Α Ρ Ι Σ Τ Ο Τ Ε Λ Ε Ι Ο Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο
Θ Ε Σ Σ Α Λ Ο Ν Ι Κ Η Σ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Π Τ Υ Χ Ι Α Κ Η / Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η
Ε Ρ Γ Α Σ Ι Α

ΣΤΑΤΙΣΤΙΚΗ ΑΝΑΛΥΣΗ ΚΑΙ ΠΡΟΒΛΕΨΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΑΓΩΝΩΝ ΠΟΔΟΣΦΑΙΡΟΥ

ΠΑΥΛΟΣ ΠΟΛΙΑΝΙΔΗΣ
Α.Ε.Μ: 1586

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
Γρηγόριος Τσουμάκας, Λέκτορας

ΘΕΣΣΑΛΟΝΙΚΗ
ΙΟΥΛΙΟΣ 2010

ΠΕΡΙΛΗΨΗ

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η στατιστική ανάλυση καθώς και η πρόβλεψη αγώνων ποδοσφαίρου. Το πρώτο κομμάτι της εργασίας αφορά την συγκέντρωση πληροφοριών σχετικά με αγώνες ποδοσφαίρου από όλο τον κόσμο (όπως αποτελέσματα, ημερομηνία, σκορ ημιχρόνου, πρωτάθλημα κ.α.) καθώς επίσης και τις αποδόσεις στοιχήματος στους αγώνες αυτούς. Τα δεδομένα συλλέγονται με αυτόματο τρόπο από αρχεία XML που είναι διαθέσιμα στον ιστότοπο του ΟΠΑΠ και αποθηκεύονται σε μια βάση δεδομένων.

Στο δεύτερο κομμάτι της εργασίας γίνεται μια προσπάθεια πρόβλεψης των αποτελεσμάτων των αγώνων, χρησιμοποιώντας τις πληροφορίες που συγκεντρώσαμε και είναι αποθηκευμένες στην βάση δεδομένων. Πρόκειται ουσιαστικά, για εφαρμογή αλγορίθμων *μηχανικής μάθησης* και συγκεκριμένα *αλγορίθμων ταξινόμησης (classification algorithms)* ώστε να γίνει η κατάταξη των επικείμενων αγώνων ποδοσφαίρου σε μια από τις τρείς κατηγορίες (νίκη γηπεδούχου ομάδας, ισοπαλία, νίκη φιλοξενούμενης ομάδας) , σύμφωνα με την δυναμική των ομάδων.

Το πρόβλημα αυτό ανήκει στην κατηγορία των προβλημάτων ταξινόμησης, στα οποία δεν μας ενδιαφέρει απλώς να κατατάξουμε ένα *στιγμιότυπο (instance)* σε κάποια από τις εναλλακτικές *κλάσεις (classification labels)*, αλλά μας ενδιαφέρει και το *κόστος* (ή ανάλογα το κέρδος) που θα έχουμε από αυτήν την κατηγοριοποίηση. Οι αλγόριθμοι αυτή, λαμβάνουν υπ' όψιν το κόστος της σωστής πρόβλεψης καθώς και το κόστος της λανθασμένης πρόβλεψης (misclassification cost) ώστε να πετύχουν όσο το δυνατόν πιο ιδανική πρόβλεψη. Το είδος αυτό της μάθησης, ονομάζεται *cost-sensitive machine learning[5,3]*. Το σύνολο των παραπάνω αλγορίθμων, ανήκει στην κατηγορία των λεγόμενων *αλγορίθμων μάθησης που είναι ευαίσθητοι ως προς το κόστος (cost-sensitive learning algorithms)*. Αποτελεί έναν νέο τομέα της μηχανικής μάθησης γύρω από τον οποίο γίνεται πολύ έρευνα, γιατί καθίσταται όλο και πιο φανερό ότι οι περισσότερες εφαρμογές μηχανικής μάθησης, πλέον απαιτούν τη χρήση cost-sensitive αλγορίθμων, οι οποίοι θα εξετάζουν τον κίνδυνο που εγκυμονεί η πρόβλεψη που θα κάνουν.

Στις σελίδες που ακολουθούν θα δείξουμε πώς θα εφαρμόσουμε *cost-sensitive algorithms with different cost per instance* (όπου το κόστος κατηγοριοποίησης είναι διαφορετικό για κάθε στιγμιότυπο) πάνω στο δικό μας πρόβλημα.

ΕΥΧΑΡΙΣΤΙΕΣ

Καταρχάς θα ήθελα να εκφράσω τις αληθινές μου ευχαριστίες στον επιβλέπων της πτυχιακής εργασίας κ. Τσουμάκα Γρηγόριο, Λέκτορα του τμήματος πληροφορικής του Αριστοτέλειου πανεπιστημίου, για την εμπιστοσύνη μου έδειξε κατά την ανάθεση της παρούσας πτυχιακής, αλλά και την αμέριστη συμπαράσταση καθ' όλη τη διάρκεια της εκπόνησης της.

Τέλος θα ήθελα να ευχαριστήσω του γονείς μου, για όλη τη βοήθεια τόσο ψυχολογική όσο και οικονομική, που μου προσέφεραν όλα αυτά τα χρόνια.

15-07-2010

Πολιανίδης Παύλος

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

ΕΙΣΑΓΩΓΗ

Η *εξόρυξη δεδομένων* είναι ένας κλάδος της επιστήμης της πληροφορικής, ο οποίος έχει αναπτυχθεί σημαντικά τα τελευταία χρόνια. Βασικός στόχος της εξόρυξης δεδομένων είναι η ανακάλυψη κρυφής γνώσης, χρησιμοποιώντας αλγορίθμους της *μηχανικής μάθησης*, μέσα από ένα μεγάλο σύνολο δεδομένων που είναι αποθηκευμένα σε βάσης ή αποθήκες δεδομένων.[1]

Στη παρούσα πτυχιακή εργασία θα χρησιμοποιήσουμε την εξόρυξη δεδομένων, για να ανακαλύψουμε τη *γνώση* που κρύβεται μέσα στα δεδομένα που έχουμε συλλέξει και αφορούν αγώνες ποδοσφαίρου. Πιο συγκεκριμένα θα εφαρμόσουμε αλγορίθμους ταξινόμησης (classification algorithms) οι οποίοι θα προβλέψουν τα αποτελέσματα των αγώνων, που βρίσκονται στο κουπόνι στοιχήματος του ΟΠΑΠ.

Θα εφαρμόσουμε διάφορους αλγορίθμους ταξινόμησης πάνω στα δεδομένα μας, ώστε να γίνει φανερό μετά από πειραματικές εκτελέσεις, ποίος επιτυγχάνει το μεγαλύτερο ποσοστό επιτυχημένης πρόβλεψης. Ονομαστικά, οι αλγόριθμοι που θα χρησιμοποιηθούν είναι :

- *BFTree*
- *J48*
- *J48Graft*
- *LADTree*
- *REPTree*
- *SimpleCart*
- *NaiveBayes*
- *NaiveBayesNominal*
- *NaiveBayesSimple*
- *Multinomial logistic regression*
- *SMO*
- *Linear logistic regression*
- *IB1*
- *DecisionTable*

Οι κλασικοί αυτοί αλγόριθμοι ταξινόμησης[1,7] έχουν ως βασικό στόχο να κατατάξουν επιτυχώς το κάθε στιγμιότυπο σε μια κλάση από ένα διακριτό σύνολο κλάσεων. Στο πρόβλημα μας, τα στιγμιότυπα είναι οι *αγώνες ποδοσφαίρου* και οι κλάσεις ανήκουν στο εξής διακριτό σύνολο: {*νίκη γηπεδούχο ομάδας, ισοπαλία, νίκη φιλοξενούμενης ομάδας*}. Οι αλγόριθμοι αυτοί προσπαθούν να μειώσουν το *ρυθμό σφάλματος (error rate)* : το σύνολο των λανθασμένων προβλέψεων ή εναλλακτικά τη πιθανότητα της εξαγωγής λάθους πρόβλεψης, και ως εκ τούτου την αύξηση των επιτυχημένων προβλέψεων (που είναι ο βασικός στόχος). Το πρόβλημα έγκυται στο ότι ο αλγόριθμος θεωρεί ότι όλες οι λάθος πρόβλεψης επιφέρουν το ίδιο κόστος, στην πραγματικότητα όμως αυτό δεν ισχύει πάντα. Στο πρόβλημα μας, θα θέλαμε ο αλγόριθμος να προσπαθεί να προβλέψει σωστά την κλάση η οποία θα επιφέρει το μικρότερο κόστος (ή ανάλογα το μεγαλύτερο κέρδος) λαμβάνοντας υπ' όψιν τις απόδοσης των σημείων.

Για να συμπεριλάβουμε το κόστος της λανθασμένης πρόβλεψης, θα εφαρμόσουμε του επονομαζόμενους *cost-sensitive αλγορίθμους[3,6,2,4]*. Οι αλγόριθμοι αυτοί θα εφαρμοστούν ως «περιτύλιγμα» (wrapper) στους παραπάνω αλγορίθμους ταξινόμησης ώστε να εισαγάγουν την έννοια του κόστους στη διαδικασία της ταξινόμησης. Θα χρησιμοποιήσουμε τους εξής τρεις:

- *MetaCost*
- *CostSensitiveClassifier*
- *CSRoullete*

ΚΕΦΑΛΑΙΟ 2: ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Όπως αναφέρθηκε προηγουμένως, σκοπός του συστήματος είναι η πρόβλεψη αγώνων ποδοσφαίρου. Για να πετύχουμε το σκοπό μας, θα χρησιμοποιήσουμε ένα σύνολο αλγορίθμων της μηχανικής μάθησης. Πιο συγκεκριμένα, η διαδικασία της πρόβλεψης αναφέρεται ως πρόβλημα *ταξινόμησης (classification problem)* στην μηχανική μάθηση, ως εκ τούτου θα γίνει χρήση *αλγορίθμων ταξινόμησης ή κατηγοριοποίησης*.

ΤΑΞΙΝΟΜΗΣΗ

2.1.1 Βασικές έννοιες

Σε αυτό το σημείο θα ήταν σκόπιμο να αναφερθούμε στη λειτουργία των αλγορίθμων ταξινόμησης. Αρχικά, θα εστιάσουμε την προσοχή μας, σε ορισμένες βασικές έννοιες που λαμβάνουν χώρα στη διαδικασία της ταξινόμησης και γενικότερα, της μηχανικής μάθησης.

- **Μοντέλο ταξινόμησης ή ταξινομητής (classifier):** μία δομή η οποία ερμηνεύει και περιγράφει το σύνολο των δεδομένων. Βάση του μοντέλου που σχηματίζεται, γίνεται η πρόβλεψη.
- **Στιγμιότυπο ή αντικείμενο (instance):** ένα αντικείμενο του κόσμου. Πολλά αντικείμενα μαζί σχηματίζουν το σύνολο στο οποίο θα στηριχθεί ο αλγόριθμος ταξινόμησης, για να σχηματίσει το μοντέλο της ταξινόμησης. Το στιγμιότυπο μπορεί επίσης να εφαρμοστεί πάνω στο μοντέλο, προκειμένου να γίνει η πρόβλεψη.
- **Χαρακτηριστικά (attribute):** μία ιδιότητα που περιγράφει ένα στιγμιότυπο. Το κάθε χαρακτηριστικό, έχει έναν τύπο.
- **Κλάση:** ανήκει σε ένα πεπερασμένο σύνολο από χαρακτηριστικά (τύπου *nominal*), και δηλώνει την τάξη στην οποία ανήκει ένα στιγμιότυπο. Είναι ουσιαστικά, το χαρακτηριστικό το οποίο καλείται να πρόβλεψη ο αλγόριθμός.
- **Δεδομένα μάθησης ή σύνολο μάθησης (training set):** ένα σύνολο από στιγμιότυπα , από το οποίο ο αλγόριθμος ταξινόμησης θα σχηματίσει τον ταξινομητή.
- **Δεδομένα ελέγχου ή σύνολο ελέγχου (test set):** ένα σύνολο από στιγμιότυπα, τα οποία ο αλγόριθμος θα προβλέψει, δηλαδή θα τα κατατάξει σε μια από τις δυνατές κλάσεις.
- **Ακρίβεια πρόβλεψης (accuracy):** ο λόγος των σωστών προβλέψεων προς το σύνολο όλων των προβλέψεων που έγιναν από τον ταξινομητή.

$$accuracy = \frac{\#correctly\_predicted\_examples}{\#predictions\_made}$$

- **Υπερπροσαργογή (overfitting):** αφορά το φαινόμενο κατά το οποίο, το μοντέλο που κατασκευάζεται από τον αλγόριθμο, προσαρμόζεται σε υπερβολικό βαθμό στα δεδομένα του συνόλου εκπαίδευσης. Αυτό έχει ως αποτέλεσμα, να μειώνεται η ακρίβεια της πρόβλεψης δεδομένων, που δεν ανήκουν στο σύνολο εκπαίδευσης.
- **Μάθηση με επίβλεψη:** μάθηση κατά την οποία ,οι κλάσεις των στιγμιότυπων του συνόλου εκπαίδευσης, είναι γνωστές. Οι αλγόριθμοι ταξινόμησης, ανήκουν σε αυτήν την κατηγορία. Στην παρούσα εργασία, θα ασχοληθούμε κυρίως με αυτό το είδος μάθησης.
- **Μάθηση χωρίς επίβλεψη:** μάθηση κατά την οποία, οι κλάσεις των στιγμιότυπων του συνόλου εκπαίδευσης δεν είναι γνωστές. Στην μάθηση αυτή, οι αλγόριθμοι προσπαθούν να βρουν συσχετίσεις και ομάδες δεδομένων από το σύνολο εκπαίδευσης. Παραδείγματα ομάδων αλγορίθμων που εφαρμόζουν αυτό το είδος μάθησης είναι:
  I. Αλγόριθμοι που εξάγουν κανόνες *συσχετίσεις (association rules)* π.χ. apriori.
  II. Αλγόριθμοι που χωρίζουν τα δεδομένα σε *συστάδες (clusters)* π.χ. k-means, DBScan.

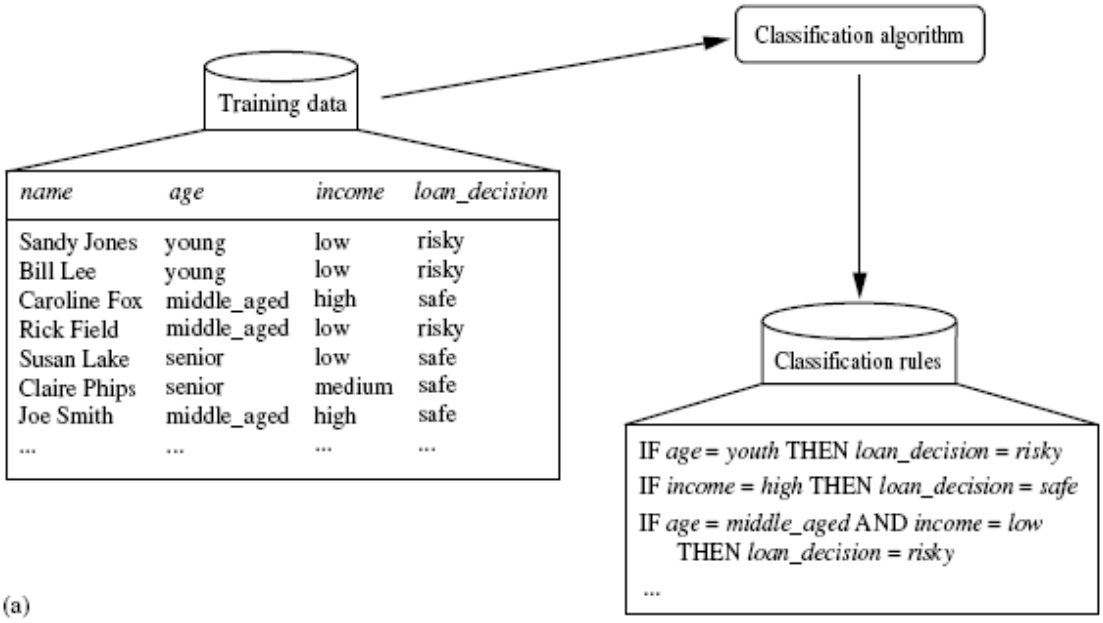
Έχοντας ξεκαθαρίσει ορισμένες βασικές έννοιες, μπορούμε πλέον να ορίσουμε την *ταξινόμηση (classification)* ως τη διαδικασία δημιουργίας ενός *ταξινομητή ή μοντέλου ταξινόμησης* από το σύνολο εκπαίδευσης, και η εφαρμογή του μοντέλου αυτού για την αντιστοίχηση των στιγμιότυπων του *συνόλου ελέγχου*, σε μια από δυνατές τάξης του διακριτού *συνόλου κλάσεων[1,9]*.

2.1.2 Βήματα εκτέλεσης

Παρακάτω, περιγράφονται συνοπτικά τα δυο βασικά βήματα της εφαρμογής του αλγορίθμου ταξινόμησης που αναφέρονται στον παραπάνω ορισμό.

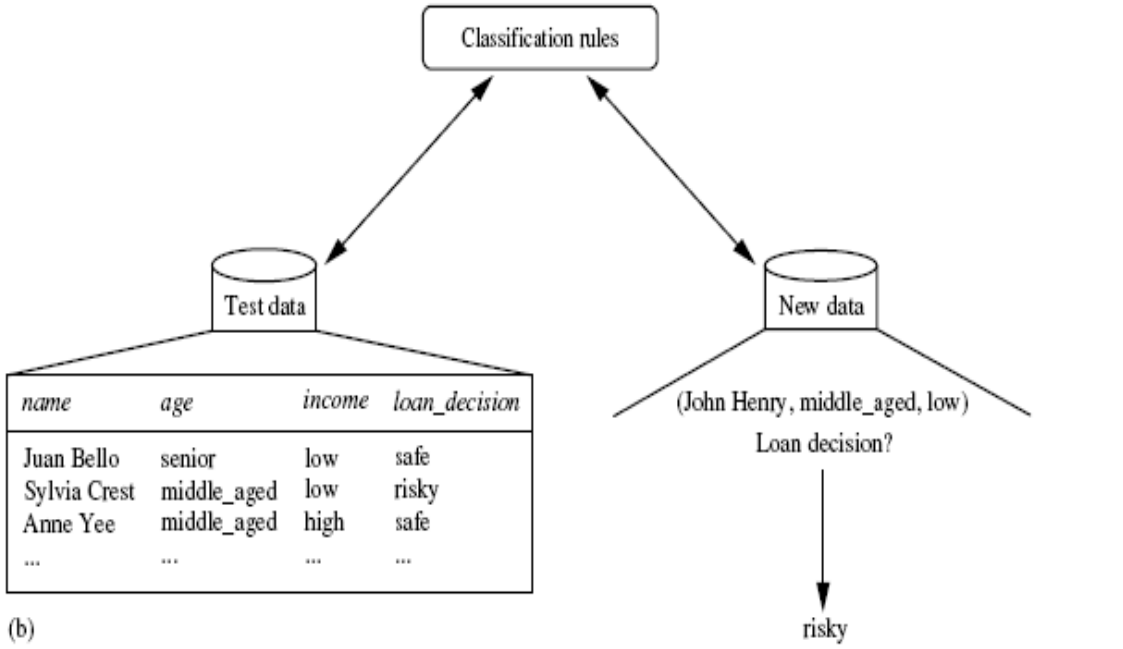
1. **Στάδιο της μάθησης:** αποτελεί το βήμα της διαδικασίας της ταξινόμησης, κατά το οποίο ο αλγόριθμος ταξινόμησης, κατασκευάζει τον *ταξινομητή (classifier),* αναλύοντας το σύνολο εκπαίδευσης (βλέπε εικόνα 2.1). Το σύνολο εκπαίδευσης όπως αναφέραμε, αποτελείται από στιγμιότυπα, που συνήθως αναπαρίστανται ως ένα *διάνυσμα χαρακτηριστικών (feature vector).* Το διάνυσμα αυτό είναι της μορφής $X = <x_1,x_2,...x_n>$, όπου x_i τα χαρακτηριστικά του στιγμιότυπου και ανήκει σε μία κλάση. ο συνδυασμός του διανύσματος και της κλάσης στην οποία ανήκει, αποτελεί την βασική ιδέα πίσω από την οποία κρύβεται η διαδικασία της κατασκευής του ταξινομητή.

2. **Στάδιο της ταξινόμησης:** στο δεύτερο βήμα, ο αλγόριθμος αρχικά κάνει μια εκτίμηση της *ακρίβειας* του μοντέλου που κατασκεύασε. Το μοντέλο εφαρμόζεται πάνω στο σύνολο εκπαίδευσης, όπου πλέον οι κλάσεις των στιγμιότυπων του συνόλου δεν είναι γνωστές και ο ταξινομητής καλείται να τις προβλέψει (βλέπε εικόνα 2.2). Υπάρχει πάντα η περίπτωση της αισιόδοξης εκτίμησης, η οποία δεν συμβαδίζει συνήθως με την πραγματικότητα. Αυτή η αισιοδοξία που παρουσιάζουν οι ταξινομητές οφείλεται στο φαινόμενο της υπερβολικής προσαρμογής (βλέπε φαινόμενο υπερπροσαρμογής). Έχουν προταθεί αρκετοί μέθοδοι για την *αποτίμηση της ακρίβειας* των ταξινομητών [7,9]. Θα αναφερθούμε σε ορισμένους από αυτούς, στην συνέχεια του κεφαλαίου.



Εικόνα 2.1

Μάθηση: Ο αλγόριθμος ταξινόμησης αναλύει το σύνολο εκπαίδευση (training data). Σε αυτήν την περίπτωση, η κλάση του κάθε παραδείγματος, είναι η *loan_decision* , και το μοντέλο που κατασκευάζεται, αναπαριστάται από ένα σύνολο *κανόνων ταξινόμησης (decision rules)*.



Εικόνα 2.2

Ταξινόμηση: Πραγματοποιείται η αποτίμηση της ακρίβειας του μοντέλου που κατασκευάστηκε στο πρώτο βήμα, με την εφαρμογή του, πάνω σε ένα σύνολο ελέγχου (test data). Στην συνέχεια, γίνεται η κατηγοριοποίηση των παραδειγμάτων του συνόλου new data

2.1.3 Αποτίμηση ακρίβειας

Η αποτίμηση της ακρίβειας ενός ταξινομητή, είναι μια πολύ σημαντική εργασία, που λαμβάνει χώρα κατά την εκτέλεση της ταξινόμησης. Αφορά σε ένα είδος δοκιμής που εκτελεί ο αλγόριθμος, προκειμένου να συμπεράνει πόσο καλός είναι ο ταξινομητής που κατασκεύασε. Η ακρίβεια είναι μια σημαντική ποσότητα για εμάς, γιατί μας φανερώνει την ικανότητα του μοντέλου να κάνει σωστές προβλέψεις (κατάταξη των στιγμιότυπων στην κλάση στην οποία πραγματικά ανήκουν). Είναι προφανές, ότι όσα μεγαλύτερη είναι η ακρίβεια, τόσο περισσότερο μπορούμε να εμπιστευτούμε τον αλγόριθμο, για τις προβλέψεις που κάνει. Συνήθως υπάρχει ένα κάτω όριο ακρίβειας, κάτω από τον οποίο ο αλγόριθμος της ταξινόμησης έχει την δυνατότητα να απορρίψει ένα μοντέλο και να ξεκινήσει την διαδικασία από το πρώτο βήμα (βλέπε βήματα εκτέλεσης αλγορίθμου), κατασκευάζοντας ένα διαφορετικό μοντέλο[7].

2.1.3.1 Μέθοδοι αποτίμησης ακρίβειας

2.1.3.1.1 hold-out

Πρόκειται για την απλούστερη μέθοδο αποτίμησης της ακρίβειας ενός κατηγοριοποιητή. Η ιδέα που κρύβεται πίσω από την μέθοδο αυτή είναι απλή, χωρίζουμε το σύνολο των δεδομένων (data set) σε 2 τμήματα, το *σύνολο εκπαίδευσης* και το *σύνολο ελέγχου*, με κάποια αναλογία. Για παράδειγμα, τα 2/3 πρώτα στιγμιότυπα του συνόλου δεδομένων θα σχηματίσουν το σύνολο εκπαίδευσης, ενώ από το υπόλοιπο 1/3 του συνόλου δεδομένων θα σχηματιστεί το σύνολο ελέγχου. Το σύνολο εκπαίδευσης θα συμβάλει στην κατασκευή του κατηγοριοποιητή ο οποίος θα κατατάξει τα όλα τα στιγμιότυπα του συνόλου ελέγχου, σε μία από τις υπάρχουσες κλάσεις. Αν υποθέσουμε ότι το X είναι ο αριθμός των στιγμιότυπων του συνόλου ελέγχου που κατατάσσονται στη σωστή κλάση, και N ο συνολικός αριθμός στιγμιότυπων του συνόλου ελέγχου, τότε η ακρίβεια *accuracy* της μεθόδου hold-out ορίζεται ως εξής:

$$accuracy = \frac{X}{N}$$

Η μέθοδος hold-out είναι μια απλή μέθοδος η οποία είναι εύκολη στην υλοποίηση, παρόλα αυτά, έχει ένα σημαντικό πρόβλημα. Η ποιότητα του κατηγοριοποιητή που θα κατασκευαστεί, εξαρτάται από την κατανομή των στιγμιότυπων μέσα στο σύνολο δεδομένων. Υπάρχει η περίπτωση το σύνολο εκπαίδευσης να περιέχει λίγα στιγμιότυπα που ανήκουν σε μια κλάση, αυτό θα έχει ως αποτέλεσμα την δημιουργία ενός μοντέλου το οποίο δεν θα μπορεί να κατατάσσει εσφαλμένα τα στιγμιότυπα του συνόλου ελέγχου που πραγματικά ανήκουν στην κλάση αυτήν. Σε αυτήν την περίπτωση η ακρίβεια του μοντέλου θα είναι υποεκτιμημένη, λόγω του ότι δεν κατατάσσει ορθά τα στιγμιότυπα της συγκεκριμένης κλάσης.

2.1.3.1.2 Τυχαία δειγματοληψία (random subsampling)

Η μέθοδος ακολουθεί παρόμοια λογική με την hold-out. Οι δυο μέθοδοι διαφέρουν στον τρόπο που διαμερίζουν το σύνολο δεδομένων στα δύο τμήματα (σύνολο εκπαίδευση και σύνολο ελέγχου), ώστε να λυθεί το πρόβλημα που εμφανίζει η hold-out. Σε αυτήν την περίπτωση το σύνολο ελέγχου επιλέγεται ως τυχαίο δείγμα. Εφαρμόζουμε τυχαία δειγματοληψία χωρίς επανατοποθέτηση και επιλέγουμε N στιγμιότυπα για το σύνολο ελέγχου. Τα εναπομένοντα στιγμιότυπα σχηματίζουν το σύνολο εκπαίδευσης. Με αυτήν την ενέργεια μειώνεται η επιρροή που μπορεί να επιφέρει η κατανομή των στιγμιότυπων στο σύνολο δεδομένων. Η προηγούμενη διαδικασία επαναλαμβάνεται κ φορές ώστε νε επιτευχθεί η μεγαλύτερη δυνατή μείωση της επιρροής. Αν υποθέσουμε ότι X_i είναι ο αριθμός των στιγμιότυπων που έχουν καταταχθεί σωστά στην i-οστη επανάληψη (οπού $1 \leq i \leq k$) , τότε η ακρίβεια της μεθόδου τυχαίας δειγματοληψίας ορίζεται ως εξής:

$$accuracy = \frac{1}{\kappa}\sum_{i=1}^{k}\frac{X_i}{N}$$

2.1.3.1.3 Πολλαπλή δια-εγκυροποίηση (k-fold cross-validation)

Θεωρείται μια από τις πλέον αξιόπιστές μεθόδους για την αποτίμηση της ακρίβειας κατηγοριοποιητών. Στην περίπτωση αυτήν εφαρμόζεται ένας πιο συστηματικός τρόπος για να λαμβάνουμε τυχαία δείγματα για το σχηματισμό του συνόλου ελέγχου. Εάν το σύνολο δεδομένων αποτελείται από M στιγμιότυπα και έχουμε ορίσει ως αριθμό επαναλήψεων το k, τότε χωρίζουμε το σύνολο σε k ισομεγέθη τμήματα μεγέθους *M/k* το καθένα. Στην *i-οστή* επανάληψη, το *i-οστό* τμήμα λειτουργεί ως σύνολο ελέγχου, ενώ τα υπόλοιπα *k – 1* τμήματα αποτελούν το σύνολο εκπαίδευσης. Η συνηθέστερη τιμή για το *k* είναι το 10 *(10-fold cross-validation).*

Η μέθοδος της δια-εγυροποίησης για την ειδική περίπτωση οπού *k =M* , ονομάζεται *άφησε-ένα-έξω (leave-one-out)*, γιατί το σύνολο ελέγχου σε κάθε επανάληψη, αποτελείται από ένα μόνο στιγμιότυπο. Η παραλλαγή αυτή χρησιμοποιείται μόνο σε μικρά σύνολα δεδομένων.

2.1.3.1.4 Bootstrap

Αποτελεί εναλλακτική μέθοδο της τυχαίας υποδειγματοληψίας. Ενώ η τυχαία υποδειγματοληψία σχηματίζει το σύνολο ελέγχου εφαρμόζοντας τυχαία δειγματο-ληψία χωρίς επανατοποθέτηση στο σύνολο δεδομένων, ο *bootstrap* εφαρμόζει τυχαία δειγματοληψία με επανατοποθέτηση. Έστω *M* το πλήθος των στιγμιότυπων του συνόλου δεδομένων. Αποδεικνύεται ότι αν από το σύνολο αυτό πάρουμε ένα δείγμα *M* στιγμιότυπων, εξαιτίας της επανατοποθέτησης ο αριθμός των διακριτών στιγμιότυπων του δείγματος αυτού, αναμένεται να ισούται με το 63.2% του *M.*

Απόδειξη:

Για ένα αντικείμενο χ, η πιθανότητα είναι: P(x $\in$ Δείγμα) = 1 – P(x $\notin$ Δείγμα). Σε κάθε μια από τις *M* προσπάθειες, το *x* έχει πιθανότητα 1/*M* να επιλεγεί, και 1 – 1/*M* πιθανότητα να μην επιλεγεί. Επομένως, μετά από *M* προσπάθειες, η πιθανότητα να μην επιλεγεί σε καμία από αυτές, δηλαδή να μην ανήκει στο δείγμα, ισούται με P(x $\notin$ Δείγμα) = (1 – 1/*M*)^M. Άρα, P(x $\in$ Δείγμα) = 1 - (1 – 1/*M*)^M. Για μεγάλες τιμές του *M* έχουμε $\lim_{M\to\infty}(1 - (1 – 1/M)^M) = 1 – e^{-1} = 0.632$.

