυλοποίηση στα δεκαδικά χαρακτηριστικά για διάφορες τιμές κατωφλιών. Τα κατώφλια είναι της μορφής (2,3,4) για τις αντίστοιχες τιμές της εξαρτημένης μεταβλητής (H,D,A). Το συγκεκριμένο παράδειγμα απαιτεί:

- Για να προταθεί το αποτέλεσμα “H” πρέπει η πιθανότητα για “H” να είναι τουλάχιστον διπλάσια από την πιθανότητα για “D” και τουλάχιστον διπλάσια από την πιθανότητα για “A”.

- Για να προταθεί το αποτέλεσμα “D” πρέπει η πιθανότητα για “D” να είναι τουλάχιστον τριπλάσια από την πιθανότητα για “H” και τουλάχιστον τριπλάσια από την πιθανότητα για “A”.
- Για να προταθεί το αποτέλεσμα “A” πρέπει η πιθανότητα για “A” να είναι τουλάχιστον τετραπλάσια από την πιθανότητα για “D” και τουλάχιστον τετραπλάσια από την πιθανότητα για “H”.

Έτσι καταλήγουμε στα αποτελέσματα που παρουσιάζονται στον πίνακα 5.4.

Κατώφλι(threshold)	Ποσοστό επιτυχίας (Μέσος Όρος)	Ποσοστό επιτυχίας (Max)
(1.0,1.0,1.0)	51	54
(1.5,1.5,1.5)	58	61
(1.5,2.0,2.0)	60	64
(2.0,2.0,2.0)	62	66
(2.0,2.5,2.5)	64	67
(2.5,2.5,2.5)	66	71
(2.5,3.0,3.0)	67	72
(3.0,3.0,3.0)	68	74

Πίνακα 5.4 Αποτελεσματικότητα Bayes χωρίς στρογγυλοποίηση για πιθανά κατώφλια.

Από τα παραπάνω είναι ξεκάθαρο ότι όσο μεγαλύτερους περιορισμούς βάζουμε στις αποδεκτές τιμές για τις προτάσεις, τόσο μεγαλύτερη ευστοχία έχει ο αλγόριθμος. Αυτό όμως δεν γίνεται χωρίς κόστος. Όσο μεγαλύτερους περιορισμούς βάζουμε τόσο λιγότεροι είναι και οι αγώνες που τελικά κάνει πρόβλεψη ο αλγόριθμος. Για παράδειγμα στην επιλογή κατωφλίου (3.0,3.0,3.0), αλγόριθμος κάνει πρόβλεψη κατά μέσο όρο στο 41% τον αγώνων. Τα υπόλοιπα τα αφήνει χωρίς πρόβλεψη. Αυτό σημαίνει ότι από τους 12 αγώνες που έχει για παράδειγμα το πρωτάθλημα της Αγγλίας σε κάθε αγωνιστική κατά μέσο όρο γίνεται πρόβλεψη στους 5. Αντίθετα στο (1.0,1.0,1.0) γίνεται πρόβλεψη στο 100% των αγώνων.

Στην εφαρμογή μας, επιλέχθηκε το κατώφλι (2.5,2.5, 2.5) το οποίο πετυχαίνει ευστοχία της τάξης του 66% κατά μέσο όρο και 71% σε μερικές των περιπτώσεων και η πρόβλεψη αποτελέσματος γίνεται κατά μέσο όρο στους 50-52% του συνόλου των αγώνων που ζητάνε πρόβλεψη.

5.3.2 Στρογγυλοποίηση στο 3ο δεκαδικό ψηφίο

Με τον ίδιο τρόπο που περιγράφθηκε παραπάνω, καταλήξαμε στα αποτελέσματα που φαίνονται στο πίνακα 5.5.

Κατώφλι(threshold)	Ποσοστό επιτυχίας (Μέσος Όρος)	Ποσοστό επιτυχίας (Max)
(1.0,1.0,1.0)	52	55
(1.5,1.5,1.5)	58	63
(1.5,2.0,2.0)	60	64
(2.0,2.0,2.0)	62	67
(2.0,2.5,2.5)	65	69
(2.5,2.5,2.5)	67	73
(2.5,3.0,3.0)	68	74
(3.0,3.0,3.0)	69	76

Πίνακα 5.5 Αποτελεσματικότητα Bayes με στρογγυλοποίηση στο 3ο δεκαδικό για πιθανά κατώφλια.

