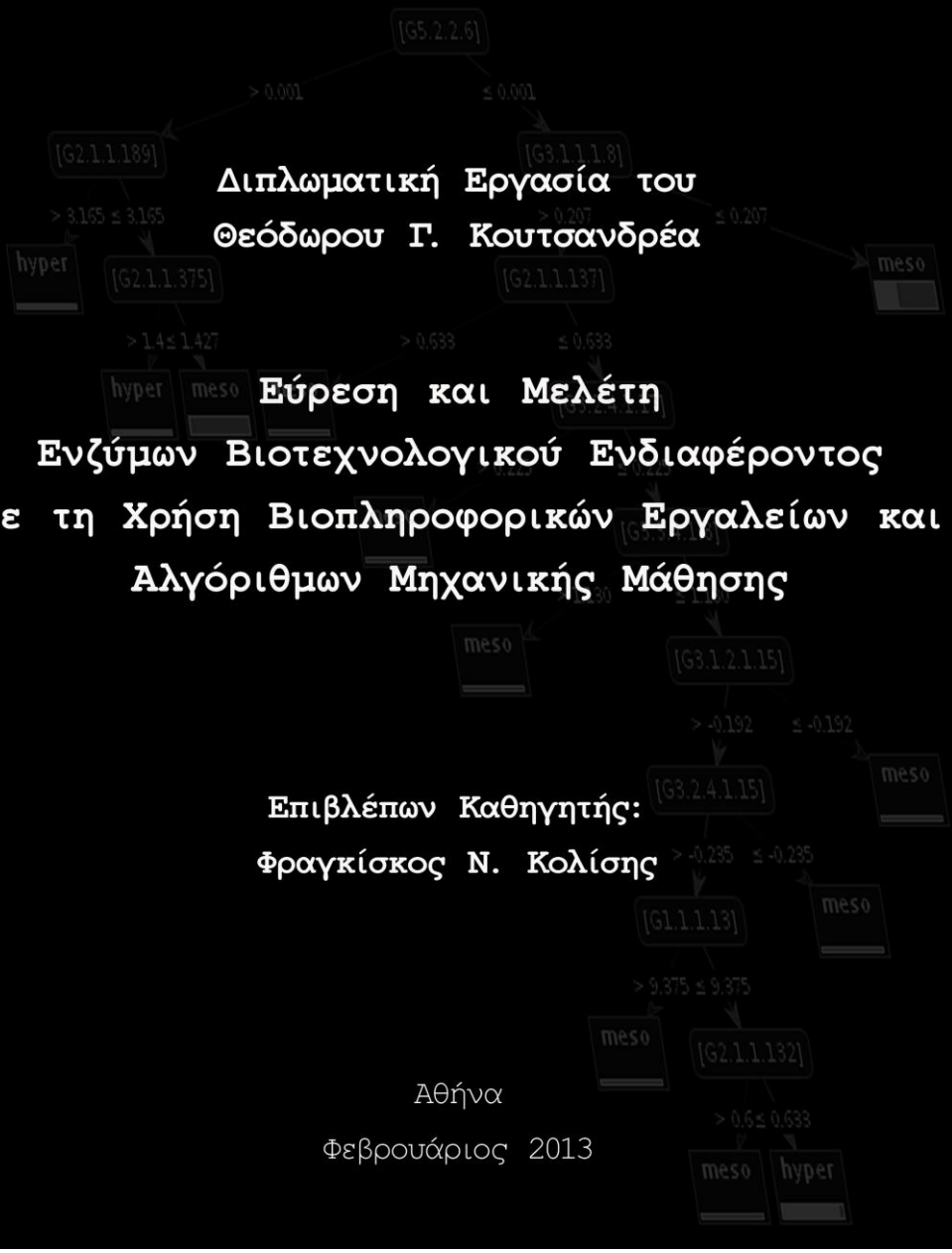
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ IV: ΣΥΝΘΕΣΗΣ ΚΑΙ ΑΝΑΠΤΥΞΗΣ
ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ



Διπλωματική Εργασία του
Θεόδωρου Γ. Κουτσανδρέα

Εύρεση και Μελέτη Ενζύμων Βιοτεχνολογικού Ενδιαφέροντος με τη Χρήση Βιοπληροφορικών Εργαλείων και Αλγόριθμων Μηχανικής Μάθησης

Επιβλέπων Καθηγητής:
Φραγκίσκος Ν. Κολίσης

Αθήνα
Φεβρουάριος 2013

Στους γονείς μου για την προσφορά τους

Και σε όσους πιστεύουν πως η γνώση
οδηγεί στην ελευθερία…

ΕΥΧΑΡΙΣΤΙΕΣ

Η εκπόνηση της διπλωματικής εργασίας ήταν η πιο ενδιαφέρουσα περίοδος στον πενταετή κύκλο σπουδών. Για τις γνώσεις που κατέκτησα, θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή της Σχολής Χημικών Μηχανικών κ. Φραγκίσκο Κολίση και τον υποψήφιο διδάκτορα Ευθύμιο Λαδουκάκη για την ευκαιρία που μου έδωσαν να έρθω σε επαφή με τον κόσμο της Βιοπληροφορικής. Το τεχνικό μέρος της εργασίας αναπτύχθηκε στο εργαστήριο Μεταβολικής Μηχανικής και Βιοπληροφορικής του Εθνικού Ιδρύματος Ερευνών. Θα ήθελα να ευχαριστήσω τον Δρ. Αριστοτέλη Χατζηιωάννου, Ερευνητή Γ' στο Ινστιτούτο Βιολογίας, Φαρμακευτικής Χημείας και Βιοτεχνολογίας του Εθνικού Ιδρύματος Ερευνών, για την δυνατότητα που μου έδωσε να βρίσκομαι στο χώρο του εργαστηρίου για πάνω από ένα εξάμηνο, καθώς και για τις συμβουλές που μου έδωσε όχι μόνο για τη διπλωματική εργασία αλλά και για τη μετέπειτα πορεία μου ως μηχανικός. Τέλος θα ήθελα να ευχαριστήσω θερμά τον Δρ. Ελευθέριο Πιλάλη για την πολύτιμη βοήθεια και τις συμβουλές που μου πρόσφερε κατά την διάρκεια εκπόνησης της εργασίας και για το ευχάριστο κλίμα συνεργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ……..……...σελ.9

ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ.....σελ.13

1. ΑΜΙΝΟΞΕΑ, ΠΡΩΤΕΙΝΕΣ ΚΑΙ ΕΝΖΥΜΑ……..……σελ.15

1.1. Αμινοξέα και Πεπτιδικός δεσμός……..……...σελ.16

1.2. Επίπεδα Δομής των Πρωτεϊνών……..……σελ.23

1.3. Λειτουργίες Πρωτεϊνών……..……σελ.31

1.3.1. Εισαγωγή……..……...σελ.31

1.3.2. Τα Ένζυμα……..……..σελ.31

1.3.3. Εξάρτηση των Ενζύμων από τη Θερμοκρασία……..…σελ.34

2. ΣΤΟΙΧΙΣΗ ΑΛΛΗΛΟΥΧΙΩΝ……..……σελ.37

2.1. Ορισμός και Τρόποι Στοίχισης……..……...σελ.38

2.2. Στοίχιση Κατά Ζεύγη……..……...σελ.41

2.2.1. Πίνακας Σημείων……..……....σελ.41

2.2.2. Αλγόριθμος Δυναμικού Προγραμματισμού……..…...σελ.42

2.2.2.1. Εισαγωγή……..……....σελ.42

2.2.2.2. Αλγόριθμος Smith-Waterman……...…σελ.43

2.2.3. Μέθοδος k-γραμμάτων……..…….σελ.46

2.2.4. Πίνακες Αντικατάστασης……..…….σελ.46

2.2.4.1. Εισαγωγή……..…….σελ.46

2.2.4.2. PAM Πίνακες Αντικατάστασης……..…….σελ.49

2.2.4.3. BLOSUM Πίνακες Αντικατάστασης……..….σελ.51

2.2.4.4. Σχετική Εντροπία……..……σελ.53

2.3. Πολλαπλή Στοίχιση……..…..σελ.56

3. ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ……..……..σελ.61

3.1. Μηχανική Μάθηση……..…σελ.62

3.1.1. Τεχνητή Νοημοσύνη και Μηχανική Μάθηση……..….σελ.62

3.1.2. Ταξινόμηση με Χρήση Μηχανικής Μάθησης……...σελ.64

3.1.3. Στοιχεία Γραμμικής Άλγεβρας……..…….....……...σελ.69

3.1.3.1. Υπερεπίπεδο……..…….....……..……..…….σελ.69

3.1.3.2. Γραφική Απεικόνιση της Πληροφορίας Εκπαίδευσης……σελ.70

3.1.3.3. Επεξεργασία μη Γραμμικώς Διαχωρίσιμης Πληροφορίας Εκπαίδευσης……σελ.71
3.1.4. Γενικός Αλγόριθμος Ταξινόμησης……σελ.74
3.2. Μπεϋζιανός Ταξινομητής……σελ.75
3.2.1. Εισαγωγή……σελ.75
3.2.2. Μάθηση - Ταξινόμηση……σελ.75
3.2.3. Το Ιστόγραμμα και οι Συναρτήσεις Πυρήνα……σελ.78
3.3. Δέντρα Απόφασης……σελ.81
3.3.1. Εισαγωγή……σελ.81
3.3.2. Μάθηση - Ταξινόμηση……σελ.81
3.3.2.1. Δομή Δέντρου - Αλγόριθμου του Hunt……σελ.81
3.3.2.2. Κριτήρια Διαχωρισμού……σελ.83
3.3.2.3. Κλάδεμα Δέντρου……σελ.87
3.4. Μηχανές Διανυσμάτων Υποστήριξης……σελ.88
3.4.1. Εισαγωγή……σελ.88
3.4.2. Μάθηση – Ταξινόμηση……σελ.88
3.4.3. Ελαστικοποίηση του Αλγόριθμου……σελ.93
3.5. Αλγόριθμος Perceptron……σελ.95
3.5.1. Εισαγωγή……σελ.95
3.5.2. Ο Βιολογικός Νευρώνας……σελ.95
3.5.3. Μάθηση – Ταξινόμηση……σελ.97
3.6. κ-Κοντινότερος-Γείτονας Ταξινομητής……σελ.100
3.6.1. Εισαγωγή……σελ.100
3.6.2. Ταξινόμηση……σελ.100
4. ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΛΟΓΙΣΜΙΚΑ……σελ.103
4.1. Βάσεις Δεδομένων……σελ.104
4.1.1. Βιβλιοθήκη UniProt……σελ.104
4.1.2. Βάση Δεδομένων Pfam……σελ.105
4.1.3. Οντολογία Γονιδίων……σελ.106
4.1.4. Ενζυμική Ονοματολογία……σελ.109
4.2. Λογισμικά……σελ.111
4.2.1. Πακέτο Λογισμικών BLAST……σελ.111
4.2.2. Πακέτο Λογισμικών HMMER……σελ.113
4.3.3. Λογισμικό PROFEAT……σελ.114

4.3.4. Λογισμικό RAPIDMINER…….…..….σελ.116
4.3.5. Λογισμικό CD-HIT…….…..……..….σελ.116
ΤΕΧΝΙΚΟ ΜΕΡΟΣ……..…..……σελ.119
1. ΕΙΣΑΓΩΓΗ……..……..σελ.121
2. ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΟΥ ΠΡΟΒΛΕΨΗΣ ΠΡΩΤΕΪΝΙΚΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΕ ΧΡΗΣΗ ΟΝΤΟΛΟΓΙΩΝ.....……σελ.122
2.1. Εισαγωγή……..……σελ.122
2.2. Προετοιμασία – Κατασκευή Κατάλληλης Βάσης Δεδομένων……σελ.123
2.3. Αλγόριθμος Πρόβλεψης Λειτουργίας Άγνωστων Αλληλουχιών……σελ.125
3. ΜΕΛΕΤΗ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ……..σελ.129
3.1. Εισαγωγή……...σελ.129
3.2. Μελέτη Αλγόριθμων Μηχανικής Μάθησης……...σελ.131
3.2.1. Προετοιμασία - Απόκτηση Πρωτεϊνικών Αλληλουχιών…σελ.131
3.2.2. Κατασκευή Πληροφορίας Εκπαίδευσης PFAM Χαρακτηριστικών……..σελ.132
3.2.3. Κατασκευή Πληροφορίας Εκπαίδευσης PROFEAT Χαρακτηριστικών……..…..…σελ.135
3.2.4. Διαδικασία Μηχανικής Μάθησης……...σελ.137
3.3. Αξιοποίηση Αλγόριθμων Μηχανικής Μάθησης……..……σελ.142
3.3.1. Κατασκευή Βάσης Δεδομένων Θερμοανθεκτικών και μη Θερμοανθεκτικών Μικροοργανισμών……...σελ.142
3.2.2. Διαδικασία Μηχανικής Μάθησης……..……..σελ.147
ΑΠΟΤΕΛΕΣΜΑΤΑ - ΣΥΜΠΕΡΑΣΜΑΤΑ.....σελ.149
1. ΠΡΟΒΛΕΨΗ ΠΡΩΤΕΪΝΙΚΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΕ ΧΡΗΣΗ ΟΝΤΟΛΟΓΙΩΝ……...…σελ.151
2. ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ……...σελ.152
2.1. Μελέτη Συμπεριφοράς Αλγόριθμων…….σελ.152
2.2. Αξιοποίηση των Αλγόριθμων για το Διαχωρισμό Ενζύμων με Βάση τη Θερμοκρασία Δράσης τους……....σελ.167
ΠΑΡΑΡΤΗΜΑ 1^{ης} ΕΝΟΤΗΤΑΣ……..……σελ.173
ΠΑΡΑΡΤΗΜΑ (Α) 2^{ης} ΕΝΟΤΗΤΑΣ……..…..σελ.179

ΠΑΡΑΡΤΗΜΑ (Β) 2ης ΕΝΟΤΗΤΑΣ…..…..…….……..σελ.190
ΒΙΒΛΙΟΓΡΑΦΙΑ……..……..……..……..……..σελ.195