Η διαδικασία της δειγματοληψίας επαναλαμβάνεται *k* φορές. Αν υποθέσουμε ότι η ακρίβεια στην *i-οστή* επανάληψη είναι a_i, τότε η ακρίβεια *accuracy* της μεθόδου ορίζεται ως εξής:

$$accuracy = \frac{1}{k}\sum_{i=1}^{k}(0.632a_i + 0.368a)$$

Όπου *a* είναι η ακρίβεια όταν το σύνολο δεδομένων χρησιμοποιείται και ως σύνολο ελέγχου και ως σύνολο εκπαίδευσης.

ΑΛΓΟΡΙΘΜΟΙ ΤΑΞΙΝΟΜΗΣΗΣ

Έχοντας αναφερθεί στις βασικές έννοιες γύρω από το πλαίσιο της ταξινόμησης, μπορούμε πλέον να συνεχίσουμε με την περιγραφεί συγκεκριμένων αλγορίθμων κατηγοριοποιήσεις. Ποίο συγκεκριμένα, στα πλαίσια της παρούσας εργασίας θα χρησιμοποιηθούν οι εξής αλγόριθμοι:

- *BFTree*
- *J48*
- *J48Graft*
- *LADTree*
- *REPTree*
- *SimpleCart*
- *NaiveBayes*
- *NaiveBayesNominal*
- *NaiveBayesSimple*
- *Multinomial logistic regression*
- *SMO*
- *Linear logistic regression*
- *IB1*
- *DecisionTable*

Όπως αναφέρομαι και προηγουμένως, η ταξινόμηση εκτελείται σε 2 βήματα:

1. Δημιουργία του μοντέλου ταξινόμησης και,
2. Κατηγοριοποίηση των στιγμιότυπων του συνόλου ελέγχου.

Οι διαφορετικότητα των αλγορίθμων που θα παρουσιάσουμε, έγκυται στο ότι δημιουργούν διαφορετικά μοντέλα ταξινόμησης, ενώ το δεύτερο βήμα είναι ίδιο για όλους. Ο κάθε αλγόριθμος αναλύει διαφορετικά το σύνολο εκπαίδευσης, βασιζόμενος σε ξεχωριστά ποσοτικά μέτρα ο κάθε ένας προκειμένου να αντλήσει την κρυφή γνώση που κρύβεται πίσω από τα δεδομένα. Γι' αυτό τον λόγο κατασκευάζονται ανόμοια μοντέλα ταξινόμησης. Οι κατηγορίες στις οποίες ανήκουν τα μοντέλα αυτά είναι οι εξής:

1. *Δένδρα απόφασης (decision trees classifiers)*. Οι κατηγοριοποιητές αυτοι έχουν δενδρική μορφή αναπαράστασης. Σε αυτήν τη κατηγορία ανήκουν τα μοντέλα που σχηματίζουν οι αλγόριθμοι BFTree, J48, J48Graft, LADTree, REPTree και SimpleCart.
2. *Κανόνες ταξινόμησης (decision rule)*. Αποτελεί εναλλακτική αναπαράσταση των δένδρων απόφασης. Ο αλγόριθμός DecisionTable ανήκει σε αυτήν την κατηγορία.
3. *Πιθανοκρατικοί κατηγοριοποιητές (probabilistic classifiers)*. Βασίζονται στην εξέταση κατανομών πιθανότητας στα δεδομένα του συνόλου εκπαίδευσης. Οι αλγόριθμοι, NaiveBayes, NaiveBayesNominal και NaivebayesSimple ανήκουν σε αυτή τη κατηγορία.

4. *Διακριτικοί κατηγοριοποιητές (discriminative classifiers).* Οι Multinomial logistic regression και linear logistic regression ανήκουν σε αυτή τη κατηγορία.
5. *Γραμμικοί κατηγοριοποιητές (linear classifiers).* Βασίζονται στο γραμμικό συνδυασμό των χαρακτηριστικών του διανύσματος χαρακτηριστικών (feature vector). Στην κατηγορία αυτή ανήκει ο SMO.

2.2.1 C4.5

Ο αλγόριθμος C4.5 [11,12] ανήκει στην μεγάλη κατηγορία αλγορίθμων ταξινόμησης οι οποίοι δημιουργούν δεντρικά μοντέλα ταξινομητών (δένδρα απόφασης). Αποτελεί απόγονο του αλγορίθμου ID3.

Ένα δένδρο απόφασης αποτελείται από κόμβους που αντιστοιχούν σε κάποιο χαρακτηριστικό του συνόλου εκπαίδευσης ο κάθε ένας και διακρίνονται στους εξής:

1. *Ρίζα (root)*: κόμβος ο οποίος βρίσκεται στην κορυφή του δένδρου και χωρίζει το σύνολο εκπαίδευσης σε 2 ή περισσότερα υποσύνολα.
2. *Εσωτερικοί κόμβοι (internal node)*: ενδιάμεσοι κόμβοι οι οποίοι με την σειρά τους χωρίζουν το κάθε υποσύνολο του υποδένδρου σε μικρότερα υποσύνολα.
3. *Φύλλα (leafs)*: αποτελεί τον τερματικό κόμβο και αντιπροσωπεύει μια κλάση από το διακριτό σύνολο κλάσεων του συνόλου εκπαίδευσης.

Όλοι οι κόμβοι εκτός από τα φύλλα έχουν εξερχόμενες ακμές, οι οποίες αντιστοιχούν σε μια συνθήκη βάση της οποία γίνεται ο διάσπαση των δεδομένων. η συνθήκη αυτή ονομάζεται *συνθήκη διάσπασης (split criteria)*.

Algorithm 1.1 C4.5(D)

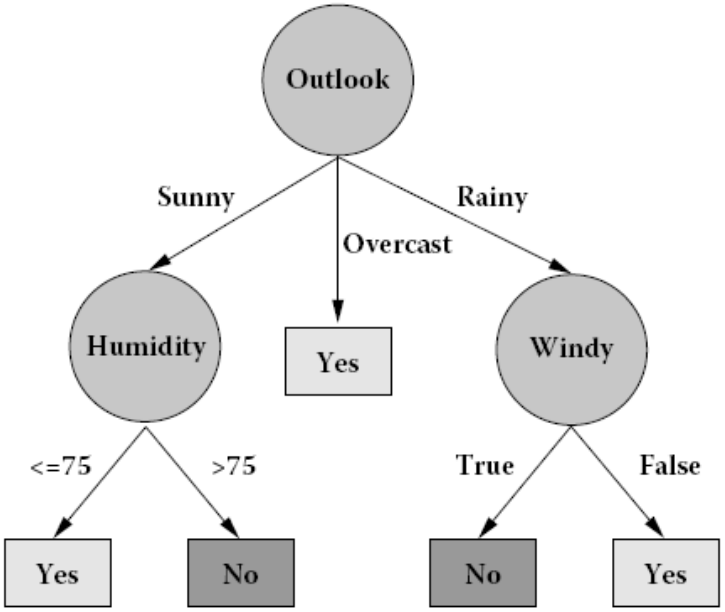
Input: an attribute-valued dataset D

```
1: Tree = {}
2: if D is "pure" OR other stopping criteria met then
3:    terminate
4: end if
5: for all attribute a ∈ D do
6:    Compute information-theoretic criteria if we split on a
7: end for
8: a_best = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests a_best in the root
10: D_v = Induced sub-datasets from D based on a_best
11: for all D_v do
12:    Tree_v = C4.5(D_v)
13:    Attach Tree_v to the corresponding branch of Tree
14: end for
15: return Tree
```

Εικόνα 2.3: Αλγόριθμος C4.5

Στην εικόνα 2.2 περιγράφεται ο αλγόριθμος C4.5. όπως βλέπουμε η διαδικασία ξεκινάει από την επιλογή του αρχικού κόμβου (ρίζα) ο οποίος θα διασπάσει το αρχικό σύνολο εκπαίδευσης με βάση την συνθήκη διάσπασης (split criteria). Η επιλογή του κατάλληλου κόμβου διαχωριστή, επιτυγχάνεται μέσω ενός ποσοτικού μέτρου. το μέτρο αυτό υπολογίζεται για όλα τα χαρακτηριστικά του συνόλου εκπαίδευσης και επιλέγεται το χαρακτηριστικό που έχει την καλύτερη τιμή. Η διαδικασία αυτή επαναλαμβάνεται αναδρομικά, έως ότου όλα τα μικρότερα υποσύνολα να είναι αμιγείς, δηλαδή, όλα τα στιγμιότυπά του υποσυνόλου να ανήκουν σε μία κλάση. Με αυτόν τον τρόπο κατασκευάζεται το δένδρο απόφασης. (βλέπε εικόνα 2.4)

Το βασικό ποσοτικό μέτρο που χρησιμοποιείται για επιλογή των διαχωριστών είναι το *κέρδος πληροφορίας (information Gain)* το οποίο βασίζεται στην *εντροπία πληροφορίας (entropy).*



Εικόνα 2.4:

Παράδειγμα από Δένδρο που προκύπτει από την εκτέλεση του C4.5

Έχουν προταθεί αρκετές παραλλαγές του C4.5, όπως C4.5-no-pruning, C4.5-rules, οι οποίοι προσπαθούν να βελτιώσουν την επίδοση του αλγορίθμου, εκτελώντας διάφορες επιπρόσθετες λειτουργίες (π.χ. κλάδεμα δένδρου).

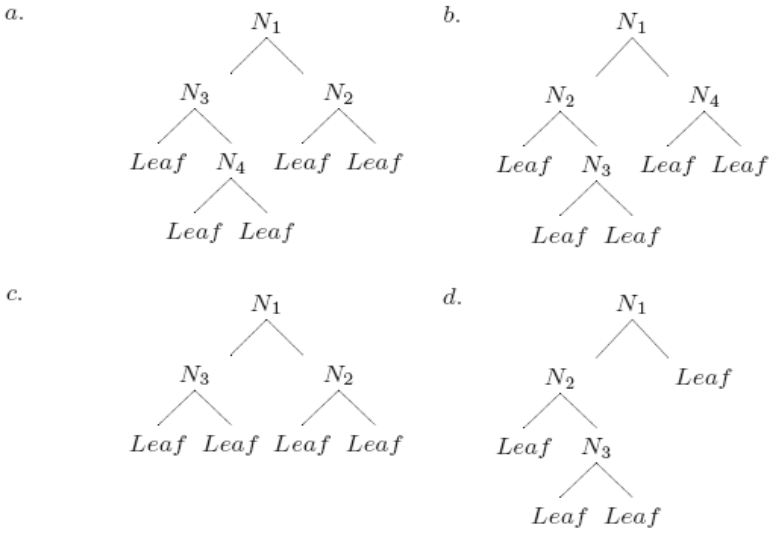
2.2.2 Best-first decision trees

Ο best-first αλγόριθμος ταξινόμησης [15], ανήκει στην μεγάλη κατηγορία των *δένδρων απόφασης (decision trees)*. Τα δένδρα απόφασης είναι μια δημοφιλής κατηγορία κατηγοριοποιητών για τους εξής λόγους:

1. Είναι γρήγορα τόσο κατά την κατασκευή μοντέλων ταξινόμησης όσο και κατά την διάρκεια της κατηγοριοποίησης.
2. Κατασκευάζουν μοντέλα ταξινόμησης με πολύ καλή επίδοση.
3. Εφαρμόζουν καλά σε «θορυβώδη» σύνολα εκπαίδευσης.
4. Είναι ιδανικά για την δημιουργία κατηγοριοποιητών, όταν υπάρχουν στιγμιότυπα με χαρακτηριστικά για τα οποία απουσιάζει η τιμή (missing values).

Τα best-first δένδρα συγκεντρώνουν όλες τις ιδιότητες των παραδοσιακών δένδρων απόφασης (π.χ. C4.5, ID3 κ.λπ.). Η μοναδική διαφορά είναι ότι, τα κλασσικά δένδρα απόφασης επεκτείνουν τους κόμβους (εσωτερικού κόμβους) με depth-first σειρά, ενώ ο best-first επεκτείνει πρώτα το «καλύτερο» κόμβο.

Η εικόνα 2.5 παρουσιάζει ένα υποθετικό best-first δένδρο και ένα κλασικό δένδρο απόφασης. Διακρίνεται ότι ο best-first δημιουργεί ένα πλήρως-επεκταμένο δένδρο (εικόνα 2.5 (α)) που είναι ίδιος με το δένδρο που δημιουργούν οι παραδοσιακοί αλγόριθμοι δένδρων απόφασης. Παρόλα αυτά, αν είναι γνωστό εκ των προτέρων το πλήθος των επεκτάσεων που θα γίνουν στο δένδρο, ο best-first δημιουργεί σχεδόν πάντα διαφορετικά δένδρα (εικόνες 2.5 (c) και 2.5 (d) αντίστοιχα). Αυτό οφείλεται στον τρόπο που αναπτύσσει το δένδρο ο best-first.



Εικόνα 2.5

(a) πλήρως-επεκταμένο best-first tree.(b)πλήρως επεκταμένο παραδοσιακό δένδρο απόφασης
(c) best-first tree με τρείς επεκτάσεις.(d) παραδοσιακό δένδρο απόφασης με τρεις επεκτάσεις

```
function BFTree (A: a set of attributes,
                 E: the training instances,
                 N: the number of expansions,
                 M: the minimal number of instances at a terminal node
) return a decision tree
    begin
        If E is empty, return failure;
        Calculate the reduction of impurity for each attribute
              in A on E at the root node RN;
        Find the best attribute A_b in A;
        Initialise an empty list NL to store nodes;
        Add RN (with E and A_b) into NL;
        expandTree(NL, N, M);
        return a tree with the root RN;
    end


expandTree(NL, N, M)
    begin
        If NL is empty, return;
        Get the first node FN from NL;
        Retrieve training instances E and the best splitting attribute A_b of FN;
        If E is empty, return failure;
        If the reduction of impurity of FN is 0 or N is reached,
            Make all nodes in NL into terminal nodes;
            return;
        If the split of FN on A_b would result in a successor node
                with less than M instances,
            Make FN into the terminal node;
            Remove FN from NL;
            expandTree(NL, N, M);
        Let SN_1 and SN_2 be the successor nodes generated by
            splitting FN on A_b on E;
        Increment the number of expansions by one;
        Let E_1 and E_2 be the subsets of instances corresponding to
            SN_1 and SN_2;
        Find the corresponding best attributes A_b1 for SN_1;
        Find the corresponding best attributes A_b2 for SN_2;
        Put SN_1 (with E_1 and A_b1) and SN_2 (with E_2 and A_b2)
            into NL according to the reduction of impurity;
        Remove FN from NL;
        expandTree(NL, N, M);
    end
```

Εικόνα 2.6
Αλγόριθμος Best-first

Όπως αναφέραμε παραπάνω, ο best-first αλγόριθμος επεκτείνει το δέντρο, επιλέγοντας να διαχωρίσει το «καλύτερο» κόμβο σε κάθε γύρω. Με το όρο «καλύτερο» εννοούμε τον κόμβο ο οποίος οδηγεί σε μεγαλύτερη μείωση της ανομοιογένειας των δεδομένων στους κόμβου. Ενώ τα παραδοσιακά δένδρα

απόφασής, αναπτύσσουν το δέντρο πρώτα σε «βάθος», δηλαδή επεκτείνουν τους κόμβους ενός μονοπατιού, μέχρι να φτάσουν σε τερματικό κόμβο. Αυτό αποτελεί την βασική και μοναδική διαφορά των 2 προσεγγίσεων. Προκειμένου να επιλέγει γρήγορα τον «καλύτερο» κόμβο, ο best-first ταξινομεί την λίστα με τους κόμβους, κατά φθίνουσα σειρά ως προς το Gini gain ή το Information gain (ανάλογα με το ποίο ποσοτικό μέτρο χρησιμοποιείται για τον διαχωρισμό των κόμβων). Με αυτό τον τρόπο, κάθε φορά επιλέγεται για διαχωρισμό, ο πρώτος κόμβος τις ταξινομημένης ο οποίος ουσιαστικά είναι και ο «καλύτερος». Ένα επιπλέον χαρακτηριστικό του best-first είναι ότι , μπορούμε να καθορίσουμε το πλήθος των επεκτάσεων που θα γίνουν στο δένδρο. Ο αλγόριθμος παρουσιάζεται στην εικόνα 2.6

2.2.3 LADTrees

Ο *LADTree* [13] είναι αλγόριθμος ταξινόμησης ο οποίος δημιουργεί ένα *εναλλακτικό δένδρο απόφασης (alternative decision tree)*. Εφαρμόζεται σε προβλήματα ταξινόμησης στα οποία έχουν πολλαπλές κλάσεις (multiclass). Βασίζεται στο αλγόριθμο *LogitBoost* ο οποίος ανήκει στην οικογένεια των Boosting αλγορίθμων. Προτού περιγράψουμε τα εναλλακτικά δένδρα, θα ήταν χρήσιμο να αναφερθούμε στην διαδικασία του *Boosting* [12] στην οποία στηρίζει την λειτουργία του ο LogitBoost (logistic boosting).

2.2.3.1 Boosting

Υποθέτουμε ότι έχουμε ένα πρόβλημα δυαδικής ταξινόμησης, δηλαδή το κάθε στιγμιότυπο μπορεί να ανήκει σε μια από τις δυο εναλλακτικές κλάσεις. Όπως είναι γνωστό, σκοπός ενός αλγορίθμου ταξινόμησης είναι να κατασκευάσει ένα μοντέλο ταξινόμησης μέσα από το σύνολο εκπαίδευση, και στην συνέχεια να προσπαθήσει να κατατάξει όλα τα στιγμιότυπα του συνόλου ελέγχου. Μία εναλλακτική λύση θα μπορούσε να είναι, ο κατηγοροιοποιητής να κατατάσσει τα στιγμιότυπα με τυχαίο τρόπο χωρίς να χρειαστεί να αναλύσει τα δεδομένα. Έτσι θα είχε 50% απώλεια (αριθμός των λάθος κατατάξεων). Είναι προφανής ότι αυτή η λύση δεν μπορεί να είναι αποδεκτή.

Υπάρχουν όμως περιπτώσεις στις οποίες οι κατηγοριοποιητές, παρόλη την ανάλυση που εκτελούν στα δεδομένα, δεν επιτυγχάνουν ποσοστό σωστής πρόβλεψης πολύ μεγαλύτερο από ότι αν εφαρμόζαμε τυχαία πρόβλεψη. Σε αυτές τις περιπτώσεις λέμε ότι ο κατηγοριοποιητής μας αποτελεί έναν *αδύναμο κατηγοροιοποιητής ή μια αδύναμη υπόθεση (weak classifier ή weak hypothesis)*. Η λογική του Boosting στηρίζεται στο γεγονός ότι μπορούμε να συνδυάσουμε ένα σύνολο αδύναμων κατηγοροιοποιητών προκειμένου να σχηματίσουμε έναν *δυνατό κατηγοροιοποιητή (strong classifier)* ο οποίος θα έχει πολύ καλύτερα αποτελέσματα από την τυχαία πρόβλεψη.

Αν θέσουμε έναν αδύναμο κατηγοροιοποιητή ως *h1,* τότε η διαδικασία του Boosting θα παράγει μια νέα κατανομή *D'* των δεδομένων από την αρχική *D* προκειμένου να ελαττωθούν τα λάθη που εμφάνισε ο *h1*. Ουσιαστικά πρόκειται για μια αλλαγή που γίνεται στα βάρη των στιγμιότυπων. Συνήθως η νέα *D'* εστιάζει την προσοχή στα στιγμιότυπα τα οποία ο *h1* κατάταξε εσφαλμένα. Στην συνέχεια μέσα από την νέα

κατανομή δημιουργείται ο νέος κατηγοριοποιητής *h2*. Ο *h2* είναι αναμενόμενο να έχει καλύτερα αποτελέσματα από τον *h1* και άρα ένας πιθανός συνδυασμός των δύο θα οδηγούσε στην δημιουργία ενός ενιαίου δυνατού κατηγοριοποιητή. Η παραπάνω διαδικασία εκτελείται σε ένα πεπερασμένο αριθμό βημάτων .

Input: Instance distribution $\mathcal{D}$;
Base learning algorithm L;
Number of learning rounds T.
Process:
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \cdots, T$:
3. $h_t = L(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $e_t = \Pr_{x \sim \mathcal{D}_t, y} I[h_t(\boldsymbol{x}) \neq y]$; % Measure the error of h_t
5. $\mathcal{D}_{t+1} = AdjustDistribution\ (\mathcal{D}_t, e_t)$
6. **end**
Output: $H(\boldsymbol{x}) = CombineOutputs(\{h_t(\boldsymbol{x})\})$

Εικόνα 2.7
Η μέθοδος Boosting

Τα εναλλακτικά δένδρα απόφασης (ADTrees) εφαρμόζουν έναν μηχανισμό, ο οποίος συνδυάζει όλους τους αδύναμους κατηγοριοποιητές που σχηματίστηκαν κατά την διαδικασία του Boosting σε μια ενιαία και κατανοητή μορφή. Ο αλγόριθμος παρουσιάζεται στην εικόνα 2.8. Σε κάθε επανάληψη *t* του Boosting ο αλγόριθμος διατηρεί δύο σύνολα, το σύνολο των προϋποθέσεων και το σύνολο των κανόνων, P_t και R_t αντίστοιχα. Επίσης ένα σύνολο *C* από αδύναμους κατηγοριοποιητές παράγεται σε κάθε επανάληψη.

2.2.4 Bayesian classifiers

Οι κατηγοροιοποιητές Bayes [7,12,1] ανήκουν στους πιθανοκρατικούς αλγορίθμους ταξινόμησης. Πιο συγκεκριμένα, εξετάζουν την κατανομή πιθανοτήτων στα δεδομένα εκπαίδευσης, προκειμένου να υπολογίσουν την πιθανότητα ένα στιγμιότυπο να ανήκει σε μια συγκεκριμένη κλάση. Όπως αποκαλύπτει και το όνομα, οι Bayesian κατηγοροιοποιητές βασίζονται στο γενικό θεώρημα του Bayes (Bayes' theorem) το οποίο περιγράφεται παρακάτω. Έχουν προταθεί αρκετοί αλγόριθμοι γι' αυτήν την κατηγορία (π.χ. naïve Bayes, naïve Bayes nominal, Bayesian belief network), εμείς θα αναφερθούμε στην συνέχεια στον naïve Bayes.

1. Initialize Set the weights $w_{i,t}$ associated with each training instance to 1. Set the first rule $\mathcal{R}_1$ to have a precondition and condition which are both true. Calculate the prediction value for this rule as $a = \frac{1}{2} \ln \frac{W_+(c)}{W_-(c)}$ where $W_+(c)$, $W_-(c)$ are the total weights of the positive and negative instances that satisfy condition c in the training data. The initial value of c is simply True.

2. Pre-adjustment Reweight the training instances using the formula $w_{i,1} = w_{i,0}e^{-ay_t}$ (for two-class problems, the value of y_t is either +1 or -1).

3. Repeat for t = 1, 2, ..., $\mathcal{T}$

(a). Generate the set $\mathcal{C}$ of weak hypotheses using the weights associated with each training instance $w_{i,t}$

(b). For each base precondition $c_1 \in \mathcal{P}_t$ and each condition $c_2 \in \mathcal{C}$ calculate

$$Z_t(c_1, c_2) = 2\left(\sqrt{W_+(c_1 \wedge c_2)W_-(c_1 \wedge c_2)} + \sqrt{W_+(c_1 \wedge \neg c_2)W_-(c_1 \wedge \neg c_2)}\right) + W(\neg c_1)$$

(c). Select c_1, c_2 which minimize $Z_t(c_1, c_2)$ and set $\mathcal{R}_{t+1}$ to be $\mathcal{R}_t$ with the addition of the rule r_t whose precondition is c_1, condition is c_2 and two prediction values are:

$$a = \frac{1}{2} \ln \frac{W_+(c_1 \wedge c_2) + \epsilon}{W_-(c_1 \wedge c_2) + \epsilon}, \qquad b = \frac{1}{2} \ln \frac{W_+(c_1 \wedge \neg c_2) + \epsilon}{W_-(c_1 \wedge \neg c_2) + \epsilon}$$

(d). Set $\mathcal{P}_{t+1}$ to be $\mathcal{P}_t$ with the addition of $c_1 \wedge c_2$ and $c_1 \wedge \neg c_2$.

(e). Update the weights of each training example according to the equation

$$w_{i,t+1} = w_{i,t}e^{-r_t(x_i)y_t}$$

4. Output the classification rule that is the sign of the sum of all the base rules in $R_{\mathcal{T}+1}$:

$$class(x) = sign\left(\sum_{t=1}^{\mathcal{T}} r_t(x)\right)$$

Εικόνα 2.8
Αλγόριθμος ADTrees

2.2.4.1 Θεώρημα Bayes

Το θεώρημα Bayes έχει πάρει το όνομα από τον Thomas Bayes. Αποτελεί μία απλή μαθηματική εξίσωση που χρησιμοποιείται για τον υπολογισμό υπό συνθήκη πιθανοτήτων (posterior probability). Έστω *X* μια πλειάδα η οποία αποτελείται από *n* χαρακτηριστικά τα οποία την περιγράφουν. Θέτουμε ως *H* την υπόθεση η πλειάδα *X* να ανήκει σε μια συγκεκριμένη κλάση *C* από ένα διακριτό σύνολο *m* στοιχείων. Στα προβλήματα κατηγοριοποίησης, προσπαθούμε να υπολογίσουμε την *P(H|X),* δηλαδή

την πιθανότητα η πλειάδα *X* (ή το στιγμιότυπο *X* σε όρους αλγορίθμων μάθησης) να ανήκει στην κλάση *C,* δεδομένου ότι γνωρίζουμε την περιγραφή της *X (δηλαδή το διάνυσμα χαρακτηριστικών)*. Η πιθανότητα *P(H|X)* είναι γνωστή και ως, υπό συνθήκη πιθανότητα του *H* δεδομένου του *X.*
Για να απλοποιήσουμε λίγο την περιγραφή του θεωρήματος, θα παραθέσουμε ένα απλό παράδειγμα. Θεωρούμε ότι τα στιγμιότυπα μας περιγράφουν πελάτες και αποτελούνται από 2 χαρακτηριστικά, την ηλικία και το εισόδημα. Έστω *X* ένας πελάτης ηλικίας 20 ετών με εισόδημα 30.000 ευρώ. Αν *H* η υπόθεση ο πελάτης *X* να αγοράσει έναν υπολογιστή, τότε η *P(H|X)* αποτυπώνει την πιθανότητα ότι ο πελάτης *X* θα αγοράσει έναν υπολογιστή δεδομένου ότι γνωρίζουμε την περιγραφή του, δηλαδή την ηλικία και το εισόδημα.
Το θεώρημα του Bayes μας παρέχει έναν σχετικά απλό τρόπο για να υπολογίσουμε την υπό συνθήκη πιθανότητα *P(H|X)*. το θεώρημα περιγράφεται από τον τύπο:

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

Στον παράδειγμά μας η *P(H)* είναι πιθανότητα ένας οποιοδήποτε πελάτης να αγοράσει έναν υπολογιστή ανεξαρτήτως ηλικίας και εισοδήματος. Η *P(H|X)* αποτελεί την εκ των προτέρων πιθανότητα του *H* με την έννοια ότι δεν απαιτείται να γνωρίζουμε επιπρόσθετες πληροφορίες για κάποιον πελάτη.

Η *P(X|H)* είναι η υπό συνθήκη πιθανότητα του *X* δεδομένου ότι γνωρίζουμε την *H.* Δηλαδή η πιθανότητα ένας πελάτης να είναι 20 ετών και να έχει εισόδημα 30.000 ευρώ, όταν γνωρίζουμε ότι θα αγοράσει έναν υπολογιστή.

Τέλος η *P(X)* είναι εκ των προτέρων πιθανότητα του *X.* δηλαδή είναι η πιθανότητα ένας πελάτης να είναι 20 ετών και να έχει εισόδημα 30.000 ευρώ.

2.2.4.2 Αφελής Bayes (Naïve Bayes)

Αποτελεί τον πιο απλό αλγόριθμο της κατηγορίας των Bayesian αλγορίθμων ταξινόμησης. Στηρίζει την λειτουργία του στην υπόθεση ότι τα χαρακτηριστικά ενός στιγμιότυπου είναι ανεξάρτητα μεταξύ του. Τα βήματα εκτέλεσης είναι τα εξής:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, \ldots, A_n$.
2. Suppose that there are m classes, $C_1, C_2, \ldots, C_m$. Given a tuple, $\mathbf{X}$, the classifier will predict that $\mathbf{X}$ belongs to the class having the highest posterior probability, conditioned on $\mathbf{X}$. That is, the naïve Bayesian classifier predicts that tuple $\mathbf{X}$ belongs to the class C_i if and only if

$$P(C_i|\mathbf{X}) > P(C_j|\mathbf{X}) \quad \text{for } 1 \le j \le m, j \ne i.$$

Thus we maximize $P(C_i|\mathbf{X})$. The class C_i for which $P(C_i|\mathbf{X})$ is maximized is called the *maximum posteriori hypothesis*. By Bayes' theorem (Equation (6.10)),

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}. \qquad (6.11)$$

3. As $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize $P(X|C_i)$. Otherwise, we maximize $P(X|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D.

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(X|C_i)$. In order to reduce computation in evaluating $P(X|C_i)$, the naive assumption of **class conditional independence** is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned} \tag{6.12}$$

We can easily estimate the probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ from the training tuples. Recall that here x_k refers to the value of attribute A_k for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute $P(X|C_i)$, we consider the following:

(a) If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_{i,D}|$, the number of tuples of class C_i in D.