Βλέπουμε ότι σε όλες τις περιπτώσεις παρουσιάζεται μία βελτίωση της ευστοχίας του αλγορίθμου στις προβλέψεις του, σε σχέση με αυτές που έγιναν χωρίς τον στρογγυλοποίηση. Παρουσιάζεται έτσι και στην πράξη η χρησιμότητα της στρογγυλοποίησης στους αλγορίθμους που λειτουργούν με βάση τις πιθανότητες.

5.3.3 Στρογγυλοποίηση στο 1ο δεκαδικό ψηφίο

Με τον ίδιο τρόπο που περιγράφθηκε παραπάνω, καταλήξαμε στα αποτελέσματα που φαίνονται στο πίνακα 5.6.

Κατώφλι(threshold)	Ποσοστό επιτυχίας (Μέσος Όρος)	Ποσοστό επιτυχίας (Max)
(1.0,1.0,1.0)	54	56
(1.5,1.5,1.5)	60	64
(1.5,2.0,2.0)	63	67
(2.0,2.0,2.0)	65	68
(2.0,2.5,2.5)	67	70
(2.5,2.5,2.5)	68	72
(2.5,3.0,3.0)	69	74
(3.0,3.0,3.0)	70	76

Πίνακα 5.6 Αποτελεσματικότητα Bayes με στρογγυλοποίηση στο 1ο δεκαδικό για πιθανά κατώφλια.

5.3.4 Σύνοψη αποτελεσμάτων για Bayes Naïve

Αποτελεσματικότητα

Από τα παραπάνω γίνεται αντιληπτό πως η ευστοχία του αλγορίθμου είναι ανάλογη με την στρογγυλοποίηση των δεκαδικών χαρακτηριστικών και με τα κατώφλια. Όταν ένας από τους δύο αυτούς παράγοντες αυξάνει(στρογγυλοποίηση σε μεγαλύτερο δεκαδικό, ή πιο αυστηρό κατώφλι) τότε αυξάνει και η αποτελεσματικότητα. Με τον στρογγυλοποίηση μπορούμε να βελτιώσουμε τον αλγόριθμο μέχρι και +3% σε κάποιες των περιπτώσεων, ενώ με τα κατώφλια οι βελτίωση της ευστοχίας του αλγορίθμου μπορεί να φτάσει και σε τάξεις του +21%.

Βλέπουμε ότι τα κατώφλια κατέχουν πολύ σημαντικό ρόλο στον αλγόριθμο του Bayes, αλλά πρέπει ο χρήστης να τα διαχειριστεί με προσοχή, έτσι ώστε να μην κάνει τον αλγόριθμο ιδιαίτερα απαιτητικό για την πρόγνωση αποτελέσματος. Το κατώφλι που επιλέχθηκε για την εφαρμογή αυτή είναι το (2.5,2.5,2.5) που πετυχαίνει ένα πολύ καλό ποσοστό επιτυχίας, της τάξης του 68% κατά μέσο όρο και 72% σε ορισμένες των περιπτώσεων, αλλά και δεν είναι τόσο αυστηρό ώστε να έχουμε ικανοποιητικό αριθμό προβλέψεων. Όπως αναφέραμε και προηγουμένως κάνει πρόβλεψη στο 51% του συνόλου των αγώνων που ζητάνε πρόβλεψη.

Τέλος προτιμήθηκε η στρογγυλοποίηση των δεκαδικών χαρακτηριστικών στο 1ο δεκαδικό ψηφίο, αφού προφέρουν, έστω και μικρή, βελτίωση του αλγορίθμου, χωρίς κάποιο υπολογιστικό κόστος.

5.4 Σύγκριση CART - Bayes

Από όλα τα παραπάνω είναι εμφανής η προτίμηση του αλγορίθμου Bayes έναντι του CART, τόσο λόγου του χρόνου εκτέλεσης και του υπολογιστικού κόστους, όσο και λόγο της επιτυχίας του κάθε αλγορίθμου. Τέλος ο Bayes λόγο της υλοποίησης της λειτουργικότητάς του, με την βοήθεια των πιθανοτήτων τον καθιστά και ένα σημαντικό εργαλείο για άλλες υπηρεσίες οι οποίες ίσως υλοποιηθούν μελλοντικά. Η σύγκριση των δύο αλγορίθμων παρουσιάζεται στον πίνακα 5.7. Η τιμές που παρουσιάζονται, βασίζονται στις ακόλουθες επιλογές:

1. CART:
    - Split Method : Gini Impurity
    - Κατώφλι κέρδους πληροφορίας : 0.3
    - Στρογγυλοποίηση στο 1ο δεκαδικό ψηφίο
2. Bayes naïve
    - Στρογγυλοποίηση στο 1ο δεκαδικό ψηφίο
    - Κατώφλι πιθανότητας του προτεινόμενου αποτελέσματος : (2.5,2.5,2.5)

	CART	Bayes Naive
Χρόνος Εκπαίδευσης	6.8 secs	0.06 secs
Χρόνος Πρόγνωσης	$5.8*10^{-5}$ sec	0
Συνολικός χρόνος	6.8 secs	0.06 secs
Ποσοστό Επιτυχίας(Μ.Ο.)	50	68
Ποσοστό Επιτυχίας(Max)	52	72

Πίνακα 5.7 Σύνοψη αποτελεσμάτων

Θα πρέπει να αναφερθεί ότι οι παραπάνω υπολογισμοί χρόνου πραγματοποιήθηκαν σε ηλεκτρονικό υπολογιστή με τα παρακάτω χαρακτηριστικά:

- **Επεξεργαστή:** AMD Athlon 64x2 3800+ 2.00GZ
- **Μνήμη RAM:** Kingston 2GB στα 533MHz
- **Σκληρός δίσκος:** 7200RPM SATA2

6. Μελλοντική εργασία

Ο αρχικός στόχος, ενός ποσοστού επιτυχίας της τάξης του 60% μπορεί να μην επιτεύχθηκε στον αλγόριθμο Cart, αλλά επιτεύχθηκε με μεγάλη επιτυχία στον αλγόριθμο Bayes naïve. Όπως φαίνεται στο κεφάλαιο 5 υπάρχουν περιπτώσεις όπου ο αλγόριθμος του Bayes επιστρέφει αποτελέσματα με πιθανότητα ευστοχία της τάξης του 72%. Τα αποτελέσματα αυτά μπορούνε να βελτιωθούν και να επιτευχθεί ακόμα καλύτερη ευστοχία με μία πιο λεπτομερή αναπαράσταση του προβλήματος.

Κάποια παραδείγματα που επιβεβαιώνουν τον παραπάνω ισχυρισμό είναι οι ακόλουθες εφαρμογές :

1. Score Card : Αναπτύχθηκε από τους J.Coleman(καθηγητής Operations Management & Quantitative Methods στο Coggin College Of Business της North Florida) και A.Lynch(Associate Professor of Economics & Quantitative Methods στο Stetson School of Business and Economics του Mercer). Η εφαρμογή χρησιμοποιούσε 50 ήδη πληροφοριών και μπορούσε να προβλέψει με επιτυχία 86%, τις εκπλήξεις, δηλαδή αγώνες που έχουν τελικό αποτέλεσμα αυτό με την μικρότερη πιθανότητα.
2. Dance Card: Αναπτύχθηκε και πάλι από τους J.Coleman και A.Lynch. Κατάφερε με την χρησιμοποίηση 42 τύπων χαρακτηριστικών να πετύχει μέχρι και 97% ευστοχία στις προβλέψεις του

Οι παραπάνω παρατηρήσεις μας επιβάλουν την συνέχεια της ενασχόλησης με το συγκεκριμένο πρόβλημα, που πιθανώς να οδηγήσει σε μία νέα έκδοση της εφαρμογής με καλύτερες επιδόσεις και λειτουργίες.

Πιο συγκεκριμένα η νέα έκδοση αναμένεται να παρέχει τα εξής επιπλέον χαρακτηριστικά:

1. Εμπλουτισμός της αναπαράστασης του προβλήματος. Η νέα έκδοση θα παρέχει μία πιο πλούσια και λεπτομερή αναπαράσταση του προβλήματος. Πιο συγκεκριμένα θα παρέχει χαρακτηριστικά για τους παίχτες των ομάδων(όπως το πόσα τέρματα πέτυχαν σε κάθε αγώνα, πόσα φάουλ κέρδισαν, πόσα σουτ επιχείρησαν κ.α.), χαρακτηριστικά για την κρισιμότητα των αγώνων, χαρακτηριστικά για το ενδιαφέρον κάθε ομάδας για τον αγώνα κ.α.
2. Ακόμα μεγαλύτερο σύνολο εκπαίδευσης. Αν και τα 5 χρόνια θεωρούνται ικανοποιητικά σαν μέγεθος εκπαίδευσης, ωστόσο η αύξηση του συνόλου εκπαίδευσης μόνο θετικά μπορεί α επηρεάσει την αποτελεσματικότητα των αλγορίθμων(ιδιαίτερα στην περίπτωση του CART)
3. Υποστήριξη ακόμα περισσότερων αλγορίθμων. Σίγουρα θα προστεθούν και άλλοι αλγόριθμοι από την μάθηση κατά Bayes, αλλά θα δοκιμαστούν και

αλγόριθμοι από άλλες τεχνικές μηχανικής μάθησης με επίβλεψη, όπως Νευρωνικά Δίκτυα, Μάθηση Κανόνων κ.α.