ΕΙΣΑΓΩΓΗ

Ο σκοπός της εργασίας είναι διπλός, καθώς ασχολείται με τις γνωστές και ευρέως χρησιμοποιούμενες μεθόδους ανάλυσης βιολογικών αλληλουχιών και τη σύγχρονη τάση χρησιμοποίησης της μηχανικής μάθησης για την εξαγωγή χρήσιμων πληροφοριών από βιολογικά δεδομένα. Αφορμή για αυτή την εργασία ήταν η ανάγκη ανάπτυξης ενός βιοπληροφορικού λογισμικού για το Ευρωπαϊκό πρόγραμμα HotZyme, από το εργαστήριο Μεταβολικής Μηχανικής και Βιοπληροφορικής του Ινστιτούτου Βιολογίας, Φαρμακευτικής Χημείας και Βιοτεχνολογίας του Εθνικού Ιδρύματος Ερευνών σε συνεργασία με το εργαστήριο Βιοτεχνολογίας της Σχολής Χημικών Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου. Το HotZyme είναι ένα ερευνητικό πρόγραμμα που υποστηρίζεται από την Ευρωπαϊκή Ένωση και έχει ως στόχο την εύρεση θερμοανθεκτικών ενζύμων (συγκεκριμένα υδρολασών) σε πληθυσμούς αρχαίων και βακτηρίων, που αναπτύσσονται σε θερμά περιβάλλοντα, με τη χρήση μεταγονιδιοματικής ανάλυσης. Το πρόγραμμα είναι τετραετές και συμμετέχουν συνολικά 13 ερευνητικές ομάδες από 10 Ευρωπαϊκά κράτη και τις ΗΠΑ.

Η δομή της εργασίας αποτελείται από τρία κύρια μέρη, το θεωρητικό, το τεχνικό και τα αποτελέσματα - συμπεράσματα. Το θεωρητικό μέρος αποτελείται από τέσσερα κεφάλαια. Στο πρώτο κεφάλαιο αναφέρονται βασικές γνώσεις για τη δομή και τη λειτουργία των πρωτεϊνών, ενώ με περισσότερη λεπτομέρεια γίνεται αναφορά στα ένζυμα, τον τρόπο λειτουργίας τους και την εξάρτησή τους από τη θερμοκρασία. Στο δεύτερο κεφάλαιο περιγράφονται οι τεχνικές σύγκρισης άγνωστων αλληλουχιών με γνωστές. Οι τεχνικές αυτές χρησιμοποιούνται για την εξαγωγή συμπερασμάτων για τη λειτουργία άγνωστων αλληλουχιών και αποτελούν τη βάση των βιοπληροφορικών εργαλείων που έχουν αναπτυχθεί τα τελευταία χρόνια και ορισμένα από αυτά χρησιμοποιούνται στην πρώτη ενότητα του τεχνικού μέρους. Στο τρίτο κεφάλαιο περιγράφεται η μέθοδος της μηχανικής μάθησης και στη συνέχεια το μαθηματικό υπόβαθρο πέντε αλγόριθμων μηχανικής μάθησης, του Μπεϋζιανού Ταξινομητή Πυρήνα, των Δέντρων Απόφασης, των Μηχανών Διανυσμάτων Υποστήριξης, του αλγόριθμου Perceptron και του κ-

Κοντινότερου-Γείτονα Ταξινομητή. Οι αλγόριθμοι αυτοί χρησιμοποιούνται στη δεύτερη ενότητα του τεχνικού μέρους. Τέλος στο τέταρτο κεφάλαιο αναφέρονται οι βάσεις δεδομένων και τα λογισμικά που χρησιμοποιήθηκαν στο τεχνικό μέρος. Γενικά, σκοπός του θεωρητικού μέρους είναι να παρέχει όλες τις σημαντικές πληροφορίες για την φύση των ενζύμων και τον τρόπο λειτουργίας των βιοπληροφορικών εργαλείων και των αλγόριθμων μηχανικής μάθησης, ώστε να κατανοηθούν καλύτερα οι δυο ενότητες του τεχνικού μέρους.

Το τεχνικό μέρος της εργασίας πραγματοποιήθηκε σε περιβάλλον Linux, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Perl και το εργαλείο MySQL Workbench για την κατασκευή βάσεων δεδομένων. Στην πρώτη ενότητα αναπτύχθηκε ένας αλγόριθμος εύρεσης πρωτεϊνικής λειτουργίας με χρήση Οντολογιών Γονιδίων, ο οποίος χρησιμοποιεί το βιοπληροφορικό λογισμικό BLAST, την βάση δεδομένων UniProt-GOA και την Ενζυμική Ονοματολογία. Στη δεύτερη ενότητα μελετήθηκε η συμπεριφορά των αλγόριθμων μηχανικής μάθησης που αναφέρθηκαν παραπάνω, στην ταξινόμηση ενζύμων με βάση τον κωδικό Ενζυμικής Ονοματολογίας τους. Πιο συγκεκριμένα, αποκτήθηκαν οι αλληλουχίες των ενζυμικών κατηγοριών 3.1.-.-3.2.-.-, 3.4.-.-, 3.5.-.-, 3.6.-.- (υποκατηγορίες των υδρολασών) και οι αλληλουχίες που δεν ήταν υδρολάσες από τη βιβλιοθήκη UniProt. Για κάθε κατηγορία υδρολασών μελετήθηκε η ικανότητα των αλγόριθμων να τη διαχωρίζουν από αλληλουχίες μη υδρολασών με βάση δυο διαφορετικά είδη δεδομένων. Η πρώτη κατηγορία δεδομένων ήταν οι κωδικοί PFAM κάθε αλληλουχίας, οι οποίοι προέκυπταν από το λογισμικό HMMER3 και πληροφορούν για την οικογένεια που ανήκει η κάθε πρωτεΐνη. Η δεύτερη κατηγορία ήταν φυσικοχημικά δεδομένα, τα οποία προέκυπταν από το λογισμικό PROFEAT. Επιπλέον για κάθε αλγόριθμο βρέθηκαν οι τιμές των παραμέτρων που μεγιστοποιούσαν την απόδοσή του. Με βάση τα αποτελέσματα της μελέτης, χρησιμοποιήθηκαν οι πιο αποτελεσματικοί αλγόριθμοι για τον διαχωρισμό ενζύμων κατηγορίας 3.1.-.-, και 3.6.-.- που διαφέρουν μόνο στην συμπεριφορά τους σε υψηλές θερμοκρασίες. Πιο συγκεκριμένα, κατασκευάστηκε μια βάση δεδομένων αλληλουχιών υδρολασών, οι οποίες χαρακτηρίζονταν από την Ενζυμική Ονοματολογία και τη θερμοκρασία βέλτιστης ανάπτυξης του μικροοργανισμού στον οποίο

ανήκουν. Για την κατηγορία 3.1.-.- πραγματοποιήθηκε ο διαχωρισμός μεσόφιλων-υπερθερμόφιλων και μεσόφιλων-θερμόφιλων ενζύμων, ενώ για την κατηγορία 3.6.-.- ο διαχωρισμός μεσόφιλων-υπερθερμόφιλων.