(b) If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{6.13}$$

so that

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \tag{6.14}$$

5. In order to predict the class label of X, $P(X|C_i)P(C_i)$ is evaluated for each class C_i. The classifier predicts that the class label of tuple X is the class C_i if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i. \tag{6.15}$$

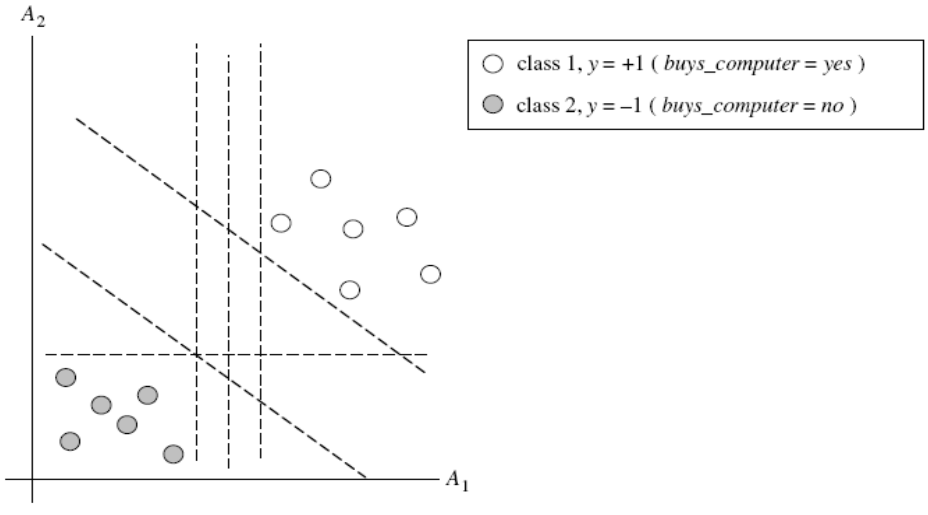
In other words, the predicted class label is the class C_i for which $P(X|C_i)P(C_i)$ is the maximum.

2.2.5 Support Vector Machines

Αποτελεί έναν σχετικά νέο αλγόριθμο ταξινόμησης, ο οποίος μπορεί να εφαρμοστεί τόσο σε γραμμικά όσο και σε μη γραμμικά δεδομένα (η διαφορά τους περιγράφεται παρακάτω). Συνοπτικά θα μπορούσαμε να περιγράψουμε την λειτουργία του αλγορίθμου ως εξής. Έστω ότι έχουμε ένα πρόβλημα δυαδικής ταξινόμησης. Αρχικά ο αλγόριθμος προσπαθεί να μετατρέψει τα αρχικά δεδομένα σε μια ανώτερη διάσταση. Στην συνέχεια αναζητάει στην διάσταση αυτή το βέλτιστο γραμμικό υπέρ-επίπεδο, το οποίο διαχωρίζει τα στιγμιότυπα της μιας κλάσης από τα στιγμιότυπα της άλλης κλάσης και αποτελεί ουσιαστικά το μοντέλο ταξινόμησης. Το υπέρ-επίπεδο αυτό αναπαριστάται από μια εξίσωση. Για να κατατάξει ένα στιγμιότυπο του συνόλου ελέγχου, ο αλγόριθμος ελέγχει το αποτέλεσμα της εξίσωσης και ανάλογα καθορίζει και την κλάση του εκάστοτε στιγμιότυπου.

2.2.5.1 Γραμμικώς διαχωρίσιμα δεδομένα

Έστω ότι έχουμε ένα πρόβλημα δυαδικής ταξινόμησης, όπου τα στιγμιότυπα της κάθε κλάσης είναι γραμμικώς διαχωρίσιμα. Θεωρούμε το σύνολο δεδομένων D το οποίο αποτελείται από ζεύγη (X_1,y_1), (X_2,y_2), ... , $(X_{|D|},y_{|D|})$ όπου X_i το διάνυσμα χαρακτηριστικών του κάθε στιγμιότυπου και y_i η αντίστοιχη κλάση. Τα y_i μπορούν να πάρουν δυο τιμές, είτε +1 ή -1. Η εικόνα 2.9 δείχνει ένα παράδειγμα στο οποίο όλα τα στιγμιότυπα έχουν δυο χαρακτηριστικά, το A_1 και A_2. Όπως είναι φανερό, τα δεδομένα μας είναι γραμμικώς διαχωρίσιμα, δηλαδή υπάρχει μία τουλάχιστον ευθεία η οποία διαχωρίζει όλα τα στιγμιότυπα της κλάσης +1 από τα στιγμιότυπα της κλάσης -1. Προφανώς, υπάρχουν άπειρες τέτοιες ευθείες γραμμές. Το πρόβλημα έγκειται στην εύρεση της βέλτιστης ευθείας η οποία θα επιφέρει το μικρότερο σφάλμα ταξινόμησης. Στην γενικότερη περίπτωση όπου το διάνυσμα χαρακτηριστικών περιέχει πάνω από τρία στοιχεία, ο αλγόριθμος αναζητάει το βέλτιστο *υπέρ-επίπεδο(hyperplane).*



Εικόνα 2.9
Δεδομένα στις 2 διαστάσεις που είναι γραμμικώς διαχωρίσιμα. Υπάρχουν άπειρες ευθείες οι οποίες διαχωρίζουν τα δεδομένα.

Στην εικόνα 2.10 παρουσιάζονται δύο πιθανά υπέρ-επίπεδα που διαχωρίζουν τα δεδομένα καθώς και τα αντίστοιχα *περιθώρια (margin)*. Διαισθητικά αναμένουμε ότι το υπέρ-επίπεδο με το μεγαλύτερο περιθώριο θα μπορεί να κατατάσσει τα στιγμιότυπα του συνόλου ελέγχου με μεγαλύτερη ακρίβεια (και κατ' επέκταση με μικρότερο σφάλμα κατηγοριοποίησης) σε σχέση με το υπέρ-επίπεδο με το μικρότερο περιθώριο. Όπως αναφέραμε προηγουμένως, ο *S.V.M (Support Vector Machine)* αναζητάει το βέλτιστο υπέρ-επίπεδο το οποίο αντιστοιχεί στο υπέρ-επίπεδο με το μεγαλύτερο περιθώριο (*Maximum Marginal Hyperplane ή MMH*).

Το διαχωριστικό υπέρ-επίπεδο μπορεί να παρασταθεί με τον μαθηματικό τύπο:

$$W \cdot X + b = 0$$

Όπου W το διάνυσμα με τα βάρη των στιγμιότυπων $W = \{w_1, w_2, \ldots, w_n\}$, n το πλήθος των χαρακτηριστικών και b η κλίση του υπέρ-επιπέδου.

Για την εικόνα 2.10(b) η παραπάνω σχέση γράφεται ως εξής:

$$w_0 + w_1x_1 + w_2x_2 = 0$$

Στην σχέση αυτήν, θέσαμε την κλίση b ως επιπλέον βάρος w_0.

Είναι προφανές ότι οποιοδήποτε σημείο βρίσκεται πάνω από υπέρ-επίπεδο ικανοποιεί την σχέση:

$$w_0 + w_1x_1 + w_2x_2 > 0$$

Και αντίστοιχα τα σημεία που βρίσκονται κάτω από το υπέρ-επίπεδο ικανοποιούν την σχέση:

$$w_0 + w_1x_1 + w_2x_2 < 0$$

προσαρμόζοντας τα βάρη w_i μπορούμε αναπαραστήσουμε τις δυο «πλευρές» του περιθωρίου με τις παρακάτω σχέσεις:

$$HP_1: \; w_0 + w_1x_1 + w_2x_2 \geq +1 \; \gamma\iota\alpha \; y_i = +1$$

$$HP_2: \; w_0 + w_1x_1 + w_2x_2 \leq -1 \; \gamma\iota\alpha \; y_i = -1$$

Δηλαδή κάθε στιγμιότυπο το οποίο βρίσκεται πάνω από την HP_1 ανήκει στην κλάση +1 και αντίστοιχα κάθε στιγμιότυπο το οποίο βρίσκεται κάτω από την HP_2 ανήκει στην κλάση -1. Αν συνδυάσουμε τις δυο παραπάνω σχέσεις, προκύπτει

$$y_i(w_0 + w_1x_1 + w_2x_2) \geq +1 \;\; \forall \, i$$

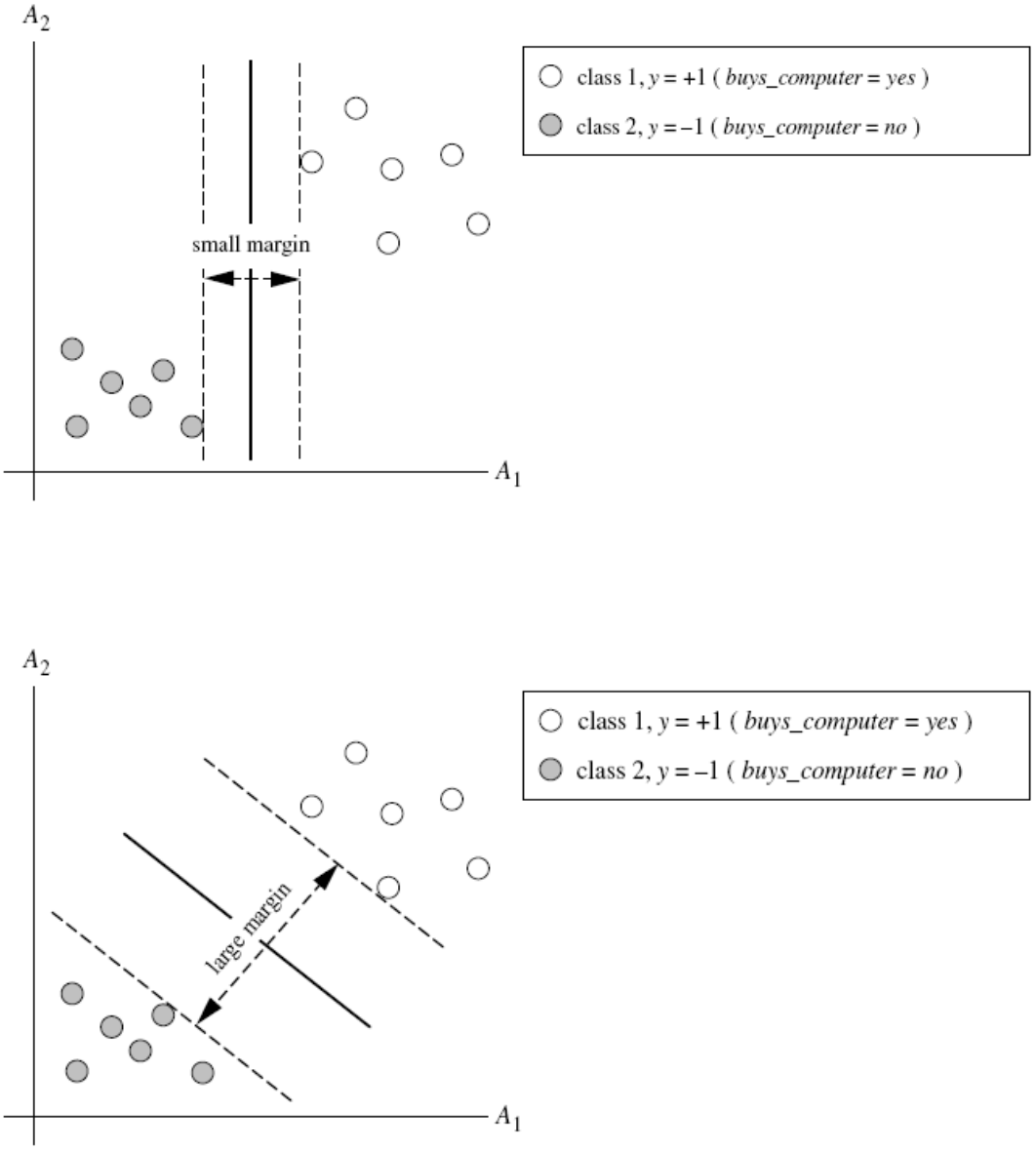
Τα στιγμιότυπα του συνόλου εκπαίδευσης τα οποία βρίσκονται πάνω στην HP_1 ή HP_2 ονομάζονται *διανύσματα υποστήριξης (support vectors)*. Στην εικόνα 2.11 τα support vectors είναι τα σημεία με τον έντονο κύκλο.

Εφαρμόζοντας μια σειρά από μαθηματικές πράξεις [1] καταλήγουμε στην σχέση:

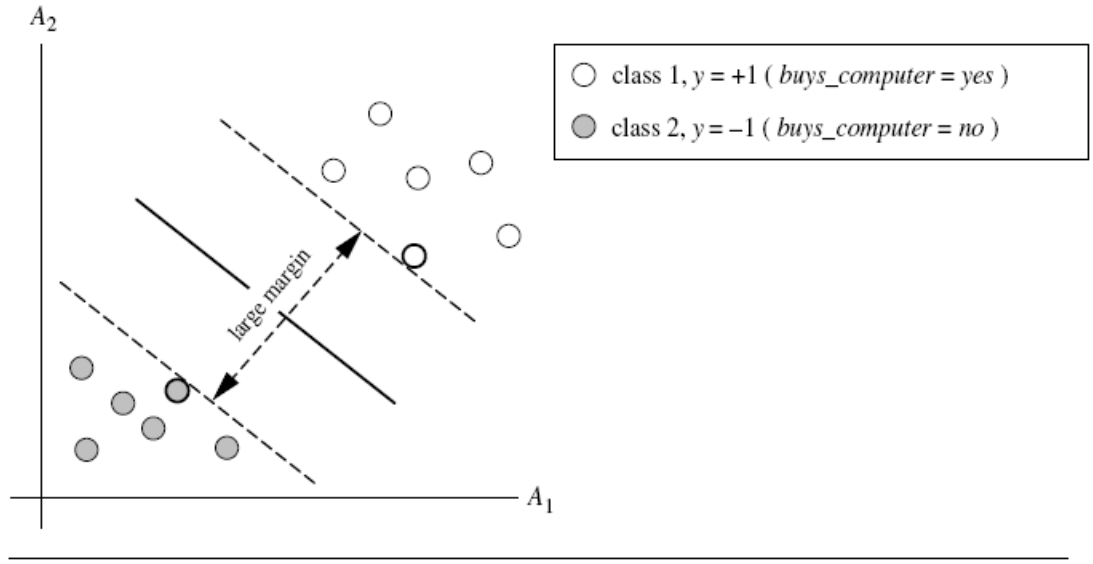
$$d(X^T) = \sum_{i=1}^{l} y_i a_i X_i X^T + b_o$$

Όπου y_i η κλάση στην οποία ανήκουν τα διανύσματα υποστήριξης X_i , X^T το στιγμιότυπο από το σύνολο ελέγχου. a_i και b_0 είναι κάποιες αριθμητικές παράμετροι που προκύπτουν κατά την διαδικασία εκτέλεσης του αλγορίθμου. τέλος l είναι το πλήθος των διανυσμάτων υποστήριξης.

Δεδομένου ενός στιγμιότυπου X^T από το σύνολο ελέγχου, εφαρμόζουμε την παραπάνω σχέση και εξετάζουμε το πρόσημο του αποτελέσματος. Από το πρόσημο προκύπτει εάν το στιγμιότυπο θα βρίσκεται πάνω ή κάτω από την *ΜΜΗ* και κατ' επέκταση κλάση στην οποία αυτό ανήκει.



Εικόνα 2.10
Δυο πιθανά υπέρ-επίπεδα και τα αντίστοιχα περιθώρια τους.



Εικόνα 2.11
Support vectors

COST-SENSITIVE ΜΑΘΗΣΗ

Στο προηγούμενο εδάφιο περιγράψαμε τη λειτουργία βασικών αλγορίθμων ταξινόμησης. Όπως αναφέραμε, οι αλγόριθμοι αυτοί έχουν ως στόχο την ανάθεση του κάθε αντικειμένου του συνόλου ελέγχου σε μια κλάση από ένα πεπερασμένο σύνολο κλάσεων. Οι πλειοψηφία αυτών, προσπαθεί να ελαχιστοποιήσει το *ποσοστό σφάλματος*, δηλαδή το ποσοστό των λανθασμένων προβλέψεων. Στην προσπάθεια τους αυτή, θεωρούν ότι όλες οι άστοχες προβλέψεις έχουν το ίδιο *κόστος*. Όμως στις περισσότερες εφαρμογές της εξόρυξης δεδομένων η υπόθεση αυτή δεν ισχύει.

Για παράδειγμα, έστω σε μια εφαρμογή διάγνωσης ενός συγκεκριμένου τύπου καρκίνου έχουμε δυο κλάσεις, τη θετική κλάση η οποία δηλώνει ότι ο ασθενής έχει καρκίνο και την αρνητική κλάση δηλαδή ο ασθενής είναι υγιείς. Η πρόβλεψη ότι ένας ασθενής είναι υγιείς ενώ στην πραγματικότητα πάσχει από καρκίνο, είναι πολύ πιο σοβαρή από την αντίστροφη πρόβλεψη (δηλαδή την πρόβλεψη ο ασθενής να έχει καρκίνο ενώ στην ουσία να είναι υγιείς.). Η πρώτη περίπτωση μπορεί να οδηγήσει τον ασθενή σε θάνατο, γιατί ο ασθενής θα αργήσει να ξεκινήσει την θεραπεία. Ομοίως, η λάθος πρόβλεψη ότι ένας επιχειρηματίας πρόκειται να προσφέρει δωρεά για έναν φιλανθρωπικό σκοπό, έχει μεγαλύτερο κόστος από την αντίστροφη πρόβλεψη, γιατί θα χρειαστεί να πληρώσουμε τα έξοδα αποστολής της πρόσκλησης στον επιχειρηματία.

Όπως γίνεται αντιληπτό, υπάρχουν πάρα πολλές εφαρμογές οι οποίες θα θέλαμε να έχουν μεγάλο ποσοστό επιτυχίας, αλλά παράλληλα να λαμβάνουν υπ' όψιν και το κόστος της πρόβλεψης. Το είδος της μάθησης κατά τα την οποία οι αλγόριθμοι συνυπολογίζουν το κόστος της λανθασμένης πρόβλεψης, ονομάζεται ***cost-sensitive***

μάθηση[14]. Αποτελεί μια σχετικά καινούργια και εναλλακτική προσέγγιση της κλασικής έννοιας της μηχανικής μάθησης, γύρω από την οποία παρατηρείται τα τελευταία χρόνια, έντονη δραστηριότητα της επιστημονικής κοινότητας που ασχολείται με την περιοχή της τεχνητής νοημοσύνης .

Στις γραμμές που ακολουθούν, αρχικά θα περιγράψουμε το γενικό πλαίσιο της cost-sensitive μάθησης και στη συνέχεια θα δώσουμε ορισμένα παραδείγματα cost-sensitive αλγορίθμων που έχουν προταθεί τα τελευταία χρόνια.

2.3.1 Περιγραφή cost-sensitive μάθησης

Τα τελευταία χρόνια έχουν παρουσιαστεί πολλές έρευνες πάνω στο θέμα τις cost-sensitive μάθησης, οι οποίες περιγράφουν διάφορες προσεγγίσεις για το πώς θα μπορούσαμε να εφαρμόσουμε την τεχνική αυτή στις KDD εφαρμογές. Γενικότερα, κατατάσσουμε την cost-sensitive μάθηση στις εξής 2 κατηγορίες:

1. Η πρώτη κατηγορία ασχολείται με τον σχεδιασμό αλγορίθμων ταξινόμησης οι οποίοι διαφέρουν από τους κλασσικούς που γνωρίζουμε, στο γεγονός ότι συμπεριλαμβάνουν την έννοια του κόστους. Παραδείγματα αλγορίθμων είναι τα cost-sensitive δένδρα απόφασης, hybrid genetic decision trees κ.α.
2. Στην δεύτερη κατηγορία ανήκουν η επονομαζόμενοι αλγόριθμοι «περιτυλίγματα» (wrappers). Η προσέγγιση αυτή αποτελεί ένα γενικότερο τρόπο υλοποίησης της cost-sensitive μάθησης, στην οποία μπορούμε να μετατρέψουμε οποιοδήποτε υπάρχον cost-insensitive αλγόριθμο σε cost-sensitive, χωρίς να τον μεταβάλουμε. Τέτοιοι αλγόριθμοι είναι ο MetaCost[3], Costing [4], CSRoulette [6] και ο CostSensitiveClassifier.

Οι αλγόριθμοι της πρώτης κατηγορία ονομάζονται ***direct Methods*** ενώ οι αλγόριθμοι της δεύτερης κατηγορίας ονομάζονται ***meta-learning methods***. Στην πράξη προτιμούνται οι meta-learning αλγόριθμοι γιατί επιτρέπουν την επαναχρησιμοποίηση αλγορίθμων ταξινόμησης που είδη υπάρχουν, χωρίς να κάνουμε κάποιες αλλαγές. Διακρίνουμε τρείς διαφορετικές τεχνικές υλοποίησης των meta-learners:

1. Η πρώτη τεχνική επανακαθορίζει (*Relabeling*) τις κλάσεις των στιγμιότυπων, εφαρμόζοντας το κριτήριο του ελάχιστου αναμενόμενου κόστους (Minimum Expected Cost criterion). Η προσέγγιση αυτή μπορεί να υλοποιηθεί με δυο τρόπους:
    I. Επανακαθορισμός των κλάσεων των στιγμιότυπων εκπαίδευσης (π.χ. MetaCost) και
    II. Επανακαθορισμός των κλάσεων των στιγμιότυπων ελέγχου (π.χ. CostSensitiveClassifier).

2. Η δεύτερη τεχνική ονομάζεται *Weighting*. Εδώ αναθέτουμε σε κάθε στιγμιότυπο ένα βάρος, σύμφωνα με το κόστος της λανθασμένης ταξινόμησης.

3. Η Τρίτη και τελευταία τεχνική είναι η *Sampling*, στην οποία γίνεται μεταβολή της κατανομής του συνόλου εκπαίδευσης, σύμφωνα με το κόστος της λανθασμένης πρόβλεψης (π.χ. Costing και CSRoulette).

2.3.1.1 Πλαίσιο λειτουργίας Cost-Sensitive μάθησης

Σε αυτό το σημείο θα περιγράψουμε τη θεωρία στην οποία στηρίζεται η cost-sensitive μάθηση. Θα δείξουμε πώς οι cost-sensitive αλγόριθμοι εκμεταλλεύονται την πληροφορία για το κόστος λάθους ταξινόμησης. Για λόγους απλότητας, θα παραθέσουμε ένα παράδειγμα.

Έστω ότι έχουμε ένα πρόβλημα δυαδικής ταξινόμησης, όπου η μια κλάση είναι η *Positive* και η άλλη η *Negative.* Θα συμβολίζουμε με *FP (false positive)* το κόστος της λανθασμένης πρόβλεψης *Positive (*δηλαδή κατάταξη του στιγμιότυπου στην κλάση *Positive* ενώ αυτό στην πραγματικότητα ανήκει στην κλάση *Negative)* και αντίστοιχα με *FN (false negative)* θα συμβολίζουμε το κόστος της αντίστροφης λανθασμένης πρόβλεψης. Με *TP* και *TN* θα συμβολίζουμε τις αντίστοιχες σωστές προβλέψεις. Τις τιμές των *FP, FN, TP* και *TN* τις οποίες εναλλακτικά τις γράφουμε ως *C(i,j,)* τις συγκεντρώνουμε στον *πίνακα κόστους(cost matrix)* όπως φαίνεται στον πίνακα 2.1.

	Actual negative	Actual positive
Predict negative	*C(0,0)*	*C(0,1)*
Predict positive	*C(1,0)*	*C(1,2)*

Πίνακας 2.1
Cost matrix για το πρόβλημα της δυαδικής ταξινόμησης

Ένα βασικό ζήτημα είναι ο καθορισμός των τιμών του κάθε κόστους, γιατί για διαφορετικές τιμές του κόστους, ο cost-sensitive αλγόριθμος ταξινόμησης θα δίνει διαφορετικά αποτελέσματα. Γι' αυτό τις περισσότερες φορές οι τιμές αυτές δίνονται από τον ειδικό του εκάστοτε τομέα.

Έχοντας γνώση του πίνακα κόστους, κατατάσσουμε το κάθε στιγμιότυπο στην κλάση η οποία έχει το *μικρότερο αναμενόμενο κόστος (minimum expected cost)*. Ουσιαστικά το αναμενόμενο κόστος *R(i|x)* είναι η ποσότητα που μας δείχνει το ρίσκο που εγκυμονεί η κατάταξη του στιγμιότυπου *x* στην κλάση *i.* Η τιμή *R(i|x)* αποκαλείται και ως *υπό-συνθήκη ρίσκο (conditional risk)* και υπολογίζεται από την σχέση:

$$R(i \mid x) = \sum_{j} P(j \mid x) C(i, j)$$

Η ποσότητα *P(j|x)* είναι η υπό-συνθήκη πιθανότητα του *j* δεδομένου του χ. Δηλαδή είναι η πιθανότητα το στιγμιότυπο *x* να ανήκει στην κλάση *j* δεδομένου ότι γνωρίζουμε το διάνυσμα χαρακτηριστικών του *x*.

Οπότε, στο παράδειγμά μας ο αλγόριθμος θα κατατάξει το στιγμιότυπο *x* στην κλάση *Positive* αν και μόνο αν:

$P(0|x)C(1,0) + P(1|x)C(1,1) \leq P(0|x)C(0,0) + P(1|x)C(0,1)$

Το οποίο ισοδυναμεί με την ανίσωση:

$P(0|x)(C(1,0)- C(0,0)) \leq P(1|x)(C(0,1)-C(1,1))$ *(1)*

Από την σχέση μπορούμε να καταλάβουμε, ότι η απόφαση του αλγορίθμου να κατατάξει το στιγμιότυπο *x* στην κλάση *Positive* δεν θα επηρεαστεί εάν προσθέσουμε κάποια σταθερά σε μια στήλη του αρχικού πίνακα κόστους. Επομένως, μπορούμε να απλοποιήσουμε τον πίνακα 2.1 αφαιρώντας τη σταθερά *C(0,0)* από την πρώτη στήλη και την *C(1,1)* από τη δεύτερη. Ο νέος πίνακας κόστους έχει την μορφή:

	Actual negative	Actual positive
Predict negative	*0*	*C(0,1) - C(1,1)*
Predict positive	*C(1,0) - C(0,0)*	*0*

Πίνακας 2.2

Απλοποιημένος πίνακας κόστους

Άρα, για να κατατάξει ο αλγόριθμος το στιγμιότυπο *x* στην κλάση *Positive*, θα πρέπει πλέον να ισχύει η σχέση:

$P(0|x)C(1,0) \leq P(1|x)C(0,1)$

Από τη θεωρία πιθανοτήτων ξέρουμε ότι ισχύει : $P(0|x) = 1 - P(1|x)$, οπότε η παραπάνω ανισότητα είναι ισοδύναμη με:

$$P(1 \mid x) \geq \frac{C(1,0)}{C(1,0) + C(0,1)} \qquad (2)$$

Άρα έχουμε το κατώφλι $p^* = \frac{C(1,0)}{C(1,0) + C(0,1)}$, τέτοιο ώστε ο αλγόριθμος να επιλέξει την κλάση 1 (δηλαδή *Positive)* για το στιγμιότυπο *x* εάν ισχύει η συνθήκη :

$P(1|x) \geq p^*$ *(3)*

Ένας οποιοσδήποτε cost-insensitive αλγόριθμος ταξινόμησης, ο οποίος είναι σε θέση να υπολογίζει την υπό-συνθήκη πιθανότητα *P(1|x)* για τα στιγμιότυπα *x* του συνόλου ελέγχου (π.χ. *Naïve Bayes*) , μπορεί να μετατραπεί σε cost-sensitive αν χρησιμοποιήσει την σχέση *(3)* κατά τη διαδικασία της ταξινόμησης. Αρκετοί meta-learning αλγόριθμοι εφαρμόζουν αυτή τη μέθοδο.

Όπως αναφέρθηκε και πριν, για να εφαρμοστεί η προηγούμενη μέθοδος, ο cost-insensitive αλγόριθμος ταξινόμησης θα πρέπει να μπορεί να υπολογίζει τις υπό-συνθήκη πιθανότητες που απαιτούνται. Παρόλα αυτά, αρκετοί αλγόριθμοι υπολογίζουν άλλα ποσοτικά μεγέθη προκειμένου να ταξινομούν τα στιγμιότυπα (π.χ.

C4.5). Επομένως, γίνεται αντιληπτό ότι η μέθοδος αυτή δεν είναι αρκετή για να μετατρέψουμε οποιοδήποτε cost-insensitive αλγόριθμο σε cost-sensitive.

Στο πρόβλημα της δυαδικής ταξινόμησης, οι cost-insensitive αλγόριθμοι προβλέπουν τις κλάσεις των στιγμιότυπων χρησιμοποιώντας ένα σταθερό κατώφλι t το οποίο ισούται με 0.5. Όμως στα πλαίσια της cost-sensitive μάθησης, δείξαμε προηγουμένως ότι θα θέλαμε ο αλγόριθμος να παίρνει τις αποφάσεις βάση της σχέσης *(3)* όπου το κατώφλι είναι p^*. Για να το πετύχουμε αυτό, μπορούμε να αλλάξουμε την ισορροπία του συνόλου εκπαίδευσης (δηλαδή την αναλογία των *Positive* και *Negative* στιγμιότυπων) εφαρμόζοντας δειγματοληψία. Στην γενική περίπτωση, προτιμούμαι να επιλέγουμε όλα τα στιγμιότυπα της σπάνιας κλάσης και να αλλάζουμε μόνο το πλήθος των στιγμιότυπων της συχνής κλάσης. Ο αριθμός των στιγμιότυπων της συχνής κλάσης που θα περάσει στο νέο σύνολο υπολογίζεται από την σχέση:

$$\#RareClassAfterSampling = \#RareClassBeforeSampling \cdot \frac{p^*}{1-p^*}\frac{1-p_0}{p_0} \quad (4)$$

Το p_0 είναι το κατώφλι που χρησιμοποιεί ο cost-insensitive αλγόριθμος. Στο παράδειγμα μας, το πρόβλημα ανάγεται σε πρόβλημά δυαδικής ταξινόμησης, οπότε ισχύει ότι : $p_0 = 0.5$ και $p^* = \frac{C(1,0)}{C(1,0)+C(0,1)}$ και άρα αριθμός των στιγμιότυπων της συχνής κλάσης θα πρέπει να πολλαπλασιαστεί με

$$\frac{p^*}{1-p^*} = \frac{C(1,0)}{C(0,1)}$$

Με αυτόν τρόπο, δημιουργείται ένα νέο σύνολο εκπαίδευσης S'. Έτσι, ένας cost-insensitive αλγόριθμος που χρησιμοποιεί ένα κατώφλι p_0 θα μπορεί να κατασκευάζει το μοντέλο ταξινόμησης βάση του συνόλου S' προκειμένου να εκτελεί cost–sensitive ταξινόμηση.