4. Προσθήκη νέων εναλλακτικών εξαρτημένων μεταβλητών. Για παράδειγμα σε περιπτώσεις που δε μπορεί να προβλεφθεί ένα συγκεκριμένο αποτέλεσμα, να μπορεί να προτείνει δύο. Ακόμα η υποστήριξη πρόβλεψης για Over/Under, πρόβλεψη για αποτέλεσμα ημιχρόνου ή ακόμα και πρόβλεψη για το αν θα σκοράρουν και οι δύο ομάδες στον αγώνα.

Πέρα όμως από μία πιθανή νέα έκδοση της εφαρμογής, θα γίνουν περαιτέρω πειραματισμοί με την συγκεκριμένη μοντελοποίηση. Για παράδειγμα, είναι στα σχέδια ο έλεγχος της συγκεκριμένης μοντελοποίησης πάνω σε άλλα πρωταθλήματα. Η επιλογή του Αγγλικού πρωταθλήματος όπως προαναφέρθηκε, ίσως τελικά να μην ήταν και η πιο εγκεκριμένη, καθώς ο μεγάλος αριθμός των εκπλήξεων δυσκολεύει την διαδικασία πρόγνωσης. Ένα πρωτάθλημα στο οποίο τα αποτελέσματα είναι πιο αναμενόμενα, όπως για παράδειγμα το ελληνικό, ίσως να ήταν πιο ενδεδειγμένο.

Σε κάθε περίπτωση η ενασχόληση με ένα τόσο δύσκολο και απρόβλεπτο πρόβλημα αποτελεί μία πραγματική πρόκληση.

Βιβλιογραφία

Connoly, T., Begg, C. & Stracan, A. (1999). *Data Base Systems: A practical approach to Design, Implementation and Management. Second Edition.* England: Press of University Paisley.

Elmasri, R. & Narathe, S.B. (2007). *Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων (3η Έκδοση)* (Τομ.Α & Τομ.Β) (Χατζόπουλος, Μ., Μετάφ.). Αθήνα: Εκδόσεις Δίαυλος.

Ramakrishman, R. & Gehrke, J. (2002). *Συστήματα Διαχείρισης Βάσεων Δεδομένων (2η Έκδοση) (Τομ.Α & Τομ.Β) (Μετάφρ.). Θεσσαλονίκη: Εκδόσεις Τζιόλα.*

Segaran, T.(2007). *Programming Collective Intelligence.* United States of America: O'Reilly Press.

Tan, P.N., Steinback, M. & Kumar, V. (2006). *Introduction to Data Mining.* Minnesota: Person International Editor.

Βερύκιος, Β. & Θεοδορίδης, Γ. (2004). *Data Mining:Εισαγωγικά και Προηγμένα Θέματα Εξόρυξης Γνώσης από Δεδομένα.* Αθήνα: Εκδόσεις Νέων Τεχνολογιών. Μετάφραση από: Dunham, Margaret H. (2003). *Data Mining Introductory and Advanced Topics.* Prentice Hall: Pearson Education Inc.

Βλαχάβας, Ι., Κεφάλας, Π., Βασιλειάδης, Ν., Ρεφανίδης, Ι., Κόκκορας, Φ. & Σακελλαρίου, Η. (2002). *Τεχνητή Νοημοσύνη.* Θεσσαλονίκη: Εκδόσεις Γαρταγάνης.

Πηγές Διαδικτύου:

http://www.swaroopch.com/notes/Python_el:Περιεχόμενα (2010, Ιούνιος).

http://docs.djangoproject.com/en/1.2/ (2010, Μάιος).

http://www.djangobook.com/ (2010, Μάιος).

http://docs.python.org/tutorial/index.html (2010, Ιούνιος).

http://dev.mysql.com/tech-resources/articles/mysql_intro.html (2010, Φεβρουάριος).

http://www.w3schools.com/ (2010, Μάιος).