Το συμπέρασμα που προέκυψαν από την πρώτη ενότητα του τεχνικού μέρους είναι η μεγάλη χρησιμότητα βάσεων δεδομένων όπως η UniProt-GOA, η οποία παρέχει άμεσα πληροφορίες σχετικές με τη λειτουργία των άγνωστων εξεταζόμενων αλληλουχιών. Τα αποτελέσματα της δεύτερης ενότητας έδειξαν πως ο Μπεϋζιανός Ταξινομητής Πυρήνα είναι ο καταλληλότερος για το διαχωρισμό ενζύμων με PFAM δεδομένα, ενώ τα Δέντρα Απόφασης προτιμούνται για τα PROFEAT δεδομένα. Επιπλέον τα δυο είδη χαρακτηριστικών διαχωρίζουν τα ένζυμα διαφορετικών θερμοκρασιών με απόδοση περίπου 84%, ενώ σημαντικό ρόλο στον διαχωρισμό με τα PROFEAT χαρακτηριστικά έχουν τα ποσοστά διπεπτιδίων των ενζύμων.

Ι. ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

1

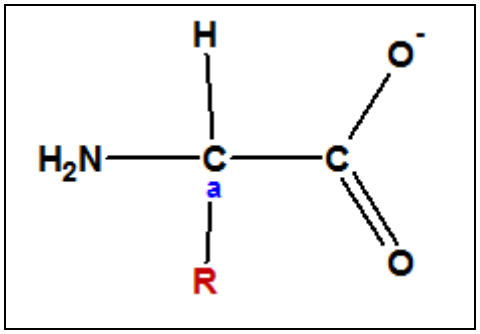
ΑΜΙΝΟΞΕΑ, ΠΡΩΤΕΪΝΕΣ ΚΑΙ ΕΝΖΥΜΑ

«Τα ένζυμα υπερ(θερμόφιλων) μικροοργανισμών διακρίνονται για τις σημαντικές τους δυνατότητες στις βιοτεχνολογικές εφαρμογές της βιομηχανίας. Οι συγκεκριμένοι βιοκαταλύτες χαρακτηρίζονται από υψηλή ακαμψία, η οποία τους επιτρέπει να είναι σταθεροί και να λειτουργούν σε υψηλές θερμοκρασίες και κάτω από αντίξοες συνθήκες χημικών αντιδράσεων.»

HotZyme, "*Collaborative project for systematic screening of novel hydrolases from hot environments*"

1.1. ΑΜΙΝΟΞΕΑ ΚΑΙ ΠΕΠΤΙΔΙΚΟΣ ΔΕΣΜΟΣ

Οι πρωτεΐνες είναι πολυμερή αμινοξέων, τα οποία συνδέονται σε μια μακριά αλυσίδα με ομοιοπολικούς δεσμούς. Υπάρχουν 20 διαφορετικά αμινοξέα που εντοπίζονται στα μόρια των πρωτεϊνών. Τα αμινοξέα υπάρχουν κατ' επανάληψη σε όλες τις πρωτεΐνες, δημιουργώντας μια μακριά αλληλουχία, μοναδική για κάθε πρωτεΐνη. Η βασική δομή τους περιγράφεται στην παρακάτω εικόνα:



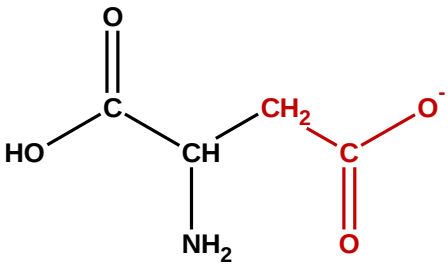
Εικόνα 1.1.1: Συντακτικός τύπος αμινοξέος (L μορφή).[3]

Κάθε αμινοξύ αποτελείται από ένα κεντρικό άτομα άνθρακα (a), το οποίο ενώνεται με μια καρβοξυλική ομάδα (COO^-), μια αμινομάδα (NH_3^+), ένα άτομα υδρογόνου (H) και μια ***πλευρική ομάδα (side chain) (R)***, η οποία διαφοροποιείται για κάθε αμινοξύ. Η πλευρική ομάδα δεν συμμετέχει στον ομοιοπολικό δεσμό μεταξύ των αμινοξέων, αλλά προσδίδει σε κάθε αμινοξύ τις μοναδικές του ιδιότητες.

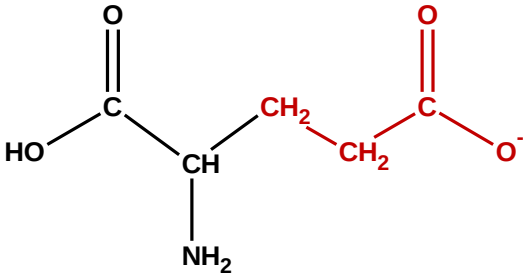
Οι ιδιότητες των αμινοξέων είναι πολύ σημαντικές για την τελική διαμόρφωση και λειτουργία που θα έχει το μόριο της πρωτεΐνης. Πέντε από τα 20 αμινοξέα έχουν πλευρικές ομάδες ικανές να σχηματίσουν ιόντα όταν βρεθούν σε διάλυμα ενώ τα δέκα είναι πολικά και υδρόφιλα και τα άλλα δέκα μη πολικά και υδρόφοβα. Παρακάτω παρουσιάζονται οι συντακτικοί τύποι των 20 αμινοξέων, κατηγοριοποιημένοι με βάση την πολικότητα των πλευρικών ομάδων [1][2]:

- **Πολικά (υδρόφιλα) αμινοξέα:**

Όξινες πλευρικές αλυσίδες:

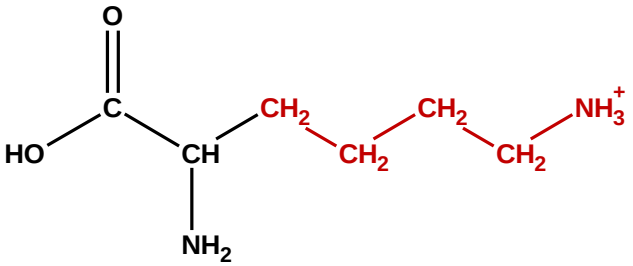


Ασπαρτικό οξύ
(Asp ή D)

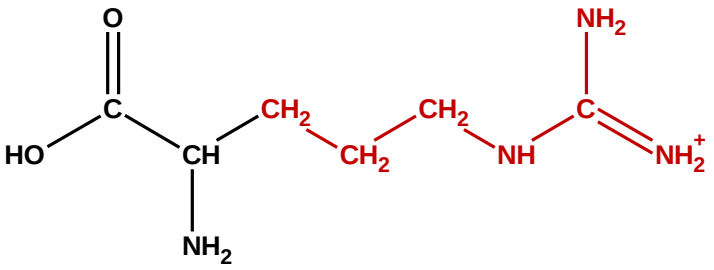


Γλουταμινικό οξύ
(Glu ή E)