Σχεδόν όλοι οι meta-learning αλγόριθμοι εφαρμόζουν την τεχνική της δειγματοληψίας ή την προσέγγιση που βασίζεται στην σχέση *(3)*.

2.3.2 Cost-Sensitive Αλγόριθμοι

Στις προηγούμενες γραμμές, αναφερθήκαμε στην έννοια της cost-sensitive μάθησης. Περιγράψαμε το γενικό πλαίσιο λειτουργίας των cost-sensitive αλγορίθμων και είδαμε ότι μπορούμε να τους χωρίσουμε στις δυο γενικές κατηγορίες:

1. *Direct cost-sensitive αλγόριθμοι και*
2. *Cost-Sensitive meta-learning αλγόριθμοι*

Στα πλαίσια της παρούσας εργασία δεν θα ασχοληθούμε με τους αλγορίθμους της πρώτης κατηγορία. Θα εστιάσουμε την προσοχή μας στους meta-learners, για το λόγο ότι μας επιτρέπουν να χρησιμοποιήσουμε αλγορίθμους ταξινόμησης που υπάρχουν είδη, χωρίς να χρειαστεί να τους μεταβάλουμε.

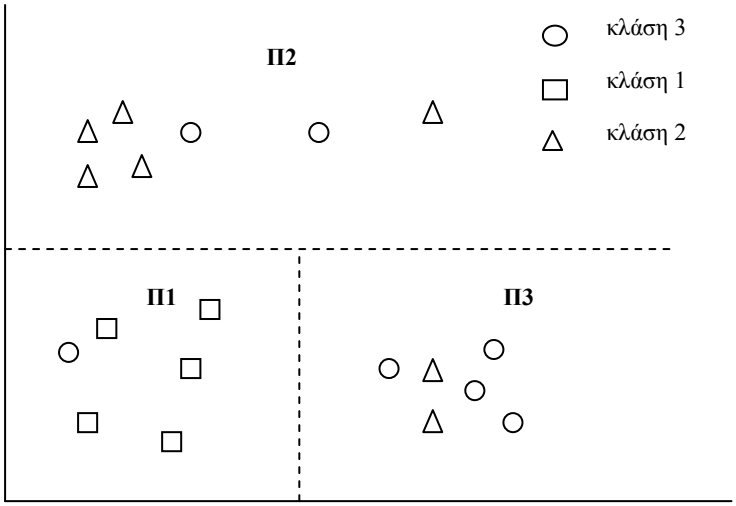
Στη συνέχεια θα περιγράψουμε meta-learning αλγορίθμους οι οποίοι εφαρμόζουν τις τεχνικές *relabeling και Sampling* τις οποίες περιγράψαμε πιο πριν.

2.3.2.1 MetaCost

Ο MetaCost[3] είναι ένας meta-learning αλγόριθμος ο οποίος βασίζεται στην τεχνική του *relabeling,* δηλαδή επανακαθορίζει τις κλάσεις των στιγμιότυπων του συνόλου εκπαίδευσης, σύμφωνα με το ελάχιστο αναμενόμενο κόστος (minimum expected cost). Το αναμενόμενο κόστος όπως είδαμε υπολογίζεται από την σχέση:

$$R(i \mid x) = \sum_j P(j \mid x) C(i, j)$$

Στην εικόνα 2.12 μπορούμε να δούμε σχηματικά την ερμηνεία της παραπάνω σχέσης.



Εικόνα 2.12

Η εικόνα αναπαριστά ένα πρόβλημα ταξινόμησης με τρείς κλάσεις. Ο χώρος X των στιγμιότυπων του συνόλου εκπαίδευσης διαμοιράστηκε σε τρείς περιοχές. Στην περιοχή i η κλάση i είναι η βέλτιστη, δηλαδή έχει το μικρότερο αναμενόμενο κόστος. Π.χ. στην Π1 βρίσκονται τα στιγμιότυπα στα οποία η σχέση $R(i \mid x)$ έδωσε ως *βέλτιστη κλάση* την *κλάση 1*. Παρατηρούμε ότι σε κάθε περιοχή υπάρχουν στιγμιότυπα στα οποία η βέλτιστη κλάση διαφέρει από την πραγματική.

Ο MetaCost βασίζεται στην ιδέα ότι σε κάθε περιοχή θα πρέπει να βρίσκονται στιγμιότυπα που ανήκουν στην ίδια κλάση. Οπότε μεταβάλλει την κλάση στην οποία πραγματικά ανήκουν, στην βέλτιστη σύμφωνα με το μικρότερο αναμενόμενο κόστος. Στην συνέχεια εφαρμόζεται ένας cost-insensitive αλγόριθμος ο οποίος σύμφωνα με την σχέση *(3)* που δείξαμε παραπάνω, ταξινομεί τα νέα στιγμιότυπα. Μάλιστα ο

MetaCost επιτρέπει να χρησιμοποιηθούν cost-insensitive αλγόριθμοι οι οποίοι δεν παράγουν υπό-συνθήκη πιθανότητες.

Ο MetaCost χρησιμοποιεί μια παραλλαγή της μεθόδου *bagging*: πρόκειται για τη διαδικασία η οποία δέχεται ένα σύνολο εκπαίδευσης μεγέθους *s* και εφαρμόζει *δειγματοληψία με επανατοποθέτηση* προκειμένου να παράγει ένα νέο σύνολο εκπαίδευσης μεγέθους *s* και αυτό. Έτσι μερικά στιγμιότυπα του αρχικού συνόλου ενδέχεται να εμφανίζονται μία, παραπάνω από μια, ή καμία φορά στο νέο σύνολο. Στην συνέχεια εφαρμόζεται κάποιος αλγόριθμος ταξινόμησης ο οποίος παράγει ένα μοντέλο. Η διαδικασία αυτή επαναλαμβάνεται *m* φορές. Στο τέλος, ο αλγόριθμος εφαρμόζει την διαδικασία της *ψηφοφορίας (voting)* στα *m* μοντέλα που κατασκευάστηκαν. Σύμφωνα με την ψηφοφορία, ένα στιγμιότυπο κατατάσσεται στην κλάση στην οποία καταλήγει η πλειοψηφία των *m* μοντέλων.

Όπως βλέπουμε στην εικόνα 2.13, ο MetaCost ξεκινάει εφαρμόζοντας τη μέθοδο *bagging*. Σε κάθε επανάληψη *i* δημιουργείται ένα νέο σύνολο S_i από το οποίο ο cost-insensitive αλγόριθμος L κατασκευάζει το μοντέλο ταξινόμησης M_i . Στη συνέχεια υπολογίζεται η εκτίμηση πιθανότητας της κάθε κλάσης για όλα τα στιγμιότυπα του αρχικού συνόλου εκπαίδευσης συμφωνά με το μοντέλο αυτό. Τέλος, βάση του ελάχιστου αναμενόμενου κόστους, επανακαθορίζονται οι κλάσεις των αντικειμένων.

Inputs:
- S is the training set,
- L is a classification learning algorithm,
- C is a cost matrix,
- m is the number of resamples to generate,
- n is the number of examples in each resample,
- p is *True* iff L produces class probabilities,
- q is *True* iff all resamples are to be used for each example.

Procedure MetaCost (S, L, C, m, n, p, q)

For $i = 1$ to m
- Let S_i be a resample of S with n examples.
- Let M_i = Model produced by applying L to S_i.

For each example x in S
- **For each** class j
  - Let $P(j|x) = \frac{1}{\sum_i 1} \sum_i P(j|x, M_i)$
  - **Where**
    - **If** p **then** $P(j|x, M_i)$ is produced by M_i
    - **Else** $P(j|x, M_i) = 1$ for the class predicted by M_i for x, and 0 for all others.
    - **If** q **then** i ranges over all M_i
    - **Else** i ranges over all M_i such that $x \notin S_i$.
- Let x's class = $\text{argmin}_i \sum_j P(j|x)C(i, j)$.

Let M = Model produced by applying L to S.

Return M.

Εικόνα 2.13
Αλγόριθμος MetaCost

Η διαδικασία αυτή επαναλαμβάνεται σε *m* βήματα. Το τελικό μοντέλο ταξινόμησης που βγαίνει στην έξοδο είναι αυτό που παράγεται από την εφαρμογή του cost-insensitive αλγορίθμου *L* στο ανανεωμένο (με επανακαθορισμένες κλάσεις στιγμιότυπων) πλέον σύνολο S.

Παρατηρούμε ότι ο MetaCost επιτρέπει την χρήση αλγορίθμων οι οποίοι δεν μπορούν να υπολογίσουν την πιθανότητα της κάθε κλάσης (παράμετρος *p*). Επίσης, δεν είναι υποχρεωτικό να λαμβάνονται υπ' όψιν όλα τα μοντέλα που σχηματίζονται (παράμετρος *q*). Τέλος ο MetaCost χρησιμοποιεί μια παραλλαγή της μεθόδου bagging κατά την οποία το νέο σύνολο που προκύπτει από τη δειγματοληψία, μπορεί να είναι μικρότερο σε μέγεθος από το αρχικό(παράμετρος *n*).

2.3.2.2 CSRoulette

Έχουμε χωρίσει τους cost-sensitive meta-learning αλγορίθμους σε τρείς κατηγορίες, ανάλογα με την τεχνική που υλοποιούν:

1. *Relabeling*
2. *Weighting*
3. *Sampling*

Στο προηγούμενο εδάφιο περιγράψαμε τον MetaCost ο οποίος εφαρμόζει την τεχνική relabeling, δηλαδή αλλάζει τις πραγματικές κλάσεις των στιγμιότυπων του συνόλου εκπαίδευση σύμφωνα με το μικρότερο αναμενόμενο κόστος. Στην παράγραφο αυτή, θα δώσουμε ένα παράδειγμα meta-learning αλγορίθμου που υλοποιεί την τεχνική Sampling. Πιο συγκεκριμένα θα αναφερθούμε στον αλγόριθμο *CSRoulette*[6].

Όντας ένας meta-learner, μετατρέπει έναν οποιοδήποτε cost-insensitive αλγόριθμο σε cost-sensitive χωρίς να τον μεταβάλει. Για να το πετύχει αυτό, αλλάζει την ισορροπία του συνόλου εκπαίδευσης εφαρμόζοντας δειγματοληψία. Φυσικά, η έννοια που «πρωταγωνιστεί» στη διαδικασία της δειγματοληψίας, είναι το *κόστος*. Για να γίνουμε πιο σαφείς, ο CSRoulette βασίζεται στην τεχνική *CPRS (cost proportionate roulette sampling)* την οποία περιγράφουμε στη συνέχεια.

Η τεχνική *CPRS* ουσιαστικά διαφοροποιεί τον CSRoulette από τον αλγόριθμο Costing ο οποίος βασίζεται στο *rejection sampling*[14]. Το μειονέκτημα αυτού του τρόπου δειγματοληψίας είναι ότι το μέγεθος του συνόλου που παράγεται, είναι αρκετά μικρότερο από το αρχικό, ως εκ τούτου έχουμε απώλεια σημαντικής πληροφορίας. Από την άλλη, στην τεχνική *CPRS* μπορούμε να καθορίσουμε εκ των προτέρων το μέγεθος του νέου συνόλου.

Ο όρος *CPRS* προέρχεται από τις λέξεις *Cost Proportionate Roulette Sampling*. Όπως προδίδει και το όνομα, έχουμε χρήση της μεθόδου *Roulette Sampling* η οποία αποτελεί ένα είδος στοχαστικής δειγματοληψίας με επανατοποθέτηση. Στην πράξη, τα στιγμιότυπα του συνόλου εκπαίδευσης αντιστοιχούνται σε τμήματα μιας ευθείας γραμμής (sampling line). Το μέγεθος του κάθε τμήματος ισούται με το βάρος του αντιστοίχου στιγμιότυπου. Στην περίπτωση της *CPRS*, το βάρος αυτό αντιστοιχεί στο κόστος λανθασμένης ταξινόμησης για το συγκεκριμένο στιγμιότυπο. Θα

προσπαθήσουμε να περιγράψουμε τον τρόπο λειτουργίας της *CPRS* μέσω ενός παραδείγματος.

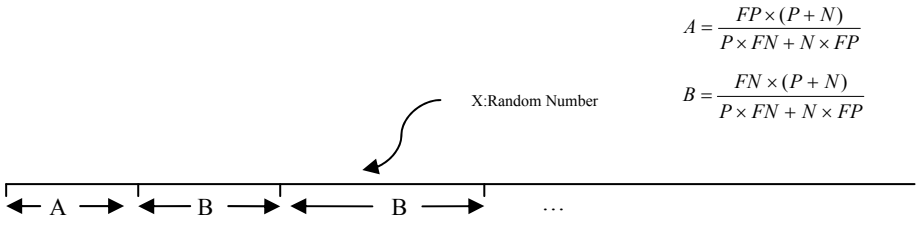
Έστω ότι έχουμε ένα πρόβλημα δυαδικής ταξινόμησης. Η μια κλάση είναι η *Positive (P)* και η άλλη *Negative (N)*. Στον πίνακα 2.2 παρουσιάζεται ο πίνακάς κόστους του προβλήματος. Αναθέτουμε σε κάθε στιγμιότυπο της κλάσης *Negative* ως βάρος την τιμή *FP* και αντίστοιχα κάθε στιγμιότυπο της κλάσης *Positive* την τιμή *FN*.

	Actual negative	Actual positive
Predict negative	*0*	*FN*
Predict positive	*FP*	*0*

Πίνακας 2.3

Στην *CPRS* κανονικοποιούμε τα βάρη των στιγμιότυπων, προκειμένου το άθροισμα τους να ισούται με το συνολικό πλήθος των στιγμιότυπων στο σύνολο εκπαίδευσης. Εάν, το σύνολο εκπαίδευσης απαρτίζεται από *P Positive* στιγμιότυπα και *N Negative,* τότε το βάρος των στιγμιότυπων των δυο κλάσεων είναι $\frac{FP \times (P+N)}{P \times FN + N \times FP}$ και $\frac{FN \times (P+N)}{P \times FN + N \times FP}$ αντίστοιχα. Τα βάρη αυτά καθορίζουν το μήκος του τμήματος της ευθείας δειγματοληψίας, που αντιστοιχεί σε κάθε στιγμιότυπο. Έτσι το άθροισμα των βαρών ισούται με *P+N*, οπότε και το μήκος της ευθείας γραμμής είναι ίσο με *P+N*.

Έχοντας χωρίσει την ευθεία γραμμή σε τμήματα, μπορούμε πλέον να εφαρμόσουμε τη δειγματοληψία. Κάθε φορά, ο αλγόριθμος παράγει έναν τυχαίο αριθμό ο οποίος κυμαίνεται στα πλαίσια του μήκους της γραμμής. Το στιγμιότυπο το οποίο αντιστοιχεί στο τμήμα της γραμμής στο οποίο «πέφτει» ο τυχαίος αριθμός, επιλέγεται στο νέο σύνολο (εικόνα 2.14). Η διαδικασία αυτή επαναλαμβάνεται έως ότου το νέο σύνολο να περιλαμβάνει *n* στιγμιότυπα. Η παράμετρος *n* δίνεται από τον χρήστη.



Εικόνα 2.14
Sampling line

Ο αλγόριθμος CSRoulette παρουσιάζεται στην εικόνα 2.15

Algorithm: *CSRoulette(T, B, C, I, x)*
Input: cost proportional roulette sampling (*CPRS*), training set *T*, based-classifier *B*, cost matrix *C*, a testing example *x*, and integer *I* (number of iterations of bootstrap).
For $r = 1$ to I do //build I classifiers
    S'=*CPRS* (T, C)
    Let $h_r = B(S')$
Output $h(x) = sign\left(\sum_{r=1}^{I} h_r(x)\right)$

Εικόνα 2.15
Αλγόριθμος CSRoulette

Παρατηρούμε ότι ο CSRoulette εφαρμόζει την μέθοδο bagging προκειμένου να βελτιώσει την απόδοση του. Πιο συγκεκριμένα, σε κάθε επανάληψη ο cost-insensitive αλγόριθμος *B* κατασκευάζει το μοντέλο ταξινόμησης h_r από το σύνολο εκπαίδευσης *S΄*. Το σύνολο *S΄* προκύπτει από την εφαρμογή της μεθόδου *CPRS*. Τέλος η ταξινόμηση των στιγμιότυπων του συνόλου γίνεται μέσω του *voting*.

2.3.3 Αξιολόγηση Cost-Sensitive κατηγοροιοποιητών

Η αποτίμηση της ακρίβειας ενός αλγορίθμου ταξινόμησης αποτελεί μια σημαντική διαδικασία, γιατί μας παρέχει την πληροφορία για την επίδοση που έχει ο εκάστοτε αλγόριθμος ταξινόμησης. Γνωρίζοντας την απόδοση ενός συνόλου αλγορίθμων, μπορούμε να επιλέξαμε τον πιο ιδανικό για την επίλυση του προβλήματος που αντιμετωπίζουμε.

Στην περίπτωση της cost-insensitive ταξινόμησης δείξαμε στην υποκεφάλαιο 2.1.31 ορισμένες μεθόδους αποτίμησης της ακρίβειας των αλγορίθμων. Τι γίνεται όμως στην περίπτωση που έχουμε cost-sensitive πρόβλημα ταξινόμησης; Μας αρκεί η πληροφορία που δίνουν οι μέθοδοι αυτοί; Πως θα αποφανθούμε για το ποίος αλγόριθμος είναι ο καλύτερος σε κάθε περίπτωση;

Στο παρόν εδάφιο θα προσπαθήσουμε να δώσουμε απαντήσεις στα προηγούμενα ερωτήματα. Θα δείξουμε πως γίνεται να αποτυπώσουμε με γραφικό τρόπο την απόδοση των cost-insensitive αλγορίθμων. Η γραφική αναπαράσταση της απόδοσης, δίνει την δυνατότητα σε έναν ειδικό να συγκρίνει δυο αλγορίθμους μεταξύ τους, ώστε να δει πόσο καλύτερα αποδίδει ο ένας έναντι του άλλου.

Θα παρουσιάσουμε δυο τρόπους γραφικής αναπαράστασης της απόδοσης:

1. *Καμπύλες ROC(ROC curves)*
2. *Καμπύλες κόστους(cost curves)*

Σε ένα πρόβλημα δυαδικής ταξινόμησης, ο χώρος *ROC* είναι μια γραφική παράσταση στο χώρο των δύο διαστάσεων. Στον άξονα *y* έχουμε τις τιμές *true positive rate* και στον άξονα των *x* έχουμε τις τιμές *false positive rate.*

$$true\ positive\ rate = \frac{\#\ predicted\ \ positive\ \text{instances}}{\#\,\text{actual positive instances}}$$

$$false\ positive\ rate = \frac{\#\ predicted\ \ positive\ \text{instances}}{\#\,\text{actual negative instances}}$$

Γνωρίζοντας τον *πίνακα σύγχυσης(confusion matrix)* που δίνει ο αλγόριθμος, μπορούμε να βρούμε το σημείο στο χώρο *ROC* που αντιστοιχεί σε αυτόν. Το σημείο αυτό ονομάζεται *σημείο ROC (ROC point)*. Αν ενώσουμε όλα τα σημεία *ROC* καθώς και τα σημεία (0,0), (1,1) , προκύπτει η *καμπύλη ROC*. Η μέθοδος με την οποία παράγεται μια ακολουθία σημείων *ROC* για έναν αλγόριθμο, εξαρτάται αποκλειστικά από τον ίδιο τον αλγόριθμο. Π.χ. στο *Naïve Bayes* τα σημεία παράγονται μεταβάλλοντας σε κάθε εκτέλεση την τιμή κατωφλίου που δέχεται ως παράμετρο ο αλγόριθμος. Για διαφορετικές τιμές κατωφλίου ο αλγόριθμος βγάζει διαφορετικά αποτελέσματα, οπότε και διαφορετικούς πίνακες σύγχυσης.

	Actual negative	Actual positive
Predict negative	10	22
Predict positive	16	12

Πίνακας 2.4
Παράδειγμα ενός confusion matrix

Στο χώρο *ROC*, ένα σημείο κυριαρχεί ενός άλλου εάν έχει μεγαλύτερη τιμή *true positive rate* και μικρότερη τη *false positive rate.* Όταν ένα σημείο *A* κυριαρχεί ενός σημείου *B* τότε το *A* έχει μικρότερο αναμενόμενο κόστος από το σημείο *B*. Πιο συγκεκριμένα, το μοντέλο που παράγει το confusion matrix που αντιστοιχεί στο σημείο *A* θα έχει μικρότερο αναμενόμενο κόστος από το αντίστοιχο του σημείου *B*.

Από την άλλη, ο χώρος του κόστους είναι μια γραφική παράσταση η οποία στο άξονα *y* έχει το αναμενόμενο κόστος κανονικοποιημένο ώστε να είναι μεταξύ του 0 και του 1. Στον άξονα *x* είναι τα «λειτουργικά σημεία» (συνδυασμός του κόστους και της κατανομής μιας κλάσης) κανονικοποιημένα συμφωνά με τη σχέση *(1)* ώστε να είναι μεταξύ του 0 και του 1.

$$PC(+) = \frac{p(+)C(-\,|\,+)}{p(+)C(-\,|\,+) + p(-)C(+\,|\,-)} \qquad (1)$$

Όπου:

C(-|+): κόστος από την ταξινόμηση ενός *positive* αντικειμένου ως *negative*

C(+|-): κόστος από την ταξινόμηση ενός *negative* αντικειμένου ως *positive*

p(+): πιθανότητα της κλάσης *positive*

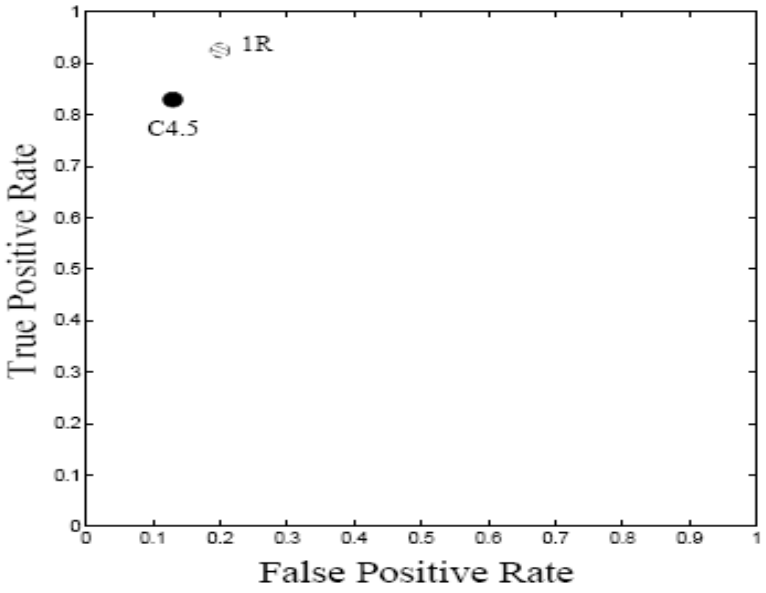
p(-)=1-p(+): πιθανότητα της κλάσης *negative*

Υπάρχει μια σχέση που συνδέει τους δυο χώρους. Συγκεκριμένα, ένα σημείο στο χώρο *ROC* αντιπροσωπεύεται από μια ευθεία γραμμή στο χώρο του κόστους, καθώς και μία ευθεία στο χώρο *ROC* αντιστοιχεί σε ένα σημείο στο χώρο του κόστους. Ένα μοντέλο ταξινόμησης που αντιστοιχεί στο σημείο *(FP,TP)* στο χώρο *ROC* είναι μια ευθεία στο χώρο του κόστους για την οποία ισχύει:

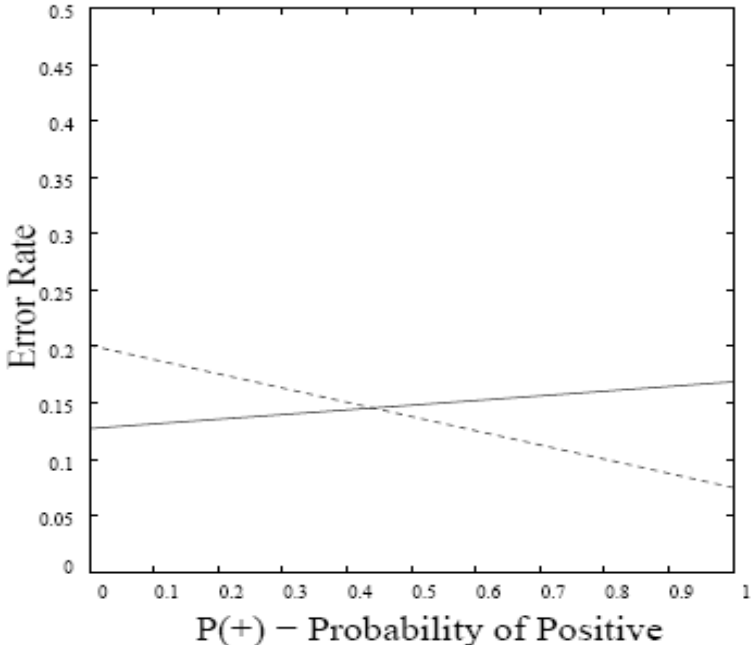
1. $y = FP$ όταν $x=0$
2. $y = 1\text{-}TP$ όταν $\chi=1$

Στην εικόνα 2.17 βλέπουμε δυο *ROC* καμπύλες που αντιστοιχούν σε δυο αλγορίθμους. Η ευθεία γραμμή είναι η εφαπτομένη στις δυο καμπύλες. Η κλίση της αντιπροσωπεύει το «λειτουργικό» σημείο στο οποίο οι δυο αλγόριθμοι ταξινόμησης αποδίδουν το ίδιο. Στα «λειτουργικά» σημεία που αντιστοιχούν σε μεγάλη κλίση της ευθείας, ο αλγόριθμος με την dotted *ROC* καμπύλη έχει καλύτερη απόδοση από τον άλλο. Το αντίστροφο ισχύει στα «λειτουργικά» σημεία που αντιστοιχούν σε μικρή κλίση της ευθείας.

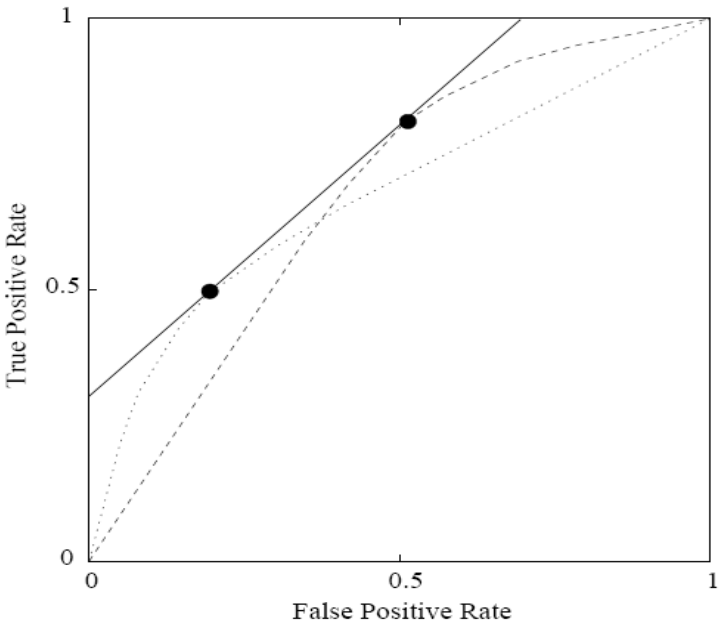
Η εικόνα 2.19 δείχνει τις καμπύλες κόστους που αντιστοιχούν στις καμπύλες *ROC* της εικόνας 2.18. παρατηρούμε ότι η dotted καμπύλη κόστους έχει μικρότερο αναμενόμενο κόστος και άρα υπερισχύει της καμπύλης κόστους 2, όταν *PC<0.5*.



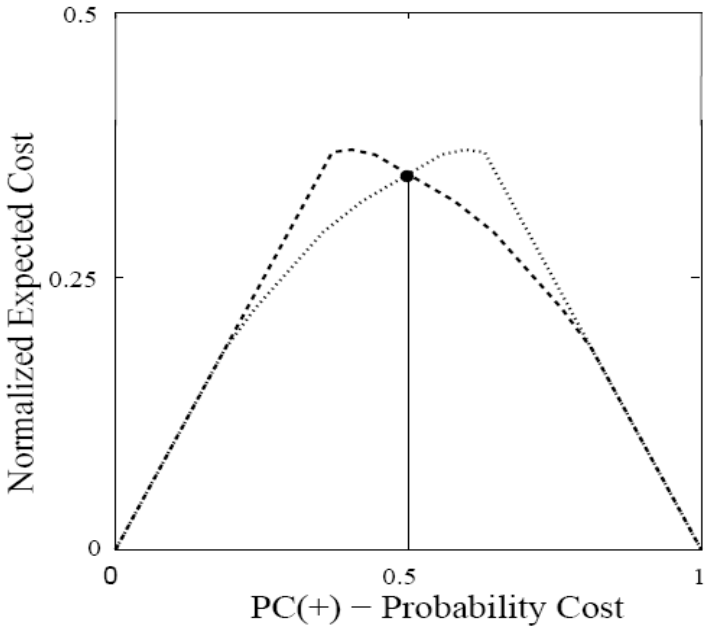
Εικόνα 2.16
Δυο σημεία *ROC*



Εικόνα 2.17
Ευθείες που αντιστοιχούν στα σημεία της 2.16



Εικόνα 2.18
Καμπύλες *ROC* δυο αλγορίθμων ταξινόμησης



Εικόνα 2.19
Καμπύλες κόστους που αντιπροσωπεύουν τις καμπύλες *ROC* της 2.18

ΚΕΦΑΛΑΙΟ 3: Η ΕΦΑΡΜΟΓΗ

Η ΕΦΑΡΜΟΓΗ

ΕΙΣΑΓΩΓΗ

Στο προηγούμενο κεφάλαιο περιγράψαμε τις βασικές έννοιες της μηχανικής μάθησης και πιο συγκεκριμένα αναφερθήκαμε στους αλγόριθμους ταξινόμησης. Αναλύσαμε συνοπτικά συγκεκριμένους αλγορίθμους ταξινόμησης οι οποίοι θα χρησιμοποιηθούν και στην εφαρμογή μας και είδαμε τους διαφορετικούς τρόπους με τους οποίους προσεγγίζουν την επίλυση προβλημάτων ταξινόμησης. Τέλος, περιγράψαμε μια σχετικά καινούργια ομάδα αλγορίθμων – τους *cost-sensitive αλγορίθμους μάθησης*- που αποτελούν το βασικό χαρακτηριστικό της εφαρμογής μας.

Στο κεφάλαιο αυτό, θα περιγράψουμε αναλυτικά το πρόβλημα που καλούμαστε να επιλύσουμε. Θα αναλύσουμε τον τρόπο με τον οποίο, θα χρησιμοποιηθούν στην πράξη οι αλγόριθμοι που περιγράψαμε στη προηγούμενη παράγραφο. Τέλος θα εστιάσουμε την προσοχή μας στην εφαρμογή που υλοποιήθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας.

3.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη ενός συστήματος το οποίο θα υλοποιεί στατιστική ανάλυση και πρόβλεψη αγώνων ποδοσφαίρου. Το πρόβλημα αυτό αναλύεται στα εξής τρία υποπροβλήματα:

1. Συλλογή πληροφοριών σχετικά με τα αποτελέσματα (καθώς επίσης και επιπρόσθετα στοιχεία όπως πρωτάθλημα, ομάδες, σκορ ημιχρόνου κ.α.) αγώνων ποδοσφαίρου από όλο τον κόσμο μαζί με τις αποδόσεις στοιχηματισμού στους αγώνες αυτούς. Η εφαρμογή συλλέγει τα δεδομένα αυτά αυτόματα από αρχεία XML, τα οποία παρέχονται από την επίσημη ιστοσελίδα του ΟΠΑΠ, και τα αποθηκεύει σε μια βάση δεδομένων.

2. Υλοποίηση ενός Graphical User Interface (GUI), το οποίο αποτελεί τη διασύνδεση του χρήστη της εφαρμογής με το σύστημα. Ένα από τα ζητούμενα είναι η παρουσίαση στατιστικών στοιχείων σχετικά με τους αγώνες που είναι αποθηκευμένοι στη βάση δεδομένων.

3. Το πιο ενδιαφέρον κομμάτι της εργασίας είναι η ανάπτυξη ενός υποσυστήματος-πράκτορα, ο οποίος θα παράγει συστάσεις στοιχημάτων προς τους χρήστες, με στόχο τη μεγιστοποίηση των κερδών τους.

Όπως γίνεται αντιληπτό, ότι έχουμε αναφέρει μέχρι στιγμής, αποτελεί τη βάση για την ανάπτυξη του υποσυστήματος-πράκτορα. Πιο συγκεκριμένα, θα γίνει χρήση ενός *cost-sensitive* αλγορίθμου ταξινόμησης, ο οποίος θα παράγει τον κατηγοριοποιητή που θα προβλέπει τα παιχνίδια που βρίσκονται στο πιο πρόσφατο κουπόνι του ΟΠΑΠ - το οποίο θα λαμβάνεται αυτόματα από την επίσημη ιστοσελίδα του ΟΠΑΠ.

Στο σημείο αυτό θα πρέπει να αναφέρουμε, ότι η επιλογή του *cost-sensitive* αλγορίθμου δεν ήταν εύκολη. Μετά από αρκετά πειράματα (τα αποτελέσματα παρουσιάζονται στο παράρτημα Ι) στα οποία εξεταστικέ κυρίως η απόδοση των αλγορίθμων ως προς το κόστος, καταλήξαμε στον αλγόριθμο *MetaCost* (βλέπε κεφάλαιο 2.3.2.1) ο οποίος χρησιμοποιεί ως base classifier τον αλγόριθμο Logistic.

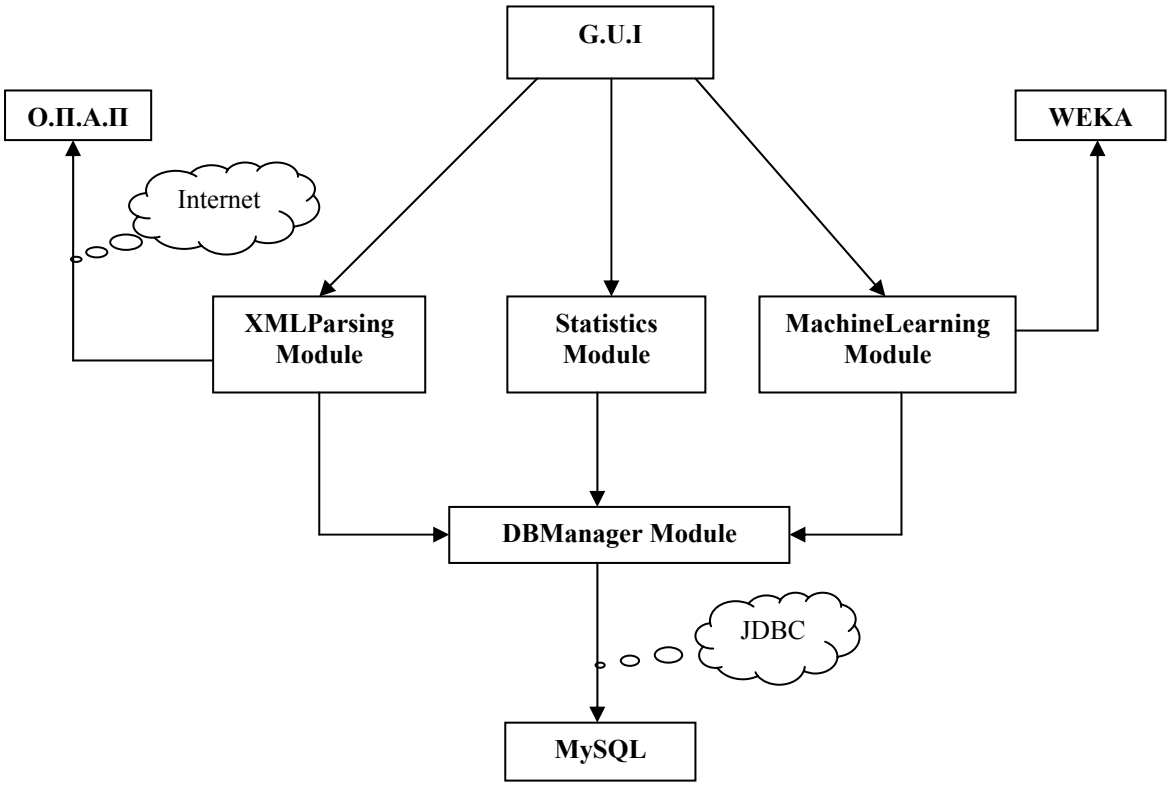
Ένα εύλογο ερώτημα που τίθεται είναι, γιατί να χρησιμοποιηθεί ένας *cost-sensitive* αλγόριθμος ταξινόμησης; Η απάντηση είναι απλή, σκοπός της εφαρμογής είναι να προβλέπει αγώνες ποδοσφαίρου, με στόχο **τη μεγιστοποίηση των κερδών του χρήστη!** Δηλαδή, δεν θέλουμε έναν αλγόριθμό ο οποίος απλώς να προβλέπει τους αγώνες ποδοσφαίρου – ίσως και με μεγάλο ποσοστό επιτυχίας- αλλά θέλουμε έναν αλγόριθμο που θα προβλέπει επιτυχώς τα σημεία που έχουν μεγάλες αποδόσεις.

Στο σημείο αυτό θα ήταν ίσως σκόπιμο να περιγράψουμε σύντομα τον γενικό τρόπο με τον οποίο πραγματοποιείται ο στοιχηματισμός των αγώνων ποδοσφαίρου. Τις ημέρες που υπάρχουν αγώνες ποδοσφαίρου σε όλο τον κόσμο, ο ΟΠΑΠ εκδίδει ένα κουπόνι στο οποίο συγκεντρώνει ένα υποσύνολο των αγώνων που πραγματοποιούνται. Στο κουπόνι αυτό, για κάθε αγώνα καταγράφονται οι αποδόσεις που έχουν τα τρία πιθανά σημεία ενός αγώνα (1-νίκη γηπεδούχο ομάδας, Χ-ισσοπάλια και 2-νίκη φιλοξενούμενης ομάδας). Ανάλογα με τη δυναμική της κάθε ομάδας αλλά και άλλων χαρακτηριστικών (π.χ. κατάσταση παιχτών, σημαντικότητα του αγώνα για κάποια ομάδα κ.α.), οι αποδόσεις ποικίλουν. Συνήθως, οι ομάδες που θεωρούνται επικρατέστερες για τη νίκη σε έναν αγώνα, έχουν πολύ μικρή απόδοση. Στη γενική περίπτωση, ένα παίχτης μπορεί να ποντάρει όσα παιχνίδια θέλει και να καταβάλει ένα χρηματικό ποσό. Για να κερδίσει, θα πρέπει να προβλέψει σωστά όλα τα παιχνίδια που πόνταρε αλλιώς χάνει τα χρήματα που κατέβαλε. Στην περίπτωση που κερδίσει, παίρνει τα χρήματα που έδωσε πολλαπλασιαζόμενα με το γινόμενο των αποδόσεων που πόνταρε.

Στόχος της εφαρμογής είναι να εκτελεί την παραπάνω διαδικασία αυτόματα για τον χρήστη, χωρίς να απαιτεί την παρέμβαση του. Πιο συγκεκριμένα, το κέρδος από τη σωστή πρόβλεψη διαφέρει από περίπτωση σε περίπτωση και εξαρτάται από την απόδοση του αποτελέσματος καθώς και το ποντάρισμα, το οποίο θεωρούμε ότι είναι σταθερό. Επόμενος η εφαρμογή καλείται να προβλέψει τις αποδόσεις οι οποίες θα μεγιστοποιήσουν το κέρδος του χρήστη ή αντίστοιχα θα ελαχιστοποιήσουν το κόστος του χρήστη. Οπότε καθίσταται επιτακτική η υλοποίηση *cost-sensitive* αλγορίθμων ταξινόμησης.

3.1.1 Αρχιτεκτονική συστήματος

Η αρχιτεκτονική του συστήματος παρουσιάζεται στην εικόνα 3.1.



Εικόνα 3.1
Αρχιτεκτονική Συστήματος

Όπως παρατηρούμε, στην κορυφή βρίσκεται το G.U.I που αποτελεί το μοναδικό τρόπο διασύνδεσης του χρήστη με το σύστημα. Τα υπόλοιπα τμήματα του συστήματος θα αναλυθούν περεταίρω στη συνέχεια του κεφαλαίου αυτού.

3.2 ΠΡΟΣΕΓΓΙΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Προτού συνεχίσουμε με την περιγραφή και την ανάλυση της εφαρμογής, θα αναφέρουμε ορισμένα στοιχεία τα οποία έπρεπε να λάβουμε υπόψη προτού ξεκινήσουμε την ανάπτυξη της εφαρμογής.

3.2.1 Επιλογή εργαλείων

Μια σημαντική διαδικασία για την υλοποίηση της εφαρμογής, ήταν η επιλογή των απαραίτητων εργαλείων. Τα εργαλεία που επιλέχθηκαν να χρησιμοποιηθούν είναι τα εξής:

1. Ένα σύστημα διαχείρισης βάσης δεδομένων (ΣΔΒΔ) το οποίο επιτρέπει τη δημιουργία μια βάσης δεδομένων που είναι απαραίτητη για την αποθήκευση των απαραίτητων πληροφοριών που περιγράψαμε παραπάνω. Χρησιμοποιήθηκε η *MySQL* που είναι ένα ΣΔΒΔ ανοιχτού κώδικα το οποίο παρέχεται δωρεάν και είναι γνωστό για τη σταθερότητα και τη ταχύτητα που προσφέρει. Ο λόγος επιλογής του συγκεκριμένου ΣΔΒΔ είναι προφανής.

2. Για τη σύνδεση της εφαρμογής με τη MySQL χρησιμοποιήθηκε το interface *JDBC* το οποίο προσφέρεται από το *jdk* της *Java*, και το Driver *mysql-connector-java* το οποίο υλοποίει τη διασύνδεση JDBC.

3. Για το parsing των XML αρχείων που περιλαμβάνουν τα δεδομένα για τους αγώνες ποδοσφαίρου, χρησιμοποιήθηκε ο *jdom* xml parser. Ο jdom εκτελεί το λεγόμενο *Document Object Model Parsing*. Ουσιαστικά αναπαριστά το XML αρχείο ως έναν γράφο, στον οποίο πραγματοποιεί διάσχιση προκειμένου να διαβάσει όλα τα elements του XML αρχείου.

4. Το εργαλείο *WEKA* το οποίο συγκεντρώνει ένα μεγάλο αριθμό αλγορίθμων μηχανικής μάθησης όπως αλγόριθμοι ταξινόμησης, αλγόριθμοι παραγωγής συσχετίσεων καθώς και αλγόριθμοι που σχηματίζουν συστάδες στα δεδομένα. Στα πλαίσια της παρούσας εργασίας, εμπλουτίσαμε το εργαλείο WEKA, αναπτύσσοντας τον αλγόριθμο *CSRoulette* που περιγράψαμε στο προηγούμενο κεφάλαιο. Επίσης, τροποποιήσαμε ελαφρώς τον αλγόριθμο *MetaCost* ώστε να μπορεί να επιλύει προβλήματα ταξινόμησης, στα οποία υπάρχει διαφορετικό κόστος ανά στιγμιότυπο (στη βασική έκδοση του WEKA, δεν παρέχεται η δυνατότητα αυτή).

5. Η συγγραφή του κώδικα έγινε στο *NetBeans* που είναι ένα IDE (integrated development environment) το οποίο παρέχεται δωρεάν από την εταιρία SUN. Για την υλοποίηση του GUI χρησιμοποιήθηκε το εργαλείο Matisse, το οποίο είναι ενσωματωμένο στο NetBeans και παρέχει έναν γραφικό τρόπο ανάπτυξης GUI. Επίσης, χρησιμοποιήθηκαν οι βιβλιοθήκες, Swing, OpenSwing, Liquidlnf, SwingX και miglayout.

6. Για την απομακρυσμένη σύνδεση της εφαρμογής με την ιστοσελίδα του ΟΠΑΠ, χρησιμοποιήσαμε το πακέτο java.net. Το java.net περιλαμβάνει κλάσεις που επιτρέπουν τη σύνδεση σε απομακρυσμένους πόρους, μέσω ενός δικτύου (στην περίπτωση μας, μέσω διαδικτύου).

7. Τέλος, για τους σκοπούς της αντικειμενοστραφής ανάλυσης, χρησιμοποιήθηκε το εργαλείο *Visual Paradigm*. Με το εργαλείο αυτό, προδιαγράψαμε τις περιπτώσεις χρήσης του συστήματος και αναπτύξαμε διαγράμματα κλάσεων, ακολουθίας και συνεργασίας (Μερικά από αυτά παρουσιάζονται στο παράρτημα).

3.2.2 Επιλογή χαρακτηριστικών

Έχοντας επιλέξει πλέον τα κατάλληλα εργαλεία για την ανάπτυξη της εφαρμογής, έπρεπε στη συνέχεια να καθορίσουμε από πια χαρακτηριστικά θα απαρτίζονται τα στιγμιότυπα του συνόλου εκπαίδευσης. Όπως αντιλαμβάνεστε, η διαδικασία αυτή δεν ήταν απλή. Χρειάστηκε να εκτελέσουμε αρκετά πειράματα, με διαφορετικά σύνολα εκπαίδευσης μέχρι να εντοπίσουμε τα χαρακτηριστικά που σχηματίζουν τα στιγμιότυπα, βάση των οποίων ο αλγόριθμος κατασκευάζει το πιο αποδοτικό (ως προς το κέρδος) μοντέλο ταξινόμησης. Τα χαρακτηριστικά που επιλέχθηκαν είναι:

1. HomeTeamWinRatio: ο λόγος των νικών της γηπεδούχο ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε στην έδρα της.

2. HomeTeamTieRatio: ο λόγος των ισοπαλιών της γηπεδούχο ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε στην έδρα της.

3. HomeTeamLoseRatio: ο λόγος των ηττών της γηπεδούχο ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε στην έδρα της.

4. HomeTeamMeanApodosi1: ο μέσος όρος της απόδοσης του σημείου 1, στα τελευταία 3 παιχνίδια της γηπεδούχο ομάδας που έγιναν στην έδρα της.

5. HomeTeamMeanApodosiX: ο μέσος όρος της απόδοσης του σημείου X, στα τελευταία 3 παιχνίδια της γηπεδούχο ομάδας που έγιναν στην έδρα της.

6. AwayTeamWinRatio: ο λόγος των νικών της φιλοξενούμενης ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε εκτός έδρας.

7. AwayTeamTieRatio: ο λόγος των ισοπαλιών της φιλοξενούμενης ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε εκτός έδρας.

8. AwayTeamLoseRatio: ο λόγος των ηττών της φιλοξενούμενης ομάδας, στα τελευταία 3 παιχνίδια, στα οποία έπαιξε εκτός έδρας.

9. AwayTeamMeanApodosi2: ο μέσος όρος της απόδοσης του σημείου 2, στα τελευταία 3 παιχνίδια της φιλοξενούμενης ομάδας, στα οποία έπαιξε εκτός έδρας.

10. AwayTeamMeanApodosiX: ο μέσος όρος της απόδοσης του σημείου X, στα τελευταία 3 παιχνίδια της φιλοξενούμενης ομάδας, στα οποία έπαιξε εκτός έδρας.

11. Apodosi1: η απόδοση του σημείου 1, στο συγκεκριμένο παιχνίδι.

12. ApodosiX: η απόδοση του σημείου X.

13. Apodosi2: η απόδοση του σημείου 2.

14. MatchResult: nominal χαρακτηριστικό, που δηλώνει το αποτέλεσμα μεταξύ των 2 ομάδων.

Στο σημείο αυτό θα πρέπει να διευκρινιστεί ότι τα παραπάνω χαρακτηριστικά δεν είναι τα βέλτιστα δυνατά. Η βασική ιδέα που κρύβεται πίσω από την επιλογή αυτή, είναι η φόρμα της κάθε ομάδας στα τελευταία 3 παιχνίδια, σε συνδυασμό με τις αποδόσεις που δίνονται από τον ΟΠΑΠ. Τα χαρακτηριστικά αυτά, αποτελούν τον τρόπο με τον οποίο σκέφτεται ένας άνθρωπός προτού στοιχηματίσει έναν αγώνα.

Φυσικά, υπάρχουν και άλλα στοιχεία τα οποία επηρεάζουν την έκβαση ενός αγώνα ποδοσφαίρου, όπως η κατάταξη των δυο ομάδων στον πίνακα βαθμολογίας του πρωταθλήματος, το ενδιαφέρον που έχει μια ομάδα από έναν συγκεκριμένο αγώνα, οι απουσίες παιχτών, το ενδιαφέρον μια ομάδας για την διοργάνωση στα πλαίσια της οποίας πραγματοποιείται ο αγώνας (π.χ. πολλές από τις ομάδες που διεκδικούν το πρωτάθλημα, δεν δίνουν μεγάλη βαρύτητα στα παιχνίδια του κυπέλου), και διάφορες άλλες πληροφορίες σχετικά με την εσωτερική κατάσταση της ομάδας. Για παράδειγμα, όταν μια ομάδα αλλάζει προπονητή κατά τη διάρκεια της χρονιάς, συνήθως το πρώτο παιχνίδι έχει θετικό αποτέλεσμα για την αυτήν (είτε νίκη ή ισοπαλία).

Τα χαρακτηριστικά που επιλέχτηκαν τελικά σε συνδυασμό με τα επιθυμητά χαρακτηριστικά που περιγράψαμε παραπάνω, πιθανόν να οδηγούν στην κατασκευή ακόμη πιο αποδοτικού μοντέλου ταξινόμησης. Το ερώτημα που τίθεται όμως είναι, γιατί δεν συμπεριελήφθησαν τα χαρακτηριστικά αυτά στο σύνολο εκπαίδευσης; Όπως προείπαμε παραπάνω, στόχος της εργασίας είναι η ανάπτυξη μίας αυτοματοποιημένης εφαρμογής, η οποία θα έχει την ικανότητα να συλλέγει πληροφορίες και να προβλέπει αγώνες ποδοσφαίρου, χωρίς την παρέμβαση του χρήστη, δηλαδή χωρίς να χρειαστεί ο χρήστης να εκτελέσει κάποια επιπρόσθετη ενέργεια. Οπότε χρειάζεται κάποια πηγή η οποία θα προσφέρει τις πληροφορίες αυτές σε μορφή ημι-δομημένων δεδομένων (π.χ. XML), που θα είναι κατανοητή από ένα υπολογιστικό πρόγραμμα ώστε να μπορεί να γίνεται αυτόματα η συλλογή. Κατά την εκπόνηση της παρούσας εργασίας, αναζητήθηκαν τέτοιες πηγές πληροφορίας, αλλά χωρίς ιδιαίτερα αποτελέσματα. Επιπλέον, δεν θέλουμε να αναγκάσουμε τον χρήστη να εισάγει όλα τα παραπάνω δεδομένα στην εφαρμογή, γιατί είναι μια διεργασία που έχει μεγάλο χρονικό κόστος για αυτόν και επιπλέον δεν είναι διατεθειμένοι όλοι να την εκτελέσουν. Έτσι κρίθηκε ορθό να μην συμπεριληφθούν τα παραπάνω επιθυμητά χαρακτηριστικά στο σύνολο εκπαίδευσης.

3.3 ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Όπως αναφέραμε προηγουμένως, για την δημιουργία της βάσης δεδομένων χρησιμοποιήθηκε η MySQL. Στην συνέχεια παρουσιάζουμε τους πίνακες που αποτελούν τη βάση δεδομένων της εφαρμογής.

3.3.1 Πίνακας CouponInfo

Ο πίνακας στον οποίο αποθηκεύεται το μεγαλύτερο μέρος των πληροφοριών που συλλέγονται αυτόματα από τα αρχεία XML που παρέχονται από την επίσημη ιστοσελίδα του ΟΠΑΠ. Οι πληροφορίες αυτές σχετίζονται με αποτελέσματα αγώνων, αποδόσεις σημείων κ.α. Κάθε πλειάδα συγκεντρώνει τις πληροφορίες για ένα συγκεκριμένο αγώνα ποδοσφαίρου. Παρακάτω περιγράφονται συνοπτικά τα πεδία του πίνακα.

1. EventCode: είναι τύπου int και αποτελεί το πρωτεύον κλειδί του πίνακα. Επίσης είναι auto increment, δηλαδή παίρνει αυτόματα αυξανόμενες τιμές.
2. Date: είναι τύπου date και αναφέρεται στην ημερομηνία διεξαγωγής του αγώνα ποδοσφαίρου.
3. League: είναι τύπου varchar και αναφέρεται στο πρωτάθλημα στα πλαίσια του οποίου πραγματοποιείται ο αγώνας.
4. Apodosi1: είναι τύπου float και αναφέρεται στην απόδοση του σημείου 1 του συγκεκριμένου αγώνα.
5. HomeTeam: είναι τύπου varchar και αναφέρεται στη γηπεδούχο ομάδα του συγκεκριμένου αγώνα.
6. ApodosiX: είναι τύπου float και αναφέρεται στην απόδοση του σημείου X του συγκεκριμένου αγώνα.
7. AwayTeam: είναι τύπου varchar και αναφέρεται στη φιλοξενούμενη ομάδα συγκεκριμένου του αγώνα.
8. Apodosi2: είναι τύπου float και αναφέρεται στην απόδοση του σημείου 2 του συγκεκριμένου αγώνα.
9. Apodosi1X: είναι τύπου float και αναφέρεται στην απόδοση του σημείου 1X του συγκεκριμένου αγώνα.
10. Apodosi12: είναι τύπου float και αναφέρεται στην απόδοση του σημείου 12 του συγκεκριμένου αγώνα.
11. ApodosiX2: είναι τύπου float και αναφέρεται στην απόδοση του σημείου X2 του συγκεκριμένου αγώνα.
12. ApodosiUnder: είναι τύπου float και αναφέρεται στην απόδοση του σημείου under (συνολικός αριθμός τερμάτων κάτω από 2.5) του συγκεκριμένου αγώνα.
13. ApodosiOver: είναι τύπου float και αναφέρεται στην απόδοση του σημείου over (συνολικός αριθμός τερμάτων πάνω από 2.5) του συγκεκριμένου αγώνα.

14. FirstHalf_HomeTeamScore: είναι τύπου int και αναφέρεται στο σύνολο των τερμάτων που πέτυχε η γηπεδούχος ομάδα στο πρώτο ημίχρονο του αγώνα.
15. FirstHalf_AwayTeamScore: είναι τύπου int και αναφέρεται στο σύνολο των τερμάτων που πέτυχε η φιλοξενούμενη ομάδα στο πρώτο ημίχρονο του αγώνα.
16. Final_HomeTeamScore: είναι τύπου int και αναφέρεται στο σύνολο των τερμάτων που πέτυχε η γηπεδούχος ομάδα συνολικά στον αγώνα.
17. Final_AwayTeamScore: είναι τύπου int και αναφέρεται στο σύνολο των τερμάτων που πέτυχε η φιλοξενούμενη ομάδα συνολικά στον αγώνα.
18. FS_Simio: είναι τύπου char και αναφέρεται στο νικητήριο σημείο του αγώνα.

EventCode	Date	League	Apodosi1	HomeTeam	ApodosiX	AwayTeam	Apodosi2	Apodosi1X	Apodosi12	ApodosiX2	ApodosiUNDER	ApodosiOVER	FirstHalf_HomeTeamS
6	2008-08-26	CHL	5.00	ΑΡΤΜΕΝΤΙΑ	3.60	ΠΟΥΒΕΝΤΟΥΣ	1.45	2.09	1.12	0.00	1.67	1.77	1
7	2008-08-26	CHL	6.25	ΒΙΣΛΑ ΚΡΑΚΟΒΙΑΣ	3.85	ΜΠΑΡΤΣΕΛΟΝΑ	1.35	2.38	1.11	0.00	1.85	1.60	0
8	2008-08-26	CHL	1.40	ΠΑΝΑΘΗΝΑΪΚΟΣ	3.60	ΣΠΑΡΤΑ ΠΡΑΓΑΣ	6.00	0.00	1.14	2.25	1.65	1.80	0
44	2008-08-27	CHL	2.25	ΜΠΑΤΕ ΜΠΟΡΙΣΟΦ	3.25	ΛΕΦΣΚΙ	2.40	1.33	1.16	1.38	1.57	1.90	1
46	2008-08-27	CHL	2.30	ΚΑΟΥΝΑΣ	3.30	ΑΑΛΜΠΟΡΓΚ	2.35	1.36	1.16	1.37	1.60	1.85	0
48	2008-08-27	CHL	1.75	ΝΤΙΝΑΜΟ ΚΙΕΒΟΥ	3.30	ΣΠΑΡΤΑΚ ΜΟΣΧΑΣ	3.40	1.14	1.16	1.67	1.75	1.70	2
49	2008-08-27	CHL	2.15	ΣΤΕΑΟΥΑ	3.25	ΓΑΛΑΤΑΣΑΡΑΪ	2.55	1.29	1.17	1.43	1.80	1.65	0
50	2008-08-27	CHL	1.25	ΦΕΝΕΡΜΠΑΧΤΣΕ	4.20	ΠΑΡΤΙΖΑΝ	8.00	0.00	0.00	2.75	1.90	1.57	1
52	2008-08-27	CHL	1.65	ΒΑΣΙΛΕΙΑ	3.25	ΓΚΙΜΑΡΑΕΣ	4.00	0.00	1.17	1.79	1.65	1.80	1
53	2008-08-27	CHL	1.70	ΑΤΛΕΤΙΚΟ ΜΑΔΡΙΤΗΣ	3.40	ΣΑΛΚΕ	3.55	1.13	1.15	1.74	1.65	1.80	1
54	2008-08-27	CHL	1.20	ΜΑΡΣΕΪΓ	4.80	ΜΠΡΑΝ	8.00	0.00	0.00	3.00	1.80	1.65	0
55	2008-08-27	CHL	2.25	ΝΤΙΝΑΜΟ ΖΑΓΚΡΕΜΠ	3.25	ΣΑΧΤΑΡ ΝΤΟΝΕΤΣΚ	2.40	1.33	1.16	1.38	1.70	1.75	0
56	2008-08-27	CHL	1.15	ΟΛΥΜΠΙΑΚΟΣ	5.25	ΑΝΟΡΘΩΣΙΣ	10.00	0.00	0.00	3.44	2.30	1.35	0
57	2008-08-27	CHL	2.75	ΣΛΑΒΙΑ ΠΡΑΓΑΣ	3.10	ΦΙΟΡΕΝΤΙΝΑ	2.10	1.46	1.19	1.25	1.65	1.80	0
67	2008-08-27	CHL	1.15	ΑΡΣΕΝΑΛ	5.25	ΤΒΕΝΤΕ	10.00	0.00	0.00	3.44	1.80	1.65	1
68	2008-08-27	CHL	1.12	ΛΙΒΕΡΠΟΥΛ	5.50	ΣΤΑΝΤΑΡ ΛΙΕΓΗΣ	11.00	0.00	0.00	3.67	1.80	1.65	0
910	2008-09-16	CHL	2.25	ΑΪΝΤΧΟΦΕΝ	3.20	ΑΤΛΕΤΙΚΟ ΜΑΔΡΙΤΗΣ	2.55	1.32	1.20	1.42	1.50	2.00	0
911	2008-09-16	CHL	2.25	ΒΑΣΙΛΕΙΑ	3.20	ΣΑΧΤΑΡ ΝΤΟΝΕΤΣΚ	2.55	1.32	1.20	1.42	1.85	1.60	0
912	2008-09-16	CHL	1.16	ΒΕΡΝΤΕΡ ΒΡΕΜΗΣ	5.25	ΑΝΟΡΘΩΣΙΣ	10.00	0.00	0.00	3.44	2.35	1.35	0
913	2008-09-16	CHL	2.85	ΜΑΡΣΕΪΓ	3.10	ΛΙΒΕΡΠΟΥΛ	2.10	1.48	1.21	1.25	1.45	2.10	1
914	2008-09-16	CHL	1.30	ΜΠΑΡΤΣΕΛΟΝΑ	3.90	ΣΠΟΡΤΙΝΓΚ ΛΙΣ.	8.00	0.00	1.12	2.62	1.90	1.55	1
915	2008-09-16	CHL	3.60	ΠΑΝΑΘΗΝΑΪΚΟΣ	3.05	ΙΝΤΕΡ	1.85	1.65	1.22	1.15	1.50	2.00	0
916	2008-09-16	CHL	1.22	ΡΟΜΑ	4.50	ΕΚΟΜΑΞ ΚΛΟΥΖ	9.00	0.00	0.00	3.00	1.90	1.55	1
917	2008-09-16	CHL	1.25	ΤΣΕΛΣΙ	4.20	ΜΠΟΡΝΤΟ	9.00	0.00	0.00	2.86	1.70	1.75	2
942	2008-09-17	CHL	1.65	ΠΟΥΒΕΝΤΟΥΣ	3.25	ΖΕΝΙΤ	3.85	0.00	1.16	1.76	1.65	1.80	0

Fetched: 1000 | Fetch All | Note: result is limited by 1000 rows

Εικόνα 3.2
Ο πίνακας CouponInfo

3.4 ΤΕΧΝΙΚΗ ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Έχοντας περιγράψει σε θεωρητικό επίπεδο την εφαρμογή μας, είμαστε πλέον σε θέση να δείξουμε, πώς υλοποιούνται τα παραπάνω στην πράξη. Η υλοποίηση της

εφαρμογής έγινε στην γλώσσα java. Οι λόγοι για τους οποίους επιλέχτηκε η συγκεκριμένη γλώσσα προγραμματισμού είναι:

1. Διαλειτουργικότητα: η εφαρμογές σε java έχουν την ιδιότητα να μπορούν να εκτελούνται σε μηχανές διαφορετικής αρχιτεκτονικής αλλά και με διαφορετικό λειτουργικό σύστημα, κατά τον ίδιο τρόπο.
2. Η java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, οπότε ταιριάζει με την προσέγγιση που ακολουθήσαμε για την ανάπτυξη της εφαρμογής.
3. Προσφέρει ποικίλες βιβλιοθήκες (APIs) που καθιστούν εύκολη την σχεδίαση του graphical user interface της εφαρμογής, δηλαδή τη γραφική διασύνδεση του χρήστη με την εφαρμογή.