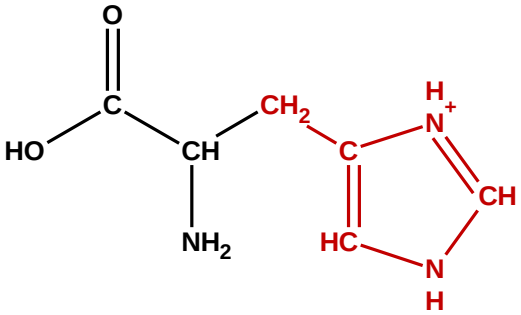
Βασικές πλευρικές αλυσίδες:



Λυσίνη
(Lys ή K)

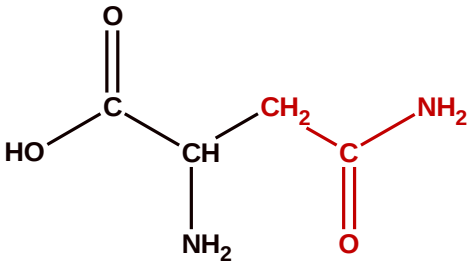


Αργινίνη
(Arg ή R)

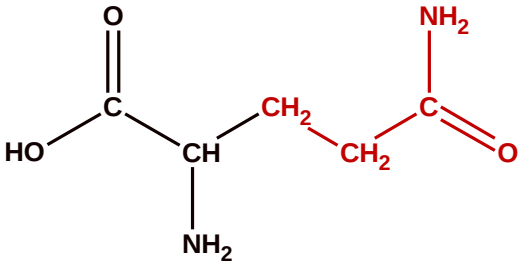


Ιστιδίνη
(His ή H)

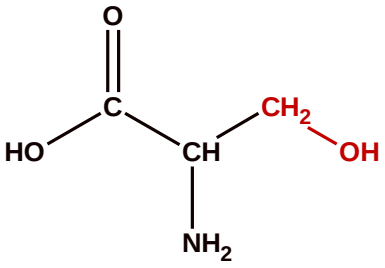
Μη φορτισμένες πλευρικές αλυσίδες:



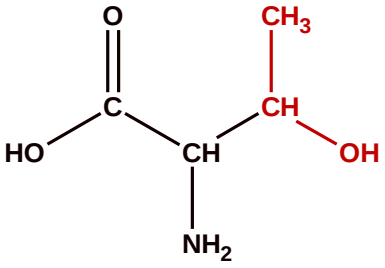
Ασπαραγίνη
(Asn ή N)



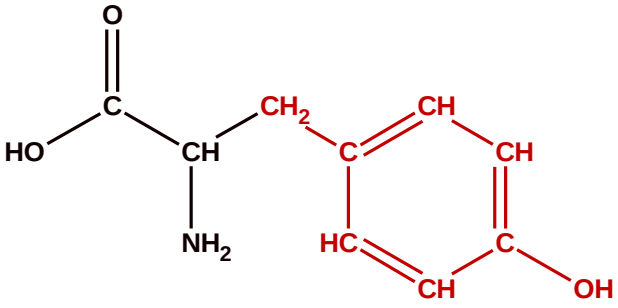
Γλουταμίνη
(Gln ή Q)



Σερίνη
(Ser ή S)

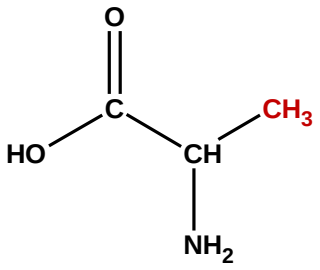


Θρεονίνη
(Thr ή T)

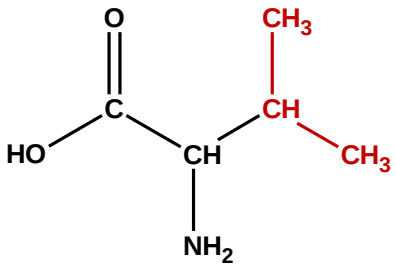


Τυροσίνη
(Tyr ή Y)

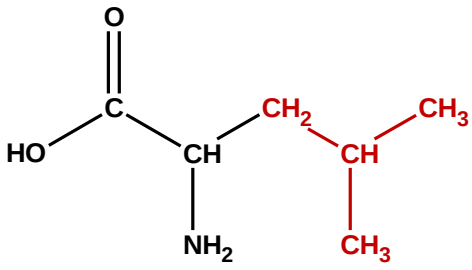
- **Μη πολικά (υδρόφοβα) αμινοξέα:**



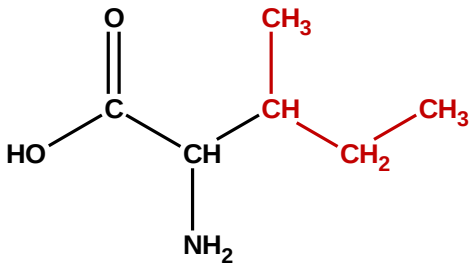
Αλανίνη
(Ala ή A)



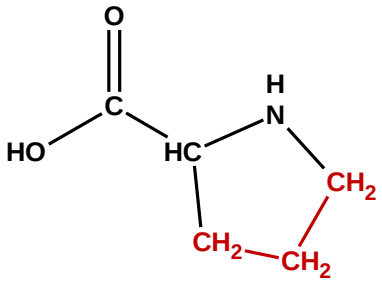
Βαλίνη
(Val ή V)



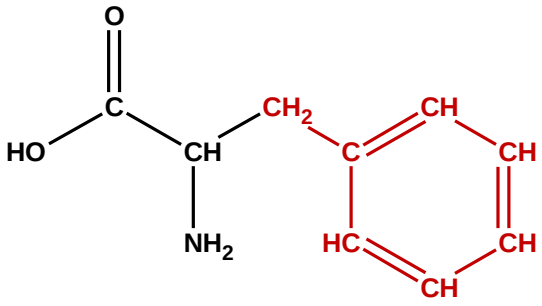
Λευκίνη
(Leu ή L)



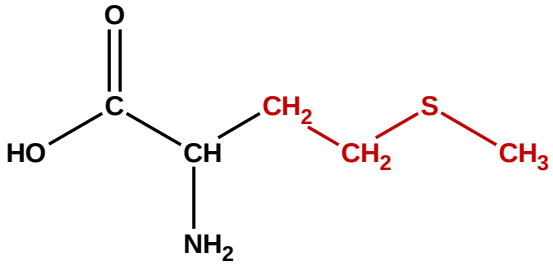
Ισολευκίνη
(Ile ή I)



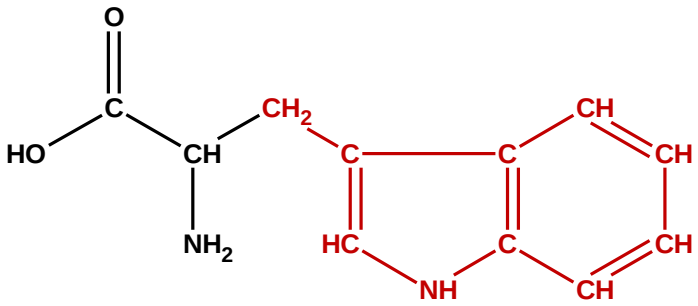
Προλίνη
(Pro ή P)