Η συγγραφή του κώδικα πραγματοποιήθηκε στο περιβάλλον NetBeans 6.9 το οποίο είναι ένα από τα πιο γνωστά ολοκληρωμένα περιβάλλοντα ανάπτυξης λογισμικού (IDE) το οποίο προσφέρει διάφορα βοηθητικά εργαλεία όπως το Matisse, το οποίο μας βοήθησε στο σχεδιασμό του GUI. Στις γραμμές που ακολουθούν, θα αναλύσουμε τον κώδικα τις εφαρμογής. Πιο συγκεκριμένα, θα περιγράψουμε τα πακέτα στα οποία χωρίζεται η εφαρμογή, τις κλάσεις καθώς και τις μεθόδους οι οποίες υλοποιούν τις λειτουργίες του συστήματος.

3.4.1 Πακέτο XMLParsing

Το πακέτο αυτό, όπως φαίνεται και στην αρχιτεκτονική του συστήματος (εικόνα 4.1), αποτελεί το τμήμα της εφαρμογής, το οποίο ασχολείται με όλες τις λειτουργίες που έχουν να κάνουν, τόσο με την προσπέλαση των απομακρυσμένων πόρων (αρχεία XML) όσο και με το parsing αυτών, προκειμένου να διαβαστούν όλες οι απαραίτητες πληροφορίες. Παρακάτω περιγράφονται συνοπτικά όλες οι κλάσεις του πακέτου αυτού.

3.4.1.1 Κλάση DOMParser

Αποτελεί ίσως την πιο σημαντική κλάση της εφαρμογής, από την άποψη ότι είναι η πηγή εισόδου πληροφοριών στο σύστημα. Είναι προφανές ότι χωρίς τις πληροφορίες αυτές, η εφαρμογή μας θα ήταν «άχρηστη». Η κλάση αύτη, είναι υπεύθυνη για τη σύνδεση στα απομακρυσμένα αρχεία XML, καθώς και για το parsing των αρχείων αυτών. Η σύνδεση στα αρχεία πραγματοποιείται μέσω διαδικτύου, με τη χρήση του ενσωματωμένου πακέτου java.net. Το parsing των αρχείων XML γίνεται με τη βοήθεια της βιβλιοθήκης jdom. Οι μέθοδοι που υλοποιούν τις παραπάνω λειτουργίες είναι:

- Public void startparsing(String URL): η μέθοδος αυτή χρησιμοποιεί τη κλάση DOMParser από τη βιβλιοθήκη jdom, για να εκτελέσει το parsing του XML αρχείου που βρίσκεται στο URL που περνάει ως παράμετρος. Επειδή το

αρχείο XML αποτελείται από αρκετά στοιχεία (elements) τα περισσότερα εκ των οποίων δεν μας χρειάζονται, η κλάση υλοποιεί ένα φίλτρο με το οποίο συλλέγει μόνο τα απαραίτητα στοιχεία. Εφόσον εκτελεστεί επιτυχώς το parsing, γίνεται σύνδεση με τον DBManager (περιγράφεται στην συνέχεια), ο οποίος εισάγει τα στοιχεία που διαβάστηκαν στον πίνακα της βάσης δεδομένων.

- Public void updateTables(boolean initialUpdate): η μέθοδος αυτή, δίνει τη δυνατότητα στο χρήστη να ενημερώσει την βάση δεδομένων. Αν η παράμετρος initialUpdate είναι true τότε, γίνεται ολική ενημέρωση της βάσης δεδομένων από την αρχή, αλλιώς πραγματοποιείται ενημέρωση από το τελευταίο αρχείο XML που διαβάστηκε και μετά. Για κάθε νέο αρχείο XML, καλεί την startparsing για να γίνει το parsing.

- Private int getValidIDs(Vector<Integer> couponIDs, int couponID): στο σημείο αυτό θα πρέπει να σημειωθεί ότι τα αρχεία XML είναι αποθηκευμένα στους server του ΟΠΑΠ. Το URL των αρχείων αυτών, αποτελείτε από ένα σταθερό τμήμα που είναι ίδιο για όλα τα αρχεία και από ένα τριψήφιο αριθμό που βρίσκεται στο τέλος της συμβολοσειράς, το οποίο είναι μοναδικό για κάθε αρχείο. Ένα πρόβλημα που αντιμετωπίσαμε κατά την σύνδεση σε αυτά τα αρχεία είναι ότι, ο τριψήφιος αριθμός στο τέλος του URL, δεν είναι συνεχόμενος π.χ. εάν για ένα αρχείο ο τριψήφιος αριθμός είναι το 154, δεν είναι σίγουρο ότι το επόμενο αρχείο θα έχει τον αριθμό 155. Οπότε έπρεπε να βρούμε έναν τρόπο ώστε να εξετάζονται πια URL αντιστοιχούν σε XML αρχεία. Η μέθοδος getValidIDs, εκτελεί αυτήν ακριβώς τη λειτουργία. Δέχεται ως παράμετρο έναν τριψήφιο αριθμό couponID και ελέγχει πια URL από couponID και μετά, αντιστοιχούν σε XML αρχεία. Τα URL αυτά τα αποθηκεύει στο Vector couponIDs (πιο συγκεκριμένα, αποθηκεύει τον τριψήφιο αριθμό των URLs) και επιστρέφει το πλήθος των αρχείων που εντόπισε. Η αναζήτηση για XML αρχεία τερματίζει όταν 20 συνεχόμενα URL δεν αντιστοιχούν σε XML αρχεία.

- Public HashSet <FootballMatchInfo> parseLastCoupon(): η μέθοδος αυτή, πραγματοποιεί parsing στο τελευταίο αρχείο XML, το URL του οποίου το ανακτά από τα μετά-δεδομένα της εφαρμογής. Τελευταίο αρχείο XML, θεωρείται το αρχείο που περιλαμβάνει τα παιχνίδια που αναμένονται να γίνουν, δηλαδή τα παιχνίδια για τα οποία γνωρίζουμε μόνο τις αποδόσεις του και όχι τα αποτελέσματα. Τα στοιχεία τα οποία κρατάει είναι, η γηπεδούχος ομάδα, η φιλοξενούμενη ομάδα, η ημερομηνία διεξαγωγής του αγώνα και οι αποδόσεις των τριών σημείων. Οι πληροφορίες αυτέ αποθηκεύονται στα αντικείμενα της κλάσης FootballMatchInfo η οποία περιγράφεται παρακάτω.

- Public void readPredictionResults(): η μέθοδος αυτή διαβάζει το XML αρχείο το οποίο περιλαμβάνει τους αγώνες για τους οποίους έχει γίνει πρόβλεψη τελευταία. Κρατάει τα ίδια στοιχεία με την parseLastCoupon καθώς και τα σκορ των αγώνων και τα αποθηκεύει σε αντικείμενα της κλάσης MatchResults.

3.4.1.2 Κλάση FootballMatchInfo

Η κλάση αυτή αναπαριστά έναν αγώνα ποδοσφαίρου. Αποθηκεύει όλες τις πληροφορίες που είναι απαραίτητες για την πραγματοποίηση της πρόβλεψης του αγώνα. Τα στοιχεία που αποθηκεύει είναι:

- league: αποτελεί τη διοργάνωση στα πλαίσια της οποίας πραγματοποιείται ο συγκεκριμένος αγώνας.
- date: είναι η ημερομηνία διεξαγωγής του αγώνα.
- homeTeam: η γηπεδούχος ομάδα του αγώνα.
- awayTeam: η φιλοξενούμενη ομάδα του αγώνα.
- apodosi1: η απόδοση του σημείου 1, στο συγκεκριμένο αγώνα.
- apodosiX: η απόδοση του σημείου X, στο συγκεκριμένο αγώνα.
- apodosi2: η απόδοση του σημείου 2, στο συγκεκριμένο αγώνα.
- prediction: η πρόβλεψη για το συγκεκριμένο αγώνα. Η προκαθορισμένη τιμή της μεταβλητής αυτή είναι ο χαρακτήρας '?'.

3.4.1.3 Κλάση MatchResults

Η κλάση αυτή αντιπροσωπεύει έναν αγώνα ποδοσφαίρου συμπεριλαμβανομένου και του αποτελέσματος. Η κλάση αυτή κληρονομεί την κλάση FootballmatchInfo, οπότε έχει τα ίδια χαρακτηριστικά με αυτήν και επιπλέον έχει άλλα τρία επιπρόσθετα χαρακτηριστικά.

- homeTeamGoalsScored: το σύνολο των τερμάτων που πέτυχε η γηπεδούχος ομάδα, στο συγκεκριμένο αγώνα.
- awayTeamGoalsScored: το σύνολο των τερμάτων που πέτυχε η φιλοξενούμενη ομάδα, στο συγκεκριμένο αγώνα.
- simio: το αποτέλεσμα του συγκεκριμένου παιχνιδιού (1,X,2).

3.4.1.4 Κλάση MetaData

Η κλάση αυτή αποθηκεύει τα μεταδομένα της εφαρμογής, τα οποία είναι χρήσιμα για την εκτέλεση διάφορων λειτουργιών, όπως η parseLastCoupon κ.α. Η κλάση αυτή

κάνει implement τη διασύνδεση serializable, ώστε τα αντικείμενα αυτής, να μπορούν να αποθηκεύονται στο δίσκο. Αποτελείται από τα εξής χαρακτηριστικά:

- lastCouponId: είναι ο τριψήφιος αριθμός του URL που αντιστοιχεί στο XML αρχείο το οποίο περιλαμβάνει τους αγώνες που βρίσκονται στο πιο πρόσφατο κουπόνι του ΟΠΑΠ.
- lastPredictedCoupon: είναι ο τριψήφιος αριθμός του URL που αντιστοιχεί στο XML αρχείο το οποίο περιλαμβάνει τα παιχνίδια για τα οποία έχει γίνει πρόβλεψη πιο πρόσφατα.

Τέλος, η κλάση αυτή περιλαμβάνει δυο μεθόδους για την αποθήκευση και την ανάγνωση των μετά-δεδομένων, από και προς το δίσκο.

3.4.2 Πακέτο Exceptions

Όπως είναι γνωστό, η java προσφέρει έναν πολύ σημαντικό μηχανισμό, που ονομάζεται *διαχείριση εξαιρέσεων (exception handling)*. Αποτελεί έναν σημαντικό μηχανισμό για την διαχείριση καταστάσεων οι οποίες οδηγούν σε μη ορθή λειτουργία του συστήματος (π.χ. IOException για διαχείριση σφαλμάτων IO). Οπότε ακολουθώντας αυτή τη λογική της java, δημιουργήσαμε δυο πιθανές εξαιρέσεις που μπορούν να προκύψουν κατά την εκτέλεση της εφαρμογής μας.

- EndOfCoupon Exception: η εξαίρεση αυτή προκύπτει όταν κατά το parsing του αρχείου XML φτάνουμε στο τέλος του.
- NoXMlPageException: η εξαίρεση αυτή προκύπτει όταν κατά την αναζήτηση αρχείων XML, εντοπιστεί ένα αρχείο διαφορετικού τύπου (τις περισσότερες φορές τα αρχεία αυτά είναι HTML).

3.4.3 Πακέτο Utils

Οι περισσότερες εφαρμογές αν όχι όλες, έχουν ένα πακέτο το οποίο υλοποιεί ένα σύνολο χρήσιμων λειτουργιών, τις οποίες χρησιμοποιούν οι κλάσεις των άλλων πακέτων. Έτσι και στην εφαρμογή μας, δημιουργήσαμε το πακέτο αυτό, το οποίο αποτελείται από μια μόνο κλάση την Util και η οποία έχει τις παρακάτω μεθόδους.

- Public String convertDate(String date): προηγουμένως δείξαμε ότι ο πίνακας couponInfo έχει ένα πεδίο date. Για να γίνει εισαγωγή ενός τέτοιου πεδίου, η MySQL απαιτεί μια συγκεκριμένη αναπαράσταση της ημερομηνίας, η οποία είναι διαφορετική από την αναπαράσταση που υπάρχει στο XML αρχείο. Η μέθοδος convertDate μετατρέπει την ημερομηνία που διαβάζεται από το XML, σε μορφή συμβατή με την MySQL.

- Public String convertMonth(String month): υλοποιεί κάτι αντίστοιχο με την convertDate μόνο που μετατρέπει μόνο το μήνα που περνάει ως παράμετρος σε κατάλληλη μορφή.

- public float stringToFloat(String elementContent): στο αρχείο XML elements όπως το " Apososi1" έχουν περιεχόμενο τύπου float. Το πρόβλημα είναι ότι, η αναπαράσταση των float αριθμών δεν είναι συμβατοί με την αναπαράσταση των float αριθμών που υποστηρίζει η java. Πιο συγκεκριμένα, στο αρχείο XML τα δεκαδικά ψηφία του πραγματικού αριθμού, χωρίζονται με το χαρακτήρα "," και όχι το χαρακτήρα "." Που απαιτεί η java. Η μέθοδος stringToFloat() μετατρέπει ένα string, που αντιστοιχεί σε έναν πραγματικό αριθμό που χρησιμοποιεί την πρώτη αναπαράσταση, σε ένα συμβατό με τη java πραγματικό αριθμό.

3.4.4 Πακέτο Statistics

Όπως αναφέραμε στην αρχή, μια από τις λειτουργίες που πρέπει να προσφέρει το σύστημα, είναι η παρουσίαση στατιστικών στοιχείων, που σχετίζονται με τους αγώνες ποδοσφαίρου. Το πακέτο αυτό περιλαμβάνει τις κλάσεις που καθιστούν εφικτή την παραπάνω λειτουργία.

3.4.4.1 Κλάση Statistic

Είναι η πιο βασική κλάση του πακέτου. Υλοποιεί την πλειοψηφία των λειτουργιών που σχετίζονται με τη στατιστική ανάλυση των δεδομένων μας. Όλες σχεδόν οι λειτουργίες απαιτούν τη σύνδεση με τη βάση δεδομένων, η οποία πραγματοποιείται με τη βοήθεια του πακέτου DBManager, το οποίο θα το περιγράψουμε στη συνέχεια. Συγκεκριμένα η κλάση υλοποιεί τις εξής μεθόδους:

- private void getDistinctTeam(): αποτελεί μια βοηθητική μέθοδο, η οποία εκτελείται μια μόνο φορά, κατά την εγκατάσταση του προγράμματος. Προσπαθεί να εντοπίσει τις ομάδες που είναι αποθηκευμένες στη βάση δεδομένων. Στη συνέχεια , τις αποθηκεύει σε ένα hashset το οποίο το εγγράφει στο δίσκο για μόνιμη αποθήκευση. Είναι μια απαραίτητη διαδικασία που πρέπει να γίνει προκειμένου να σχηματιστεί το σύνολο όλων των ομάδων για τις οποίες στη συνέχεια, θα μπορούμε να ανακτούμε ορισμένες χρήσιμες πληροφορίες. Το σύνολο αυτό αποτελείται από μοναδικά στοιχεία. Όπως θα δούμε στη συνέχεια, τις πληροφορίες για τις ομάδες θα τις χρησιμοποιήσουμε για τους σκοπούς της μηχανικής μάθησης.

- private String getCurrentPeriod(): η μέθοδος αυτή είναι απαραίτητη για να σχηματίσουμε στη συνέχεια τους πίνακες βαθμολογίας των πρωταθλημάτων.

Σκοπός της είναι να εντοπιστεί η τρέχουσα χρονική περίοδος, για την οποία θέλουμε να δούμε την βαθμολογική κατάσταση των ομάδων. Για παράδειγμα, εάν η μέθοδος εκτελεστεί την ημερομηνία 03/02/2010 τότε θα επιστρέψει ως τρέχουσα περίοδο διεξαγωγής του πρωταθλήματος, την 2009-2010, οπότε ο υπολογισμός του βαθμολογικού πίνακα που θα γίνει μετέπειτα, θα υπολογίσει τους βαθμούς των ομάδων για την περίοδο 2009-2010. Το σκεπτικό της υλοποίησης βασίζεται στην απλή ιδέα ότι, αν η ημερομηνία εκτέλεσης της μεθόδου, είναι μετά από την πρώτη Αυγούστου, τότε η περίοδος διεξαγωγής του πρωταθλήματος είναι η χρονιά της τρέχουσας ημερομηνίας και η επόμενη χρονιά απ αυτήν. Αλλιώς είναι η προηγούμενη χρονιά.

- Public HashSet <Team> getChampionshipTeamsOfCurrentPeriod(String seassonStart, String league): είχαμε αναφέρει ότι τα δεδομένα σχετικά με τους αγώνες ποδοσφαίρου συλλέγονται αυτόματα από αρχεία XML. Ξεχάσαμε να αναφέρουμε ότι, η ιστοσελίδα του ΟΠΑΠ προσφέρει τα στοιχεία για τους αγώνες που έχουν γίνει από το 2008 και μετά. Επίσης είναι γνωστό ότι σε κάθε πρωτάθλημα ποδοσφαίρου, στο τέλος της χρόνιας υπάρχουν ομάδες που υποβιβάζονται και άλλες που προάγονται. Οπότε για να μπορέσουμε να υπολογίσουμε τους βαθμούς των ομάδων ενός πρωταθλήματος, θα πρέπει να γνωρίζουμε ποίες ομάδες συμμετέχουν την τρέχουσα περίοδο στο πρωτάθλημα. Η μέθοδος, getChampionshipTeamsOfCurrentPeriod εκτελεί αυτήν ακριβώς τη δουλεία. Δέχεται ως παράμετρο την τρέχουσα χρονική περίοδο (βλέπε getCurrentPeriod) καθώς και το πρωτάθλημα, και αναζητά της ομάδες του συμμετέχουν στο συγκεκριμένο πρωτάθλημα την συγκεκριμένη χρονική περίοδο. Αφού εντοπίσει της ομάδες, τις αποθηκεύει σε ένα HashSet από αντικείμενα τύπου Team (περιγράφεται παρακάτω), το οποίο και επιστρέφει.

- Public void computeLeagueTable(String league): όπως προδίδει και το όνομα της, υπολογίζει τον πίνακα βαθμολογία των ομάδων ενός συγκεκριμένου πρωταθλήματος. Βασίζεται στον απλό κανόνα που λέει ότι όταν μια ομάδα κερδίζει σε έναν αγώνα, τότε παίρνει τρείς βαθμού, όταν το αποτέλεσμα είναι ισοπαλία τότε παίρνει έναν βαθμό και όταν χάνει δεν παίρνει κανέναν βαθμό. Οπότε ελέγχοντας όλα τα παιχνίδια κάθε ομάδας μέχρι εκείνη τη στιγμή και προσθέτοντας όλους τους βαθμούς, υπολογίζεται η συνολική βαθμολογία και κατ' επέκταση, η βαθμολογική της θέση. Οι προηγούμενες δύο μέθοδοι, βοηθούν στη διαδικασία αυτή.

- public void computeStatistics(String date, String league, String team, Vector <String> statistics): είναι η μέθοδος η οποία υπολογίζει τα στατιστικά στοιχεία για μία ομάδα. Πιο συγκεκριμένα, υπολογίζει για την ομάδα *team τα* εξής στοιχειά:
    1. Το σύνολο των αγώνων που κέρδισε η ομάδα, εντός έδρας.
    2. Το σύνολο των αγώνων που κέρδισε η ομάδα εκτός έδρας.

3. Το σύνολο των αγώνων που έληξαν ισοπαλία ανεξαρτήτως αν πραγματοποιήθηκαν στην έδρα της ομάδας ή εκτός.
4. Το σύνολο των τερμάτων που πέτυχε η ομάδα.
5. Τα τέρματα που πέτυχε η ομάδα, ανά αγώνα.
6. Ο αγώνας στον οποίο συμμετείχε η ομάδα και επιτεύχθηκαν τα περισσότερα τέρματα από οποιοδήποτε άλλο αγώνα.

Θα πρέπει να σημειωθεί ότι τα παραπάνω στατιστικά στοιχεία υπολογίζονται για μία συγκεκριμένη ομάδα, ενός συγκεκριμένου πρωταθλήματος και για την περίοδο από μια συγκεκριμένη ημερομηνία (παράμετρος date) και μετά.

3.4.4.2 Κλάση Team

Η κλάση αυτή αναπαριστά μία ομάδα ποδοσφαίρου. Κάθε αντικείμενο Team έχει τα παρακάτω χαρακτηριστικά:

- String name: το όνομα της ομάδας.
- Int points: το σύνολο των βαθμών που έχει συγκεντρώσει η ομάδα.
- Int currentPosition: η τρέχουσα βαθμολογική θέση της ομάδας.
- String league: το πρωτάθλημα στο οποίο συμμετέχει η ομάδα.

3.4.4.3 Κλάση Chart

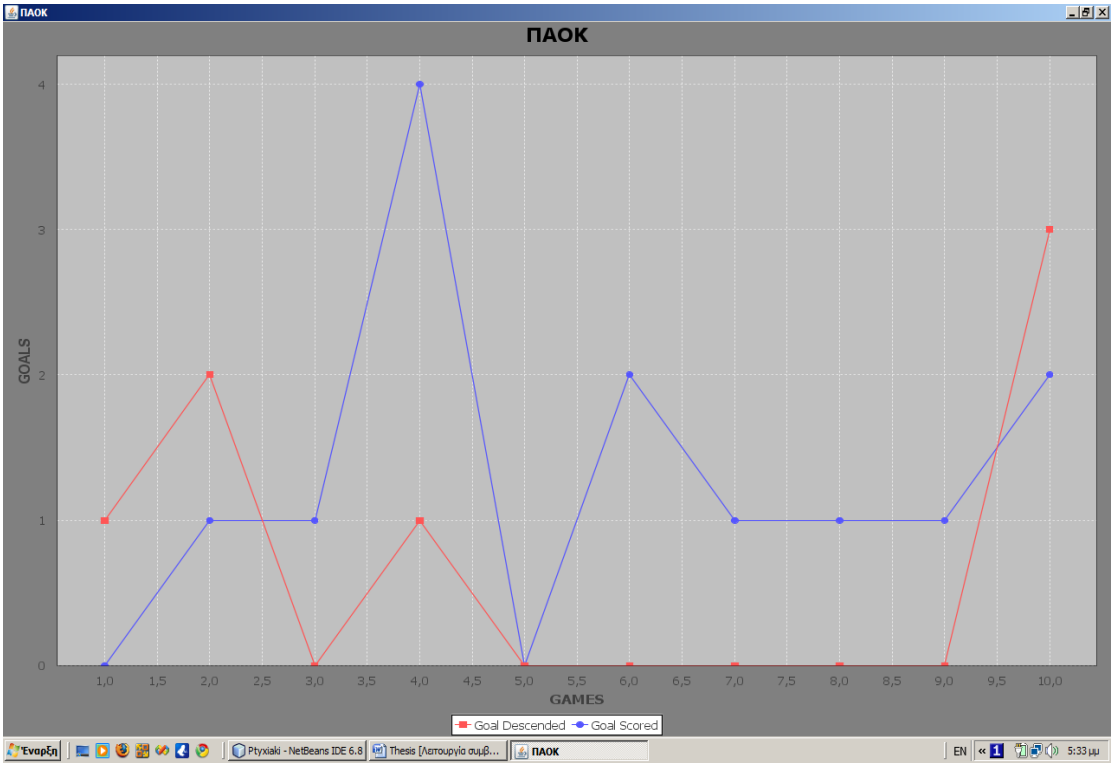
Ένα σημαντικό στοιχείο στη στατιστική ανάλυση των δεδομένων είναι και η παρουσίαση των στοιχείων διαγραμματικά ώστε να γίνεται πιο εύκολα αντιληπτή η πληροφορία που παρουσιάζεται. Έτσι κρίθηκε χρήσιμο να παρουσιαστούν τα αποτελέσματα κάθε ομάδας σε μία γραφική παράσταση. Με αυτόν τον τρόπο πιστεύω ότι, κανείς μπορεί εύκολα να σχηματίσει άποψη για την αγωνιστική κατάσταση μιας ομάδας. Η κλάση Chart, προσφέρει αυτή τη δυνατότητα στο χρήστη.

- Private XYDataset createDataset(): η μέθοδος αυτή είναι υπεύθυνη να σχηματίσει το σύνολο των δεδομένων που θα παρουσιαστούν στη γραφική παράσταση. Πιο συγκεκριμένα, τα στοιχεία αυτά είναι, τα τέρματα που πέτυχε μια ομάδα καθώς και αυτά που δέχτηκε, σε κάθε ένα από τα τελευταία δέκα παιχνίδια που έδωσε.

- private JFreeChart createChart(XYDataset dataset): η μέθοδος αυτή, δέχεται το σύνολο δεδομένων που σχημάτισε η createDataset και δημιουργεί το γράφημα σύμφωνα με αυτό.

- Public void designChart(): εμφανίζει το γράφημα που δημιούργησε η createChart στο χρήστη. Το αποτέλεσμα της εκτέλεσης της μεθόδου αυτής φαίνεται στην εικόνα 4.3. Στον κάθετο άξονα του γραφήματος βρίσκονται τα τέρματα ενώ στον οριζόντιο άξονα είναι οι αγώνες. Τα κόκκινα σημεία δείχνουν τα τέρματα που δέχτηκε η ομάδα σε κάθε αγώνα, ενώ τα μπλε σημεία δείχνουν τα τέρματα που πέτυχε. Οι δυο γραμμές που ενώνουν τα σημεία αυτά, δείχνουν την πορεία της ομάδας στα τελευταία δέκα παιχνίδια. Όπως φαίνεται και στην εικόνα, η μπλε γραμμή είναι σε πολλά σημεία πάνω από την κόκκινη, αυτό σημαίνει ότι η ομάδα έχει περισσότερες νίκες απ' ότι ήττες, και άρα συμπεραίνουμε ότι βρίσκετε σε καλή αγωνιστική κατάσταση.

Η υλοποίηση των παραπάνω λειτουργιών έγινε με την βοήθεια δύο βιβλιοθηκών (APIs), της jfreechart και της jcommon.



Εικόνα 3.3

Γράφημα που παρουσιάζει τα αποτελέσματα της ομάδας του ΠΑΟΚ στα τελευταία δέκα παιχνίδια.

3.4.5 Πακέτο DBManipulation

Το πακέτο αυτό βρίσκεται στο data layer της εφαρμογή. Αποτελεί τον μοναδικό τρόπο προσπέλασης των δεδομένων που βρίσκονται στη βάση δεδομένων του συστήματος. Στο σχήμα 4.1 φαίνεται η σημαντικότητα του πακέτου αυτού, καθώς όλα τα τμήματα στο application layer του συστήματος, έχουν την ανάγκη της προσπέλασης των πληροφοριών της βάση δεδομένων.

3.4.5.1 Κλάση DBManager

Αποτελεί τη μοναδική κλάση του πακέτου DBManipulation. Υλοποιεί όλες τις λειτουργίες που είναι απαραίτητες για τη διαχείριση της βάσης δεδομένων (εισαγωγή δεδομένων, διαγραφή, ενημέρωση κ.α.). Όπως έχουμε αναφέρει στην αρχή του κεφαλαίου αυτού, επιλέξαμε να χρησιμοποιήσουμε το ΣΔΒΔ MySQL. Η χρήση ενός ΣΔΒΔ μας λύνει τα χέρια, γιατί μας επιτρέπει να εκτελέσουμε λειτουργίες διαχείρισης βάσεων δεδομένων χωρίς να χρειαστεί να υλοποιήσουμε τίποτα εμείς. Οπότε η κλάση DBManager δεν κάνει τίποτα παραπάνω από το να ζητάει από την MySQL την εκτέλεση διάφορων λειτουργιών. Για να επιτευχθεί η επικοινωνία της κλάσης με την MySQL, χρειάστηκε να χρησιμοποιηθεί ο driver *mysql-connector-java* ο οποίος υλοποίει τη διασύνδεση JDBC (σύνολο από interfaces, που ορίζουν τους κανόνες επικοινωνίας μιας εφαρμογής java με οποιοδήποτε ΣΔΒΔ) της java. Παρακάτω περιγράφουμε τις βασικές μεθόδους της κλάσης.