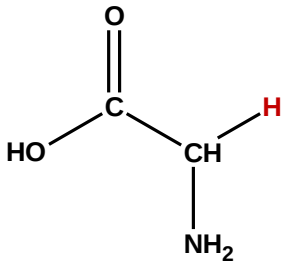
Φαινυλαλανίνη
(Phe ή F)



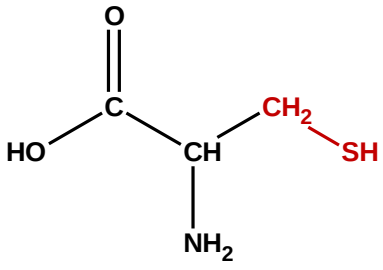
Μεθειονίνη
(Met ή M)



Τρυπτοφάνη
(Trp ή W)



Γλυκίνη
(Gly ή G)

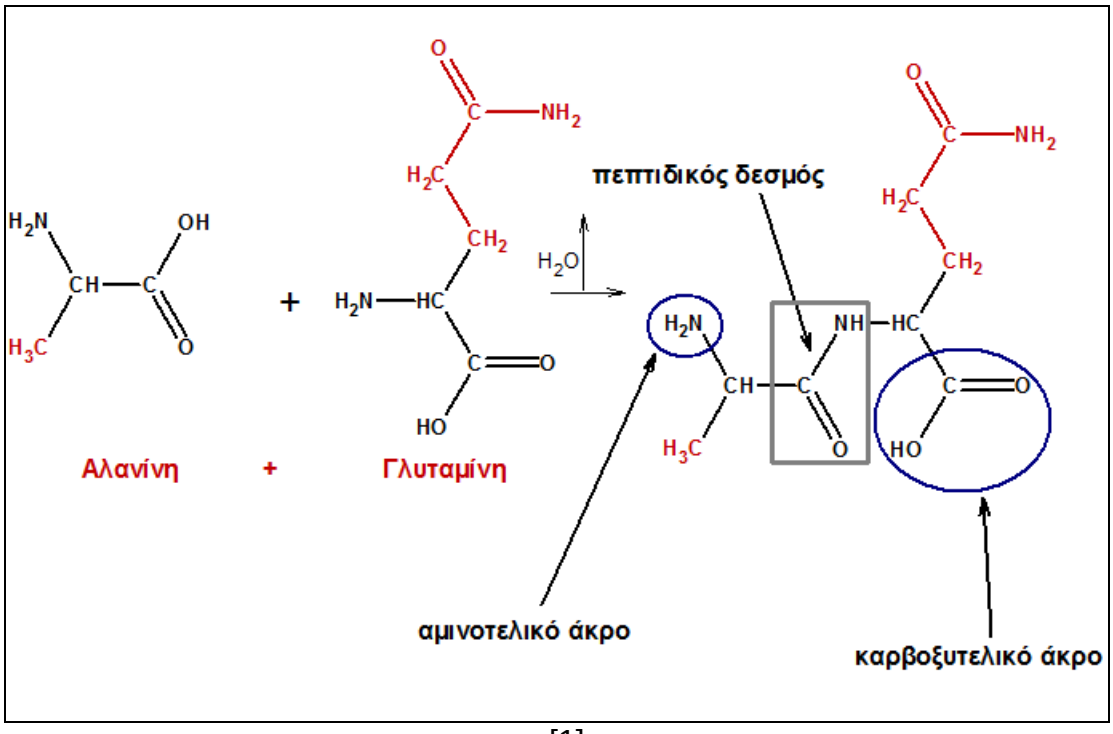


Κυστεΐνη
(Cys ή C)

Τα υδρόφιλα αμινοξέα έχουν την τάση να βρίσκονται κοντά στην επιφάνεια του μορίου της πρωτεΐνης, ενώ τα υδρόφοβα καταλαμβάνουν χώρο στο εσωτερικό (πυρήνα) του μορίου. Εκτός από την συμπεριφορά τους σε σχέση με το νερό, τα αμινοξέα χαρακτηρίζονται και από άλλες κρίσιμες

ιδιότητες. Τα αμινοξέα Tyr, His, Phe, Trp έχουν αρωματικό δακτύλιο στην πλευρική τους ομάδα, η Ala, Val, Leu, Ile, Pro και η Gly έχουν αλειφατική πλευρική ομάδα ενώ δεσμούς υδρογόνου μπορούν να σχηματίσουν η Cys, Trp, Asn, Ser, Thr, Tyr, Lys, Arg, His, Glu και το Asp. Η Cys και η Met περιέχουν θείο αλλά δισουλφιδικό δεσμό, ο οποίος σταθεροποιεί την τρισδιάστατη δομή του μορίου, μπορεί να σχηματίσει μόνο η Cys. Η Gly, λόγω της πλευρικής της ομάδας είναι το αμινοξύ που δημιουργεί κενά και κοιλότητες στην τρισδιάσταση διαμόρφωση της πρωτεΐνης. Η Pro, λόγω της αλεικυκλικής πλευρικής της ομάδας δημιουργεί άκαμπτες διαμορφώσεις ενώ Η Ser και η Thr είναι τα πολικά αμινοξέα που αντιδρούν καλύτερα με το νερό. [2][9]

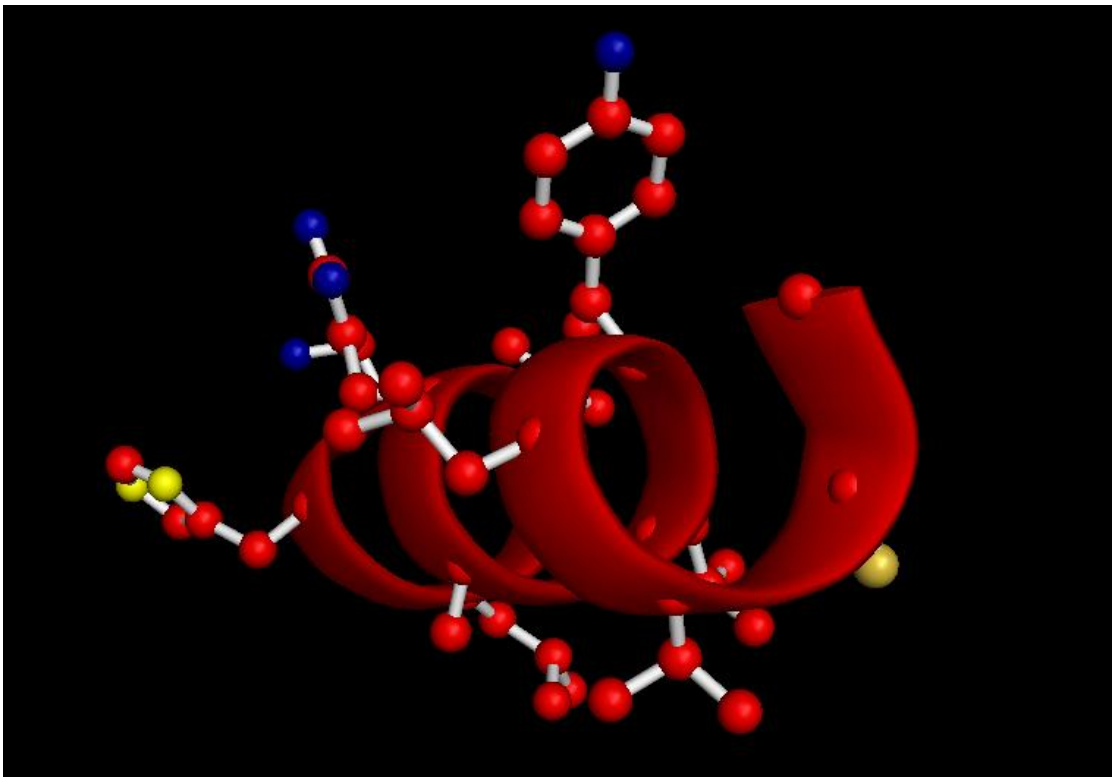
Ο ομοιοπολικός δεσμός, με τον οποίο ενώνονται τα αμινοξέα μεταξύ τους, ονομάζεται ***πεπτιδικός δεσμός (peptide bond)***. Ο σχηματισμός ενός πεπτιδικού δεσμού παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 1.1.2: Σχηματισμός πεπτιδικού δεσμού. [1]

Στο συγκεκριμένο παράδειγμα, ενώνεται μια αλανίνη με μια γλουταμίνη σχηματίζοντας ένα διπεπτίδιο. Το αμινικό άκρο του πρώτου αμινοξέος

(αλανίνη) και το καρβοξυλικό άκρο του τελευταίου (γλουταμίνη) μένουν ελεύθερα. Ο πεπτιδικός δεσμός σχηματίζεται μεταξύ της καρβοξυλικής ομάδας του πρώτου αμινοξέος και της αμινομάδας του επόμενου με ταυτόχρονη απομάκρυνση ενός μορίου H_2O. Τα τέσσερα άτομα στην γκρι περιοχή σχηματίζουν μια άκαμπτη, επίπεδη μονάδα. Η περιστροφή γύρω από τον άξονα C-H είναι αδύνατη. Όταν το επόμενο αμινοξύ προστεθεί στο συγκεκριμένο διπεπτίδιο η καρβοξυλομάδα της γλουταμίνης θα συνδεθεί με την αμινομάδα του και θα δημιουργηθεί ένα τριπεπτίδιο. Η επαναλαμβανόμενη προσθήκη αμινοξέων στην αλυσίδα θα δημιουργήσει τον ***πολυπεπτιδικό σκελετό (polypeptide backbone)***, ο οποίος μαζί με τις πλευρικές ομάδες αποτελεί την ***πολυπεπτιδική αλυσίδα (polypeptide chain)***. Το μήκος των πολυπεπτιδικών αλυσίδων ποικίλει, λιγότερο από 50 μέχρι 5000 αμινοξέα. Στην παρακάτω εικόνα απεικονίζεται ένα τμήμα του μορίου της ινσουλίνης (PDB id: 4INS).



Εικόνα 1.1.3: Πολυπεπτιδικός σκελετός – πλευρικές ομάδες (χρώματα: άνθρακας – κόκκινο, οξυγόνο – μπλε, άζωτο – κίτρινο, θείο - μπεζ).

1.2. ΕΠΙΠΕΔΑ ΔΟΜΗΣ ΤΩΝ ΠΡΩΤΕΪΝΩΝ

Η δομή των πρωτεϊνών διακρίνεται σε τέσσερα διαφορετικά επίπεδα, την πρωτοταγή, δευτεροταγή, τριτοταγή και τεταρτοταγή δομή. Η πρωτοταγής δομή αναφέρεται στην αλληλουχία των αμινοξέων ενώ τα υπόλοιπα τρία στάδια αφορούν τη διαμόρφωση της αλληλουχίας στο χώρο με βάση τους δεσμούς που αναπτύσσονται μεταξύ των αμινοξέων.

Όπως αναφέρθηκε, η αλληλουχία των αμινοξέων αποτελεί την **πρωτοταγή δομή (primary structure)** της πρωτεΐνης. Η αλληλουχία των αμινοξέων δίνει σημαντικές πληροφορίες για το γονίδιο από το οποίο εκφράζεται, καθώς με αντίστροφη μετάφραση και μεταγραφή μπορεί να βρεθεί η αλληλουχία των ριβονουκλεοτιδίων του DNA, η οποία κωδικοποιεί συγκεκριμένη πρωτεΐνη. Συνήθως, πρωτεΐνες με παρόμοιες πρωτοταγείς δομές έχουν και παρόμοια λειτουργία. Όμως έχουν βρεθεί πρωτεΐνες με παρόμοιες πρωτοταγείς δομές και διαφορετικές λειτουργίες αλλά και πρωτεΐνες που εκτελούν παρόμοιες λειτουργίες και έχουν εντελώς διαφορετικές αλληλουχίες αμινοξέων [8]. Παρόλα αυτά, η πρωτοταγής δομή επηρεάζει τα τρία ανώτερα επίπεδα διαμόρφωσης της πρωτεΐνης και για τον λόγο αυτό χρησιμοποιείται για να εξαχθούν χρήσιμες πληροφορίες για την τρισδιάστατη διαμόρφωση της. [2]

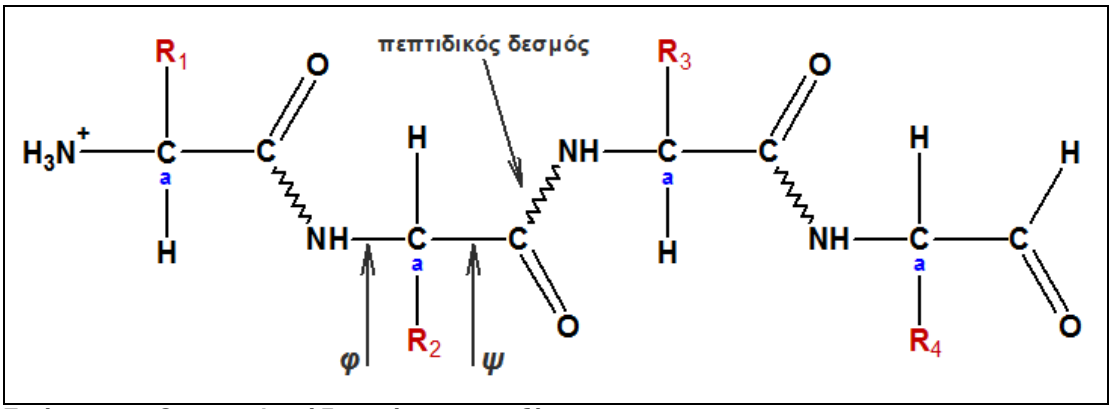
Ένα χαρακτηριστικό παράδειγμα, το οποίο καταδεικνύει τη σημαντικότητα της πρωτοταγούς δομής στην τελική διαμόρφωση του μορίου της πρωτεΐνης, είναι η αιμοσφαιρίνη (hemoglobin) των ερυθρών κυττάρων [2]. Το μόριο της αιμοσφαιρίνης αποτελείται 146 αμινοξέα και το έκτο αμινοξύ από το αμινοτελικό άκρο είναι γλουταμινικό οξύ. Η αιμοσφαιρίνη προσδίδει το σφαιρικό σχήμα στα ερυθρά κύτταρα, το οποίο βοηθά στην πήξη του αίματος στις πληγές. Άτομα που πάσχουν από δρεπανοκυτταρική αναιμία έχουν ερυθρά κύτταρα με δρεπανοειδές σχήμα, το οποίο δεν επιτρέπει στη πήξη του αίματος. Η αλλαγή του σχήματος των ερυθρών κυττάρων, και κατά συνέπεια η ασθένεια αυτή, οφείλεται σε αντικατάσταση του γλουταμινικού οξέος στην έκτη θέση της αλληλουχίας από βαλίνη. Επομένως η αλλαγή ενός