- public void CreateDatabase(): όπως γίνεται αντιληπτό, η μέθοδος αυτή είναι υπεύθυνη για τη δημιουργία της βάσης δεδομένων της εφαρμογής. Εκτελείτε μία μόνο φορά, κατά την εγκατάσταση της εφαρμογής.

- public void connectToDatabase(): προτού εκτελεστεί οποιαδήποτε λειτουργία, θα πρέπει να γίνει η σύνδεση της εφαρμογής στη βάση δεδομένων. Η μέθοδος αυτή, υλοποιεί τη σύνδεση αυτή.

- public void disconnectFromDatabase(): πραγματοποιεί την αποσύνδεση της εφαρμογής από τη βάση δεδομένων. Όταν δεν χρειαζόμαστε άλλο τη βάση δεδομένων, καλό είναι να αποδεσμεύουμε τους πόρους.

- public void insertRecords(DomParser sp, int startIndex): η μέθοδος αυτή εισάγει τα δεδομένα που διάβασε η κλάση DomParser (βλέπε κλάση DomParser) στον πίνακα της βάσης δεδομένων.

- public void emptyTables(): αδειάζει τους πίνακες της βάσης δεδομένων. Η μέθοδος αυτή χρησιμοποιείται στην περίπτωση που θέλουμε να εκτελέσουμε ολική ενημέρωση στη βάση όπως την περιγράψαμε παραπάνω.

- public ResultSet executeQuery(String query): μέθοδος που δίνει τη δυνατότητα σε οποιοδήποτε τμήμα της εφαρμογής να εκτελέσει ένα ερώτημα προς τη βάση. Επιστρέφει το αποτέλεσμα της εκτέλεσης του ερωτήματος.

3.4.6 Πακέτο MachineLearning

Περιλαμβάνει τις κλάσεις που σχηματίζουν το τμήμα της εφαρμογής που είναι υπεύθυνο για τις λειτουργίες που σχετίζονται με τη μηχανική μάθηση. Είναι το τμήμα το οποίο επικοινωνεί με το WEKA για την εκτέλεση αλγορίθμων ταξινόμησης. Στις γραμμές που ακολουθούν, παρουσιάζονται οι κλάσεις από τις οποίες αποτελείται το πακέτο.

3.4.6.1 Κλάση ARFFCreator

Η κλάση ARFFCreator είναι υπεύθυνη για τη δημιουργία των συνόλων εκπαίδευσης που χρησιμοποιήθηκαν στα πειράματα μας αλλά και του συνόλου εκπαίδευση που χρησιμοποιήθηκε στην εφαρμογή. Δίνει επίσης τη δυνατότητα αποθήκευσης των συνόλων εκπαίδευση αλλά και των συνόλων ελέγχου σε αρχεία .arff (αρχεία συμβατά με το εργαλείο weka). Αποτελείται από τις εξής μεθόδους:

- public Instances getNlastGamesOfTeamsTrainingSet(): η μέθοδος αυτή είναι υπεύθυνη για τη δημιουργία του συνόλου εκπαίδευσης, του οποίου τα χαρακτηριστικά τα έχουμε περιγράψαμε προηγουμένως. Επιστρέφει τα στιγμιότυπα του συνόλου.

- public Instances getNlastGamesOfTeamsTestSet(HashSet teamSet): στο κεφάλαιο 2, αναφέραμε πώς πέρα από το στάδιο της μάθησης, υπάρχει και το στάδιο της ταξινόμησης στο οποίο πραγματοποιείται ουσιαστικά η πρόβλεψη. Για να πραγματοποιηθεί η πρόβλεψη, θα πρέπει να εφαρμοστεί το μοντέλο ταξινόμησης που σχηματίστηκε κατά την μάθηση, σε ένα σύνολο ελέγχου. Η μέθοδος αυτή είναι υπεύθυνη για το σχηματισμό του συνόλου αυτού.

- public void createarff(Hashtable <Integer,String> instances): γενική μέθοδος για το σχηματισμό ενός συνόλου εκπαίδευσης και αποθήκευση του σε μορφή αρχείου .arff για να μπορέσει να χρησιμοποιηθεί στη συνέχεια από το εργαλείο weka. Γι' αυτό το λόγο ακολουθεί το πρότυπο ενός αρχείου .arff. Τα στιγμιότυπα που σχηματίζουν το σύνολο εκπαίδευσης, περνάνε ως παράμετρος στη μέθοδο.

- public void getResultsWithSimilarOponentsTrainingSet(boolean writeTeam): μέθοδος για τη δημιουργία του συνόλου εκπαίδευσης "similar.arff" το οποίο παρουσιάζεται στο παράρτημα. Η παράμετρος writeTeam δηλώνει εάν θα συμπεριληφθεί και το όνομα των ομάδων.

- public void getResultsWithSimilarOponentsTestSet: μέθοδος για τη δημιουργία του συνόλου ελέγχου που έχει τα ίδια χαρακτηριστικά με το σύνολο "similar.arff".

- public void getUnderOverTrainingSet(boolean writeTeam): μέθοδος για τη δημιουργία του συνόλου εκπαίδευσης "underOver.arff" το οποίο παρουσιάζεται στο παράρτημα.

- private Instances initializeTrainingSet(FastVector attributes) : όπως λέει και το όνομα της, η μέθοδος αρχικοποιεί το σύνολο εκπαίδευσης που θα χρησιμοποιηθεί στην εφαρμογή, για τη δημιουργία του συνόλου εκπαίδευσης. Τα χαρακτηριστικά του συνόλου τα περιγράψαμε παραπάνω.

3.4.6.2 Κλάση Evaluator

Κάθε φορά που στοιχηματίζουμε στους αγώνες σύμφωνα με τις προβλέψεις που κάνει το σύστημα, θα θέλαμε μετά από την ολοκλήρωση των αγώνων, να βλέπουμε το πραγματικό κέρδος ή ενδεχομένως το πραγματικό κόστος που αποκομίζουμε. Η κλάση Evaluator παρέχει την παραπάνω δυνατότητα. Οι σημαντικές μέθοδοι της κλάσης είναι:

- public void evaluatePredictions(HashSet <MatchResults> matchResults): η μέθοδος αυτή δέχεται τα αποτελέσματα των αγώνων που στοιχηματίσαμε και σε συνδυασμό με τις προβλέψεις του κατηγοριοποιητή, υπολογίζει τη χρηματική απολαβή.

- public void loadPredictionInfo(): διαβάζει τις προβλέψεις που έχει κάνει ο κατηγοριοποιητής για τα αποτελέσματα των αγώνων.

3.4.6.3 Κλάση Predictor

Είναι η κλάση η οποία συνδέεται με το WEKA ώστε να δημιουργηθεί το μοντέλο ταξινόμησης καθώς και να γίνει η πρόβλεψη. Για την αποδοτική εκτέλεση της εφαρμογής, η κλάση αυτή υλοποιεί τη διασύνδεση *Runnable*, δηλαδή εκτελείται ως

ανεξάρτητο νήμα (thread). Στη συνέχεια παρουσιάζουμε τις σημαντικές μεθόδους της κλάσης.

- private void createCostMatrix(): μέθοδος υπεύθυνη για τη δημιουργία του πίνακα κόστους (βλέπε κεφάλαιο 2.3). Το κόστος στη δική μας περίπτωση είναι διαφορετικό για κάθε παράδειγμα (*example dependent cost matrix*).

- public void buildCliassifer(): μέθοδος η οποία επικοινωνεί με το WEKA με σκοπό τη κατασκευή του κατηγοριοποιητή. Ο αλγόριθμος που χρησιμοποιείται είναι ο *MetaCost*, τον οποίο τον τροποποιήσαμε ελαφρώς για να μπορεί να επεξεργαστεί πίνακά κόστους με διαφορετικό κόστος ανά στιγμιότυπο.

- public Vector <Character> predict(): η μέθοδος αυτή εφαρμόζει το μοντέλο που δημιουργήθηκε από την buildClassifier στο σύνολο ελέγχου. Με αυτόν τον τρόπο πραγματοποιείται η πρόβλεψη των αγώνων ποδοσφαίρου. Αποθηκεύει τις προβλέψεις σε ένα Vector και τις επιστρέφει.

- private void writeClassifier(): η διαδικασία δημιουργίας του μοντέλου ταξινόμησης απαιτεί σχετικά μεγάλο χρόνο. Οπότε για να μην δημιουργούμε το μοντέλο κάθε φορά που τρέχουμε την εφαρμογή, το αποθηκεύουμε στο δίσκο με τη μέθοδο writeClassifier. Θα πρέπει να σημειωθεί ότι το μοντέλο δημιουργείται από την αρχή μόνο όταν ενημερώνουμε τη βάση δεδομένων. Με αυτόν τον τρόπο, κρατάμε τον κατηγοριοποιητή ενήμερο και προσαρμοσμένο στα δεδομένα της βάσης.

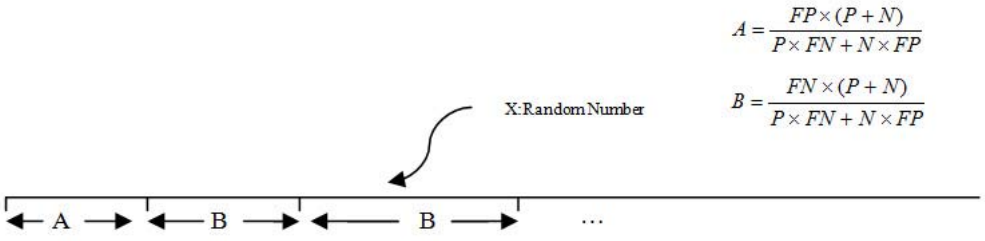
- private void readClassifier(): εκτελεί την αντίστροφη διαδικασία από την witeClassifier. Κάθε φορά που ζητάει ο χρήστης από την εφαρμογή να εκτελέσει πρόβλεψη, φορτώνεται το μοντέλο ταξινόμησης από το δίσκο.

3.4.7 Πακέτο MachineLerning.RouletteSampling

Το πακέτο αυτό περιλαμβάνει όλες εκείνες τις κλάσεις, οι οποίες υλοποιούν τον αλγόριθμο CSRoulette που παρουσιάστηκε στο κεφάλαιο 2. Αξίζει να σημειωθεί ότι, αν και αρχικά ο CSRoulette προοριζόταν να χρησιμοποιηθεί στην εφαρμογή μας, τελικά μετά από αρκετά πειράματα προτιμήθηκε ο MetaCost, ο οποίος αποδείχθηκε πιο αποδοτικός. Το πακέτο συνολικά περιλαμβάνει τέσσερις κλάσεις, τις οποίες και παραθέτουμε στη συνέχεια.

3.4.7.1 Κλάση Line

Στο κεφάλαιο 2 αναφέραμε ότι ο CSRoulette ανήκει στην κατηγορία των meta learners, οι οποίοι μεταβάλουν την κατανομής του συνόλου εκπαίδευσης σύμφωνα με το κόστος λάθους πρόβλεψης. Ουσιαστικά πραγματοποιείται δειγματοληψία στο σύνολο εκπαίδευσης και όπως δείξαμε, ο CSRoulette το επιτυγχάνει με τη βοήθεια μιας ευθείας γραμμής την οποία ονομάσαμε sampling line (εικόνα 4.4). Στο σημείο αυτό θα πρέπει να αναφέρουμε μια σημαντική λεπτομέρεια. Στο κεφάλαιο 2 ορίσαμε ως μήκος των τμημάτων της γραμμής, τα κλάσματα A και B όπως φαίνεται στην εικόνα 4.4. Ωστόσο, για την επίλυση του δικού μας προβλήματος, ακολουθήθηκε μια εναλλακτική προσέγγιση κατά την οποία, ως μήκος του κάθε τμήματος ορίζεται η νικητήρια απόδοση του στιγμιότυπου που αντιστοιχεί στο εκάστοτε τμήμα.



Εικόνα 3.4
Sampling line

Όπως γίνεται αντιληπτό, η κλάση Line αναπαριστά την παραπάνω ευθεία. Οι μέθοδοι τις κλάσης είναι:

- public Vector getSegments(): επιστρέφει το σύνολο των τμημάτων (θυμηθείτε ότι το κάθε τμήμα αντιστοιχεί σε ένα στιγμιότυπο του συνόλου εκπαίδευσης) από τα οποία αποτελείται η ευθεία. Τα τμήματα είναι αντικείμενα της κλάσης Segment, και είναι αποθηκευμένα σε μια δομή Vector.

- public float getLineLength(): η μέθοδος αυτή, επιστρέφει το μήκος της ευθείας γραμμής. Ως μήκος της ευθείας ορίζεται το άθροισμα των νικητήριων αποδόσεων όλων των στιγμιότυπων της γραμμής.

- public void addSegment(int instance, float apodosi): η μέθοδος αυτή, επιτρέπει την προσθήκη ενός νέου τμήματος στη γραμμή. Η παράμετρος instance δηλώνει το στιγμιότυπο στο οποίο θα αντιστοιχεί το νέο τμήμα, ενώ η apodosi εκφράζει το μήκος του τμήματος.

- public void removeAllSegments(): πραγματοποιεί τη διαγράφη όλων των τμημάτων που υπάρχουν στη γραμμή μέχρι στιγμής.

- public int findInstance(float number): η μέθοδος αυτή, δέχεται έναν τυχαίο πραγματικό αριθμό, βρίσκει το τμήμα που περιέχει τον αριθμό αυτό και επιστρέφει το στιγμιότυπο του τμήματος.

3.4.7.2 Κλάση Segment

Έχοντας κατανοήσει την κλάση Line, μπορούμε εύκολα να καταλάβουμε ότι τα αντικείμενα της κλάσης Segment αποτελούν τα τμήματα της ευθείας που περιγράψαμε προηγουμένως. Η Segment περιλαμβάνει τις εξής μεθόδους:

- public float getSegmentStart(): επιστρέφει το σημείο αρχικό σημείο του τμήματος.

- public float getSegmentEnd(): επιστρέφει το σημείο τερματισμού του τμήματος.

- public int getInstance(): επιστρέφει το στιγμιότυπο στο οποίο αντιστοιχεί το συγκεκριμένο τμήμα.

- public void setSegmentStart(float segmentStart): θέτει το σημείο αρχής του τμήματος.

- public void setSegmentEnd(float segmentEnd): θέτει το σημείο τερματισμού του τμήματος.

- public void setInstance(int instance): θέτει το στιγμιότυπο στο οποίο θα αντιστοιχεί το συγκεκριμένο τμήμα.

3.4.7.3 Κλάση CPRS

Η κλάση CPRS υλοποιεί την τεχνική *cost proportionate roulette sampling* ακριβώς με τον τρόπο που περιγράψαμε στο κεφάλαιο 2. Αποτελείται από τις παρακάτω μεθόδους.

- public void createLineFromFile(String trainingSet): σχηματίζει μια γραμμή δειγματοληψίας (sampling line), από τα στιγμιότυπα που βρίσκονται στο

αρχείο trainingSet. Στην ουσία διαβάζει ένα προς ένα τα στιγμιότυπα από το αρχείο και δημιουργεί σχηματίζει τα τμήματα της ευθείας.

- public void createLineFromDataSet(Instances trainingSet): παρόμοια μέθοδος με την createLineFromFile, με τη διαφορά ότι δεν διαβάζει τα στιγμιότυπα από το δίσκο, αλλά από τη μνήμη.

- public void createNewTraingSet(int samleSize): η μέθοδος αυτή, σχηματίζει το καινούργιο σύνολο εκπαίδευσης, μετά την εκτέλεση δειγματοληψίας (η δειγματοληψία περιγράφεται αναλυτικά στο κεφάλαιο 2.3).

3.4.8 Πακέτο GUI

Τέλος, μας έμεινε να περιγράψουμε την διασύνδεση του χρήστη με την εφαρμογή (εικόνα 4.1). Το πακέτο GUI, αποτελείται από τις κλάσεις που σχηματίζουν την γραφική διασύνδεση του χρήστη με το σύστημα. Η υλοποίηση έγινε με τη βοήθεια των βιβλιοθηκών Swing, OpenSwing, Liquidlnf, SwingX και miglayout.

3.4.8.1 Κλάση Splash

Κληρονομεί την κλάση javax.swing.JWindow και στη πραγματικότητα, αποτελεί το παράθυρο που εμφανίζεται κατά την εκκίνηση της εφαρμογή.

3.4.8.2 Κλάση GamesTabManipulator

Όπως θα δείξουμε στο επόμενο κεφάλαιο, η εφαρμογή περιλαμβάνει δύο καρτέλες (tabs). Η πρώτη δίνει τη δυνατότητα στο χρήστη να δει ορισμένα στατιστικά στοιχεία που αφορούν τις ομάδες και τα πρωταθλήματα, ενώ η δεύτερη παρουσιάζει τις προβλέψεις για το πιο πρόσφατο κουπόνι του ΟΠΑΠ. Η κλάση GamesTabManipulator περιλαμβάνει όλες τις απαραίτητες μεθόδους για την πρώτη καρτέλα.

- public DefaultTableModel getTableModel(String date, String league, String team): σχηματίζει το μοντέλο του πίνακα org.jdesktop.swingxJXTable, δηλαδή το περιεχόμενο του το οποίο αφορά αγώνες ποδοσφαίρου. Τα πεδία του πίνακα είναι τα εξής:
    1. League: όνομα της διοργάνωση στην οποία διεξάγεται ο αγώνας.
    2. Date: ημερομηνία διεξαγωγής του αγώνα.

3. Apodosi1: η απόδοση του σημείου 1.
4. HomeTeam: η γηπεδούχος ομάδα του αγώνα.
5. ApodosiX: η απόδοση του σημείου X.
6. AwayTeam: η φιλοξενούμενη ομάδα του αγώνα.
7. Apodosi2: η απόδοση του σημείου 2.
8. Final_HomeTeamScore: τα τέρματα που πέτυχε η γηπεδούχος ομάδα στο συγκεκριμένο αγώνα.
9. Final_AwayTeamScore: τα τέρματα που πέτυχε η φιλοξενούμενη ομάδα στο συγκεκριμένο αγώνα.

Οι παράμετροι της μεθόδου καθορίζουν τους αγώνες που θα εμφανιστούν. Δηλαδή, από πια ημερομηνία και μετά, για πιο πρωτάθλημα και για πια ομάδα θα εμφανίσει τους αγώνες.

- private void setStatisticsForGamesTab(String date, String league, String team): εμφανίζει τα στατιστικά στοιχεία που υπολόγισε η μέθοδος computeStatistics της κλάσης Statistic.

- private void showHighestScore(String date, String league, String team): εμφανίζει τον αγώνα στον οποίο σημειώθηκαν τα περισσότερα τέρματα. Οι παράμετροι έχουν την ίδια χρήση με τη μέθοδο getTableModel.

- public DefaultComboBoxModel getLeagues(): σχηματίζει το μοντέλο για το αντικείμενο της κλάσης javax.swing.JComboBox. Ουσιαστικά καθορίζει τις δυνατές επιλογές του JComboBox, οι οποίες είναι όλα τα πρωταθλήματα για τα οποία έχουμε αποθηκευμένους αγώνες στη βάση δεδομένων μας.

- public DefaultComboBoxModel getTeams(String date, String league): παρόμοια με την getLeagues, με τη διαφορά ότι οι επιλογές αφορούν τις ομάδες ενός πρωταθλήματος.

- public void dateChooserOnSelectionChange(): αποτελεί έναν *event handler* που ενεργοποιείται όταν ο χρήστης επιλέξει μια ημερομηνία από το DateChooserCombo (περιγράφεται στο επόμενο κεφάλαιο).

- public void leagueChooserCBActionPerformed(): event handler που ενεργοποιειται όταν ο χρήστης επιλέξει ένα πρωτάθλημα από το αντίστοιχο JComboBox.

- public void teamChooserCBActionPerformed(): ενεργοποιείται όταν ο χρήστης επιλέξει μια ομάδα από το αντίστοιχο JComboBox.

3.4.8.3 Κλάση PredictionTabManipulator

Η κλάση PredictionTabManipulator περιλαμβάνει όλες τις απαραίτητες μεθόδους για το tab που είναι υπεύθυνο για την παρουσιάσει των προβλέψεων. Συγκεκριμένα, περιλαμβάνει τις εξής μεθόδους:

- public void setPredictions(Vector predictions): αποθηκεύει τις προβλέψεις που έγιναν από τη μέθοδο predict() της κλάσης Predictor.

- public void updateButtonMouseClicked(): η μέθοδος αυτή εκτελείται όταν ο χρήστης πατήσει το κουμπί για την ανανέωση της βάσης δεδομένων. Η ανανέωση πραγματοποιείται με τη βοήθεια της μεθόδου updateTables της κλάσης DOMParser. Τέλος, κάθε φορά που γίνεται η ανανέωση, πραγματοποιείται ταυτόχρονα και η δημιουργία του μοντέλου ταξινόμησης.

- public DefaultComboBoxModel getLeaguesOfLastCoupon(): παρόμοια μέθοδος με την getLeagues της κλάσης GamesTabManipulator με τη διαφορά ότι εμφανίζει μόνο τα πρωταθλήματα για τα οποία υπάρχουν αγώνες στο τελευταίο κουπόνι στοιχήματος του ΟΠΑΠ.

- public void leagueChooserCB2ActionPerformed(): η μέθοδος αυτή εκτελείται κάθε φορά που ο χρήστης επιλέγει ένα πρωτάθλημα από το αντίστοιχο JComboBox.

- public DefaultTableModel getGamesToBet(): σχηματίζει το μοντέλο του πίνακα που παρουσιάζει τους αγώνες που βρίσκονται στο πιο πρόσφατο κουπόνι στοιχήματος του ΟΠΑΠ. Τα πεδία του πίνακα είναι:

  1. League
  2. Date
  3. Apodosi1
  4. HomeTeam
  5. ApodosiX
  6. AwayTeam
  7. Apodosi2
  8. Prediction

  Η τιμή του πεδίου prediction σε αυτήν τη φάση είναι ο χαρακτήρας ‘?’ για όλες τις εγγραφές.

- public void predictButtonMouseClicked(): εκτελείται όταν ο χρήστης πατήσει το κουμπί της πρόβλεψης. Η πρόβλεψη πραγματοποιείται με τη βοήθεια της μεθόδου predict() της κλάσης Predictor.
- public DefaultTableModel getPredictionTableModel(): παρόμοια με τη μέθοδο getGamesToBet(), με τη διαφορά ότι το πεδίο prediction πλέον περιλαμβάνει τις πραγματικές προβλέψεις για τους αγώνες και όχι το χαρακτήρα '?'.

3.4.8.4 Κλάση MessageStore

Είναι χρήσιμο για έναν χρήστη να γνωρίζει την κατάσταση στην οποία βρίσκεται η εφαρμογή σε κάθε στιγμή. Για παράδειγμα, στην εφαρμογή μας, όταν ο χρήστης ζητήσει την εκτέλεση της ενημέρωσης της βάσης δεδομένων, θέλει να γνωρίζει σε πια φάση βρίσκεται η διαδικασία της ενημέρωσης, δηλαδή έχει ολοκληρωθεί η πραγματοποιείται ακόμη. Για να το πετύχουμε αυτό στην εφαρμογή μας, εμφανίζουμε μηνύματα που δείχνουν την κατάσταση εκτέλεσης των διάφορων λειτουργιών της. Η κλάση MessageStore αντιπροσωπεύει μια «αποθήκη» η οποία κρατάει ένα μήνυμα κάθε φορά.

- public void setMessage(String status): προσθέτει ένα νέο μήνυμα στην «αποθήκη» των μηνυμάτων.
- public boolean newMessageCame(): η μέθοδος αυτή δηλώνει ότι έχει προστεθεί ένα νέο μήνυμα στην αποθήκη.
- public String getStatus(): εμφανίζει το μήνυμα που έχει αποθηκευτεί τελευταία.
- public void noMoreNewMessage(): δηλώνει ότι έχει διαβαστεί το μήνυμα από τον statusBarChanger (βλέπε παρακάτω).

3.4.8.5 Κλάση StatusBarChanger

Η κλάση StatusBarChanger, υλοποιεί τον μηχανισμό των μηνυμάτων που περιγράψαμε προηγουμένως. Η υλοποίηση της ακολουθεί τον τρόπο με τον οποίο επικοινωνούν οι πράκτορες σε ένα πολυπρακτορικό σύστημα. Δηλαδή, η StatusBarChanger ελέγχει μόνη της εάν υπάρχει νέο μήνυμα, το οποίο θα πρέπει να παρουσιάσει στο χρήστη. Πιο απλά, η MessageStore, δεν απασχολεί την

StatusBarChanger κάθε φορά που έρχεται ένα νέο μήνυμα. Για να επιτευχθεί αυτό, η StatusBarChanger θα πρέπει να εκτελείται ως ανεξάρτητο νήμα.

3.4.8.6 Κλάση Updater

Τέλος, η κλάση Updater είναι υπεύθυνη για την πραγματοποίηση της ενημέρωσης της βάσης δεδομένων. Η ενημέρωση πραγματοποιείται με τη βοήθεια της μεθόδου updateTables της κλάσης DOMParser. Για να αυξήσουμε την απόδοση του συστήματος, η κλάση Updater εκτελείτε ως ανεξάρτητο νήμα. Όταν ολοκληρωθεί η ενημέρωση, ο χρήστης ενημερώνεται άμεσα με τον τρόπο που περιγράψαμε προηγουμένως.

ΚΕΦΑΛΑΙΟ 4: Η ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΠΡΑΞΗ

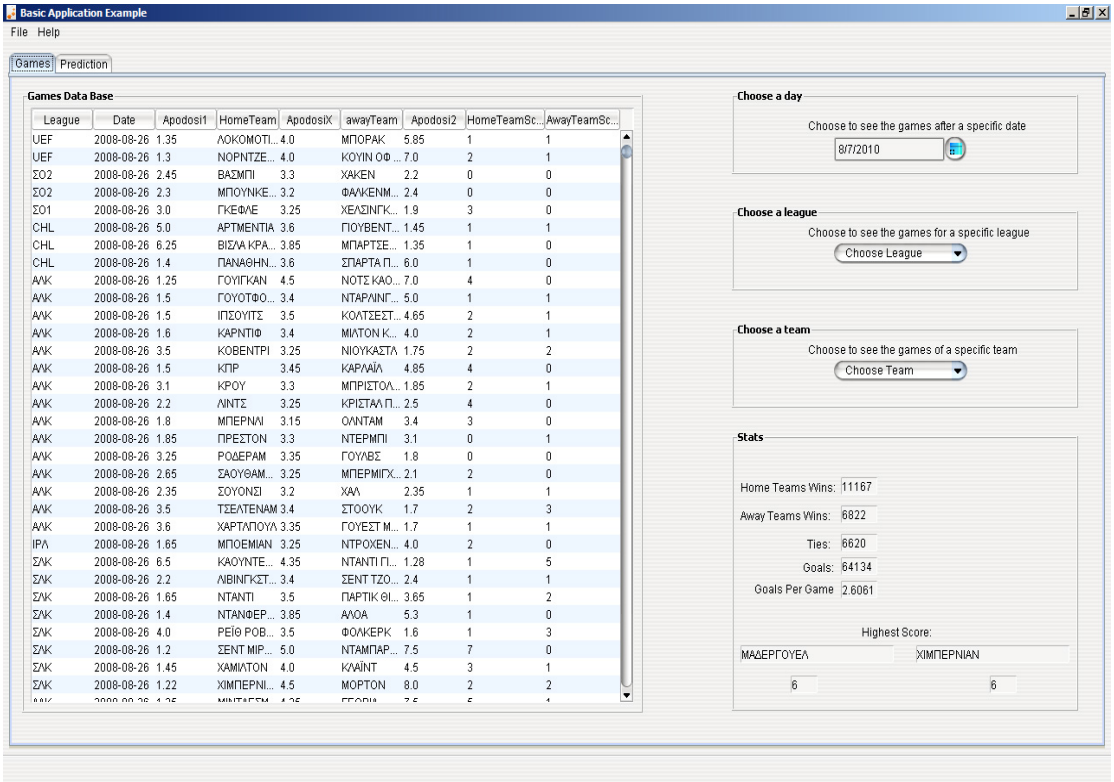
ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΠΡΑΞΗ

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα παρουσιάσουμε ορισμένα στιγμιότυπα από τη λειτουργία της εφαρμογής. Ουσιαστικά θα δείξουμε τα αποτελέσματα της εκτέλεσης του κώδικα που περιγράφτηκε στο προηγούμενο κεφάλαιο. Στις γραμμές που ακολουθούν θα δείξουμε τις λειτουργίες που ενσωματώνουν οι δυο καρτέλες της εφαρμογής.

4.1 ΚΑΡΤΕΛΑ GAMES

Η καρτέλα *Games* παρουσιάζει τα περιεχόμενα της βάσης δεδομένων. Δηλαδή, αποτελείται από έναν πίνακα ο οποίος συγκεντρώνει τα αποτελέσματα των αγώνων ποδοσφαίρου, που είναι αποθηκευμένοι στη βάση. Ο χρήστης έχει τη δυνατότητα για κάθε αγώνα να δει πληροφορίες όπως ομάδες, αποδόσεις καθώς και τέρματα των ομάδων.



Εικόνα 4.1
Καρτέλα Games

Στην εικόνα 4.1 παρουσιάζεται η καρτέλα *Games*. Όπως είπαμε, υπάρχει ένας πίνακας που έχει ως εγγραφές τους αγώνες ποδοσφαίρου της βάσης δεδομένων. Ο πίνακας δίνει τη δυνατότητα τις γρήγορης ταξινόμησης των εγγραφών με βάση κάποιο πεδίο.

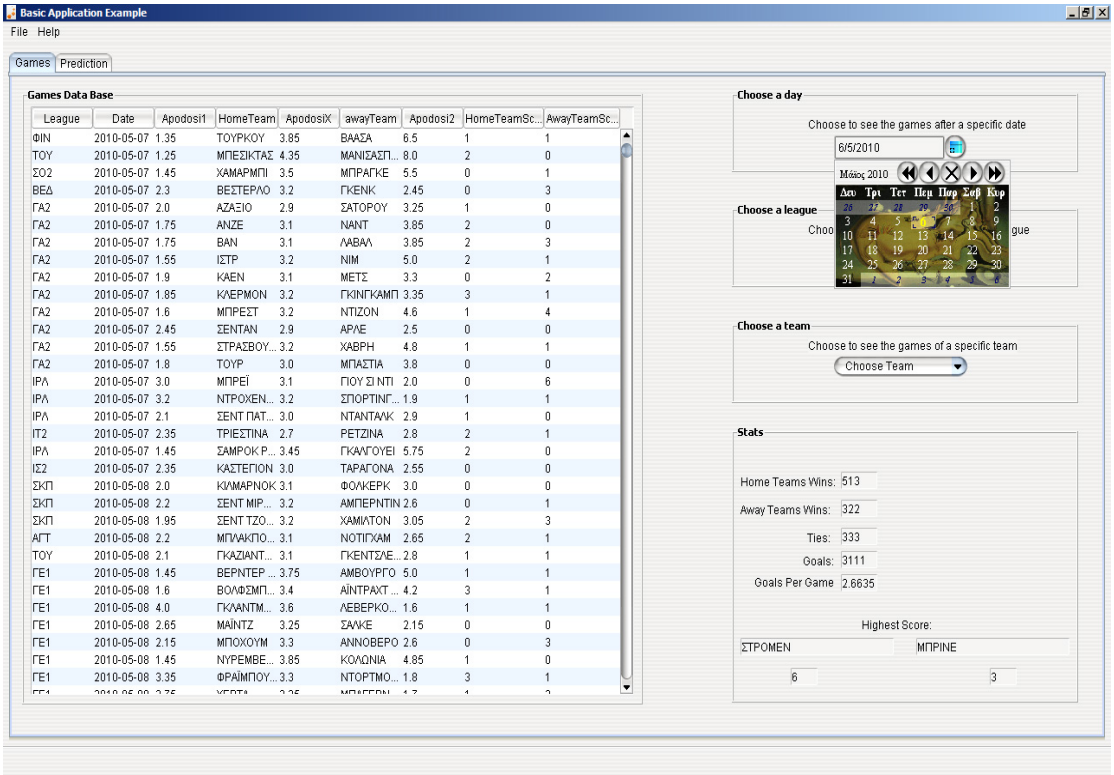
Επίσης, η καρτέλα περιλαμβάνει:

- Ένα ημερολόγιο που δίνει τη δυνατότητα στο χρήστη να επιλέξει να δει τους αγώνες που έλαβαν χώρα από μια συγκεκριμένη ημερομηνία και μετά (εικόνα 4.2).

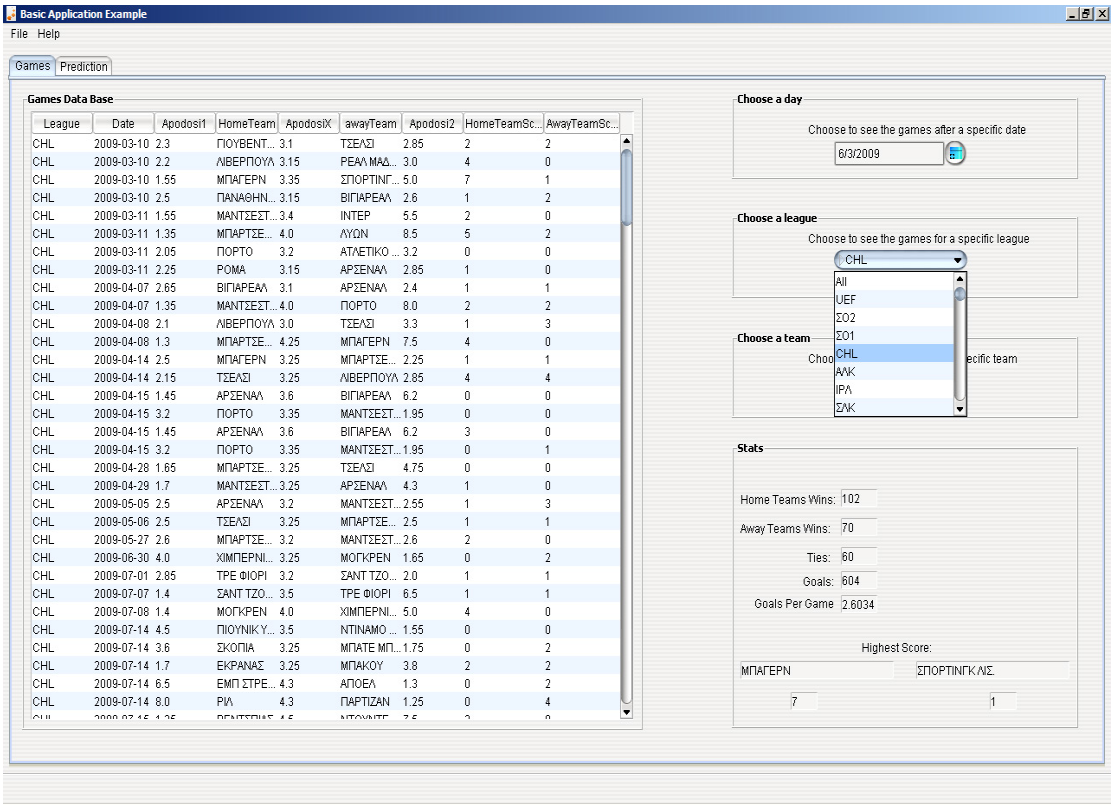
- Μια αναδιπλούμενη λίστα, από την οποία ο χρήστης μπορεί να επιλέξει τους αγώνες μιας συγκεκριμένης διοργάνωσης (εικόνα 4.3).

- Μια αναδιπλουμένη λίστα, από την οποία ο χρήστης μπορεί να επιλέξει αγώνες μιας συγκεκριμένης ομάδας (εικόνα 4.4).

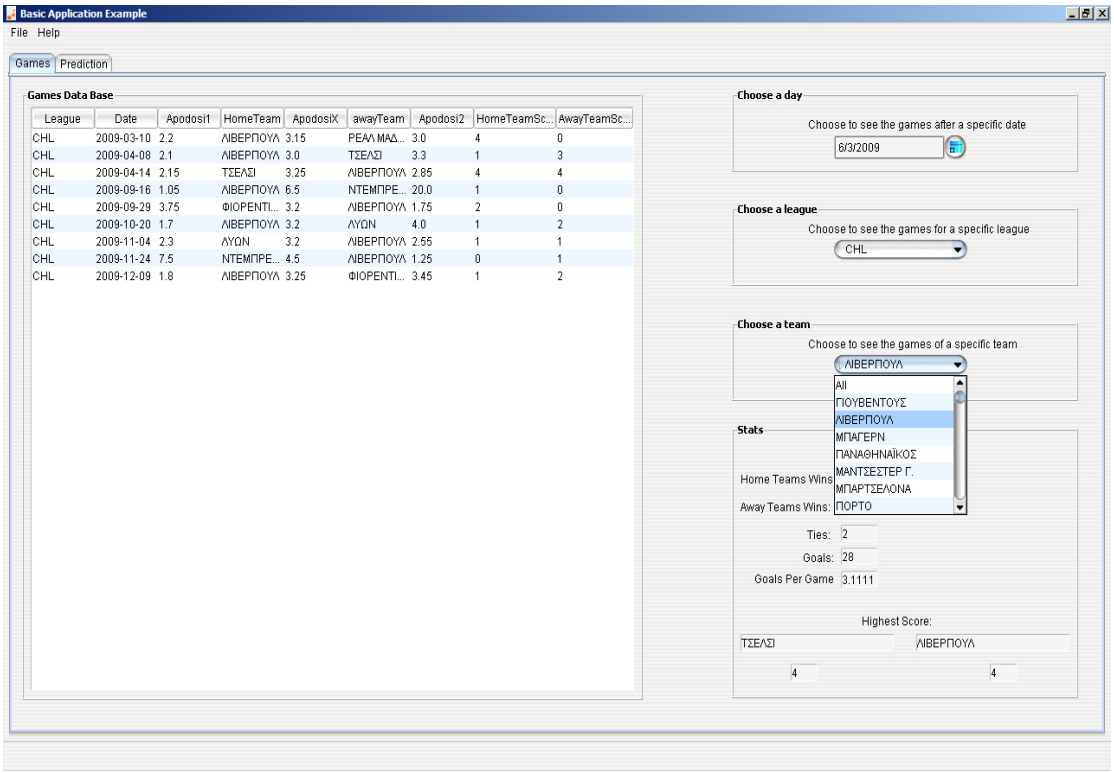
Τέλος, υπάρχει μια περιοχή που περιλαμβάνει ορισμένα στατιστικά στοιχεία (βλέπε κεφάλαιο 3) για τους αγώνες που παρουσιάζονται από τον πίνακα.



Εικόνα 4.2
Επιλογή ημερομηνίας



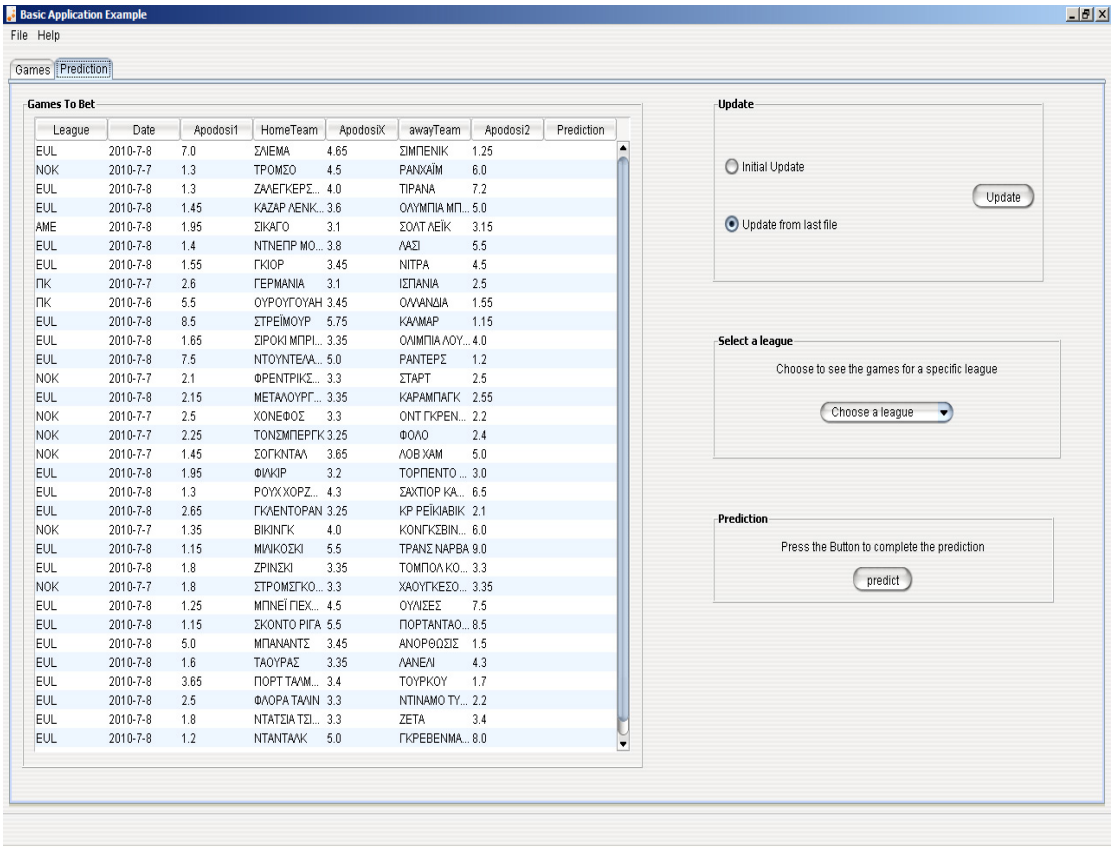
Εικόνα 4.3
Επιλογή πρωταθλήματος



Εικόνα 4.4
Επιλογή ομάδας

4.2 ΚΑΡΤΕΛΑ PREDICTION

Η καρτέλα prediction παρουσιάζει στον χρήστη τις προβλέψεις που αφορούν τους αγώνες που βρίσκονται στο πιο πρόσφατο κουπόνι του ΟΠΑΠ. Το κουπόνι αυτό το διαβάζει η εφαρμογή κατά την εκκίνηση του προγράμματος και το παρουσιάζει σε έναν πίνακα.



Εικόνα 4.5
Καρτέλα prediction πριν πραγματοποιηθεί η πρόβλεψη

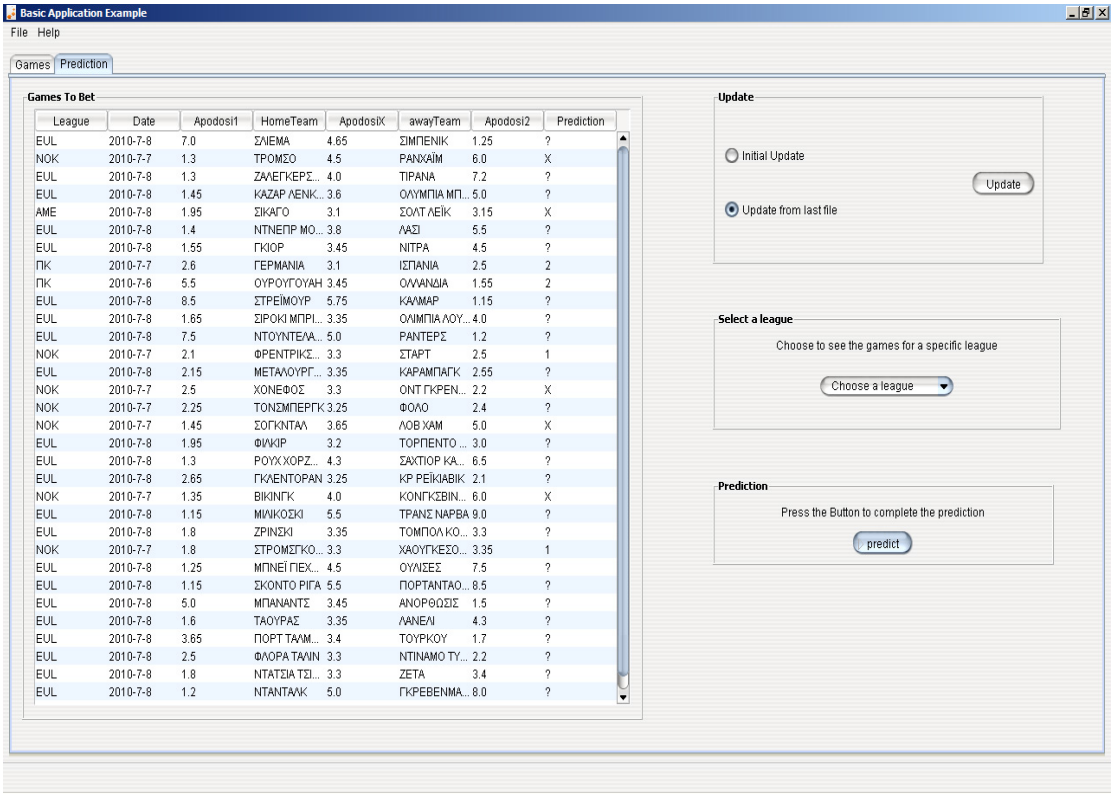
Όπως παρατηρείται, ο πίνακας είναι παρόμοιος με αυτόν που βρίσκεται στην καρτέλα games, με τη διαφορά ότι πλέον δεν υπάρχουν τα πεδία με τα τέρματα των ομάδων, μια και οι αγώνες δεν έχουν πραγματοποιηθεί ακόμη, και υπάρχει το πεδίο prediction, το οποίο στην εικόνα 4.5 είναι κενό.

Η καρτέλα προσφέρει τις εξής λειτουργίες:

- Ενημέρωση της βάσης δεδομένων. Η ενημέρωση μπορεί να γίνει είτε διαβάζοντας τα πρόσφατα κουπόνια (αρχεία XML) ή διαβάζοντας όλα τα κουπόνια που προσφέρει ο ΟΠΑΠ. Στη δεύτερη περίπτωση, ο πίνακας της βάσης δεδομένων αδειάζει και ξαναγεμίζει από την αρχή.

- Εκτέλεση της πρόβλεψης. Ουσιαστικά πραγματοποιείται η πρόβλεψη για το αποτέλεσμα του αγώνα.

- Προβολή των αγώνων ενός συγκεκριμένου πρωταθλήματος. Ο χρήστης μπορεί να επιλέξει να δει τους αγώνες που θα γίνουν στα πλαίσια μιας συγκεκριμένης διοργάνωσης.



Εικόνα 4.6
Η καρτέλα prediction μετά την πραγματοποίηση της πρόβλεψης

Η εικόνα 4.6 παρουσιάζει την κατάσταση του πίνακα, μετά από την πραγματοποίηση της πρόβλεψης. Όπως βλέπετε, το πεδίο prediction έχει πλέον μια τιμή εκ των 1, X, 2 ή ?. Το ? δηλώνει ότι δεν υπήρχαν αρκετές πληροφορίες για να προβλευθεί το συγκεκριμένο παιχνίδι.

ΠΑΡΑΡΤΗΜΑ Ι: ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο παράρτημα αυτό, παραθέτουμε τα αποτελέσματα της εκτέλεσης των αλγορίθμων μάθησης. Πιο συγκεκριμένα των αλγορίθμων MetaCost (βλέπε υποκεφάλαιο 2.3.2.1), CSRoulette (βλέπε υποκεφάλαιο 2.3.2.2) καθώς και του αλγορίθμου CostSensitiveClassifier (ο οποίος βασίζεται μόνο στο μικρότερο αναμενόμενο κόστος όπως περιγράφεται στο υποκεφάλαιο 2.3.1.1) που υπάρχει στο εργαλείο weka. Η εκτέλεση των πειραμάτων κρίθηκε αναγκαίο αφενός για να επιλέξουμε των πλέον αποδοτικό αλγόριθμο για την εφαρμογή μας και αφετέρου να δούμε τις διαφορές στην απόδοση των δύο αυτών αλγορίθμων. Χρησιμοποιήσαμε 14 διαφορετικούς cost-insensitive αλγορίθμους ως base classifiers για τον MetaCost και CSRoulette, καθώς επίσης μετρήσαμε την απόδοση αυτών χωρίς τη χρήση της cost-sensitive μάθησης .

Για τους πειραματικούς σκοπούς, χρησιμοποιήθηκαν 5 διαφορετικά σύνολα εκπαίδευσης (training set).

1) Statistic.arff : τα χαρακτηριστικά του συνόλου αυτού περιγράφονται στο υποκεφάλαιο 3.2.2.
2) StatisticWithNames.arff : είναι ίδιο με το σύνολο statistics.arff με δύο επιπλέον χαρακτηριστικά, που είναι οι δυο ομάδες του εκάστοτε αγώνα σε μορφή ακεραίου.
3) Similar.arff : τα χαρακτηριστικά του συνόλου αυτού είναι ίδια με το statistics.arff με τη διαφορά ότι οι τιμές τους υπολογίζονται συμφωνά με τα αποτελέσματα που είχε η κάθε ομάδα με ομάδες όμοιας δυναμικής με την αντίπαλη ομάδα. Το κριτήριο καθορισμού της ομοιότητας είναι η απόδοση.
4) SimilarWithNames.arff : ίδιο με το similar.arff συμπεριλαμβανομένου όμως και των δύο ομάδων.
5) UnderOver.arff : ο σκοπός του συνόλου εκπαίδευσης αυτού, είναι η κατασκευή ενός κατηγοριοποιητή για την πρόβλεψη των σημείων under και over ενός αγώνα ποδοσφαίρου.

STATISTIC.ARFF

o MetaCost

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	44	90
J48	41	62
J48GRAFT	41	67
LADTREE	42	-5

REPTREE	44	11
SIMPLECART	45	60
NAIVEBAYES	43	34
NAIVEBAYESMULTINOMIAL	48	3
NAIVEBAYESSIMPLE	44	22
LOGISTIC	45	-67
SMO	43	-61
IB1	38	81
DECITIONTABLE	37	30

- **CostSensitiveClassifier**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	38	115
J48	39	97
J48GRAFT	40	93
LADTREE	40	7
REPTREE	39	84
SIMPLECART	40	45
NAIVEBAYES	44	25
NAIVEBAYESMULTINOMIAL	46	-38
NAIVEBAYESSIMPLE	44	28
LOGISTIC	43	-31
SMO	42	-48
IB1	39	84
DECITIONTABLE	35	53

- **CSRoulette**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	40	53
J48	40	61
J48GRAFT	40	66
LADTREE	41	4
REPTREE	41	22
SIMPLECART	41	27
NAIVEBAYES	42	20
NAIVEBAYESMULTINOMIAL	45	-46
NAIVEBAYESSIMPLE	42	12
LOGISTIC	44	-29
SMO	45	-55
IB1	37	109
DECITIONTABLE	43	-50

- **No meta learners**

Ακρίβεια των αλγορίθμων χωρίς τη χρήση cost-sensitive αλγορίθμων.

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	50	87
J48	42	67
J48GRAFT	42	64
LADTREE	48	30
REPTREE	50	42
SIMPLECART	50	84
NAIVEBAYES	48	8
NAIVEBAYESMULTINOMIAL	35	132

NAIVEBAYESSIMPLE	48	0
LOGISTIC	51	19
SMO	49	85
IB1	52	0
DECITIONTABLE	38	125

STATISTICWITHNAMES.ARFF

- **MetaCost**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	42	123
J48	37	154
J48GRAFT	37	156
LADTREE	42	36
REPTREE	41	36
SIMPLECART	47	47
NAIVEBAYES	44	24
NAIVEBAYESMULTINOMIAL	34	124
NAIVEBAYESSIMPLE	44	23
LOGISTIC	43	-37
SMO	42	-56
IB1	36	140
DECITIONTABLE	38	23

- **CostSensitiveClassifier**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	39	87
J48	41	66
J48GRAFT	41	62

LADTREE	39	21
REPTREE	40	58
SIMPLECART	39	63
NAIVEBAYES	43	31
NAIVEBAYESMULTINOMIAL	31	127
NAIVEBAYESSIMPLE	44	23
LOGISTIC	42	44
SMO	41	-17
IB1	38	124
DECITIONTABLE	35	53

- **CSRoulette**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	38	137
J48	36	122
J48GRAFT	36	122
LADTREE	43	104
REPTREE	40	112
SIMPLECART	37	116
NAIVEBAYES	36	106
NAIVEBAYESMULTINOMIAL	38	137
NAIVEBAYESSIMPLE	42	15
LOGISTIC	42	3
SMO	45	-46
IB1	37	132
DECITIONTABLE	42	-34

SIMILAR.ARFF

o MetaCost

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	31	32
J48	35	23
J48GRAFT	35	23
LADTREE	35	20
REPTREE	31	18
SIMPLECART	30	40
NAIVEBAYES	39	-7
NAIVEBAYESMULTINOMIAL	33	2
NAIVEBAYESSIMPLE	39	-7
LOGISTIC	40	-6
SMO	37	-5
IB1	35	17
DECITIONTABLE	30	15

o CostSensitiveClassifier

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	34	5
J48	35	19
J48GRAFT	35	19
LADTREE	32	25
REPTREE	35	-1
SIMPLECART	32	4

NAIVEBAYES	39	-4
NAIVEBAYESMULTINOMIAL	34	5
NAIVEBAYESSIMPLE	40	-8
LOGISTIC	39	-1
SMO	34	9
IB1	35	19
DECITIONTABLE	29	6

- **CSRoulette**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	31	31
J48	39	8
J48GRAFT	35	21
LADTREE	33	23
REPTREE	36	14
SIMPLECART	26	24
NAIVEBAYES	36	1
NAIVEBAYESMULTINOMIAL	34	12
NAIVEBAYESSIMPLE	35	4
LOGISTIC	37	4
SMO	38	-2
IB1	37	11
DECITIONTABLE	35	6

SIMILARWITHNAMES.ARFF

- **MetaCost**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	39	9
J48	34	13
J48GRAFT	34	15
LADTREE	28	41
REPTREE	31	23
SIMPLECART	37	18
NAIVEBAYES	38	-3
NAIVEBAYESMULTINOMIAL	33	12
NAIVEBAYESSIMPLE	39	-5
LOGISTIC	34	12
SMO	32	19
IB1	34	11
DECITIONTABLE	30	15

- **CostSensitiveClassifier**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	34	14
J48	34	21
J48GRAFT	34	21
LADTREE	32	27
REPTREE	34	12
SIMPLECART	31	12
NAIVEBAYES	38	-3
NAIVEBAYESMULTINOMIAL	31	19

NAIVEBAYESSIMPLE	40	-8
LOGISTIC	31	25
SMO	34	6
IB1	35	15
DECITIONTABLE	29	6

- **CSRoulette**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	32	29
J48	34	23
J48GRAFT	34	22
LADTREE	33	23
REPTREE	31	26
SIMPLECART	31	32
NAIVEBAYES	35	4
NAIVEBAYESMULTINOMIAL	33	18
NAIVEBAYESSIMPLE	33	10
LOGISTIC	31	23
SMO	36	3
IB1	31	23
DECITIONTABLE	34	19

UNDEROVER.ARFF

- **MetaCost**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	51	433
J48	51	427
J48GRAFT	51	428

LADTREE	52	404
REPTREE	50	442
SIMPLECART	53	396
NAIVEBAYES	49	450
NAIVEBAYESMULTINOMIAL	51	400
NAIVEBAYESSIMPLE	48	453
LOGISTIC	52	414
SMO	52	382
IB1	52	411
DECITIONTABLE	50	421

- **CostSensitiveClassifier**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	55	379
J48	51	406
J48GRAFT	51	406
LADTREE	51	418
REPTREE	56	359
SIMPLECART	56	365
NAIVEBAYES	50	429
NAIVEBAYESMULTINOMIAL	52	407
NAIVEBAYESSIMPLE	49	441
LOGISTIC	52	418
SMO	51	427
IB1	52	416
DECITIONTABLE	50	416

- **CSRoulette**

BASECLASSIFIER	CORRECTLY CLASSIFIED %	TOTAL COST
BFTREE	48	432
J48	48	421
J48GRAFT	47	424
LADTREE	49	408
REPTREE	49	418
SIMPLECART	49	410
NAIVEBAYES	50	405
NAIVEBAYESMULTINOMIAL	48	418
NAIVEBAYESSIMPLE	50	406
LOGISTIC	49	416
SMO	49	415
IB1	48	419
DECITIONTABLE	49	421

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Τα σύνολα εκπαίδευσης: underover.arff, statistic.arff, statisticWithNames.arff αποτελούνται από 801 στιγμιότυπα, ενώ τα υπόλοιπα 2 σύνολα εκπαίδευσης αποτελούνται από 121.

- Όπως παρατηρούμε, ο αλγόριθμος SMO έχει σχεδόν πάντα τη καλύτερη απόδοση και το καλύτερο κόστος σε σχέση με τα δένδρα απόφασης (BFTree, LADTree κ.τ.λ.). Ο λόγος είναι, η ικανότητα του SMO να υπολογίζει τις υπό-συνθήκη πιθανότητες που απαιτούνται για τον υπολογισμό του *μικρότερου αναμενόμενου κόστους* (βλέπε 2.3.1.1).

ΠΑΡΑΡΤΗΜΑ ΙΙ: ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Jiawei Han, Micheline Kamber (2006): “Data Mining: Concepts and Techniques- second edition”

[2] Bianca Zadrozny, Charles Elkan:”Learning and Making Decisions When Costs and Probabilities are Both Unknown”, In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining 2001.

[3] Pedro Domingos: “MetaCost: A General Method for Making Classifiers Cost-Sensitive”, In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining 1999.

[4] Bianca Zadrozny , John Langford , Naoki Abe: “Cost-Sensitive Learning by Cost-Proportionate Example Weighting”, Proceedings of the Third IEEE International Conference on Data Mining, page 435, 2003.

[5] Charles Elkan: “The Foundation of Cost-Sensitive Learning”, Proceedings of the 17th international joint conference on Artificial intelligence – Volume 2**,** pages 973-978, 2001.

[6] Victor S. Sheng, Charles X. Ling : “CSRoulette Sampling for Cost-Sensitive Learning”, proceedings of the 18th European conference on Machine Learning, pages 724-731, 2007

[7] Αλέξανδρος Νανόπουλος, Ιωάννης Μανωλόπουλος: “Εισαγωγή στην Εξόρυξη και τις Αποθήκες Δεδομένων ”, Εκδόσεις Νέων Τεχνολογιών 2008.

[8] Βλαχάβας Ιωάννης, Τσουμάκας Γρηγόριος : “Θεωρία και Συστήματα Λήψης Απόφασης”, Πανεπιστημιακές σημειώσεις.

[9] Daniel T.Larose : “Discovering Knowledge in Data: An Introduction to Data Mining”, John Wiley & Sons, 2005.

[10] John Wang : “Encyclopedia of Data Warehousing and Mining – second edition”, page 279, Idea Group Publishing, 2009.

[11] J. Ross Quinlan : “C4.5 : programs for Machine Learning”, Morgan Kaufmann Publishers, 1993.

[12] Vipin Kumar“Top Ten Algorithms in Data Mining”, Chapman and Hall/CRC, 2009.

[13] Geo_rey Holmes, Bernhard Pfahringer, Richard Kirkby, Eibe Frank and Mark Hall : “Multiclass Alternative Decision Trees”.

[14] "Cost –Sensitive Learning"

[15] Haijian Shi : "Best-First Decision Tree Learning", 2006.

[16] Ian H. Witten, Eibe Frank : "Data Mining Practical machine Learning Tools and Techniques – second edition", Morgan Kaufmann, 2005.

[17] Victor Sheng, Charles Ling : "Thresholding for Making Classifier Cost-Sensitive", Proceedings of the 21st national conference on Artificial intelligence – Volume, pages 476-481, 2006.

[18] Βλαχάβας Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας (2005) «Τεχνητή Νοημοσύνη – 2η Έκδοση». Εκδόσεις Γαρταγάνη.