μόνο αμινοξέος είναι ικανή να τροποποιήσει σε μεγάλο βαθμό την τρισδιάστατη διαμόρφωσή.

```
>sp|O00748|EST2_HUMAN Cocaine esterase
MRLHRLRARLSAVACGLLLLLVRGQGQDSASPIRTTHTGQVLGSLVHVKGANAGVQTFLGI
PFAKPPLGPLRFAPPEPPESWSGVRDGTTHPAMCLQDLTAVESEFLSQFNMTFPSDSMSE
DCLYLSIYTPAHSHEGSNLPVMVWIHGGALVFGMASLYDGSMLAALENVVVVIIQYRLGVLG
FFSTGDKHATGNWGYLDQVAALRWVQQNIAHFGGNPDRVTIFGESAGGTSVSSLVVSPIS
QGLFHGAIMESGVALLPGLIASSADVISTVVANLSACDQVDSEALVGCLRGKSKEEILAINKP
FKMIPGVVDGVFLPRHPQELLASADFQPVPSIVGVNNNEFGWLIPKVMRIYDTQKEMDREA
SQAALQKMLTLLMLPPTFGDLLREEYIGDNGDPQTLQAQFQEMMADSMFVIPALQVAHFQ
CSRAPVYFYEFQHQPSWLKNIRPPHMKADHGDELPFVFRSFFGGNYIKFTEEEEQLSRKM
MKYWANFARNGNPNGEGLPHWPLFDQEEQYLQLNLQPAVGRALKAHRLQFWKKALPQKI
QELEEPEERHTEL
```

Εικόνα 1.2.1: Πρωτοταγής δομή εστεράσης – η αλληλουχία των αμινοξέων είναι σε μορφή fasta (fasta format).

Η ***δευτεροταγής δομή (secondary structure)*** της πρωτεΐνης αφορά γνωστά μοτίβα τοπικής τρισδιάστατης διαμόρφωσης. Κάθε αμινοξύ της πολυπεπτιδικής αλυσίδας συμμετέχει σε τρεις ομοιοπολικούς δεσμούς: 1) τον πεπτιδικό δεσμό, 2) το δεσμό μεταξύ του a άνθρακα (aC) και του αζώτου (φ δεσμός) και 3) το δεσμό μεταξύ του a άνθρακα (aC) και του καρβοξυλικού άνθρακα (ψ δεσμός). Αν όλοι οι φ δεσμοί, σε ένα τμήμα της πολυπεπτιδικής αλυσίδας, έχουν την ίδια γωνία περιστροφής και όλοι οι ψ δεσμοί έχουν την ίδια γωνία περιστροφής, το τμήμα αυτό αποκτά μια συνηθισμένη διαμόρφωση στο χώρο. Οι συνήθεις διαμορφώσεις-μοτίβα είναι η α-έλικα και το β-πτυχωτό φύλλο. [6]

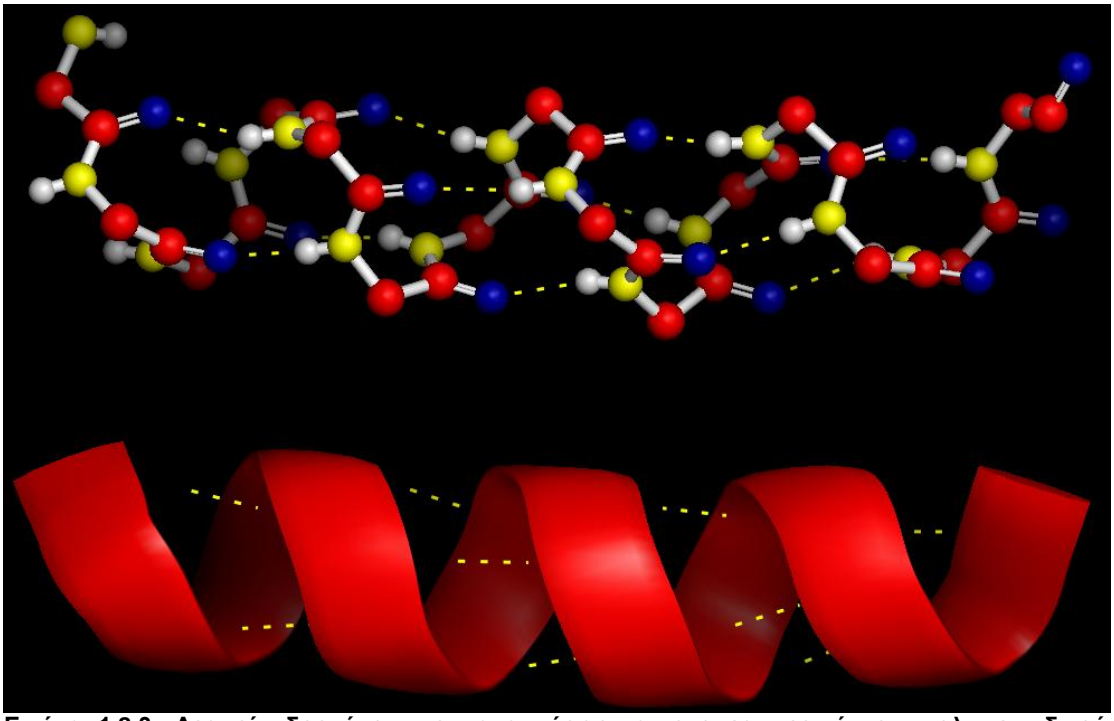


Εικόνα 1.2.2: Ομοιοπολικοί δεσμοί των αμινοξέων.

Τα δυο αυτά πρότυπα είναι κοινά σε όλες τις πρωτεΐνες. Προκύπτουν από την δημιουργία δεσμών υδρογόνου μεταξύ των ομάδων $N-H$ και $C=O$ της πολυπεπτιδικής αλυσίδας. Οι πλευρικές ομάδες δεν συμμετέχουν με κάποιο δεσμό, επομένως οι δομές αυτές μπορούν να σχηματιστούν από πολλές διαφορετικές αλληλουχίες αμινοξέων. Όμως έχει παρατηρηθεί ότι η ύπαρξη ορισμένων αμινοξέων ευνοεί ή διακόπτει την ανάπτυξη μιας δευτεροταγούς διαμόρφωσης. Για παράδειγμα, η λευκίνη, η μεθειονίνη και το γλουταμινικό οξύ ευνοούν την ανάπτυξη α-έλικας ενώ η ισολευκίνη, η βαλίνη και η φαινυλαλανίνη την ανάπτυξη β-πτυχωτού φύλλου. Επιπλέον η γλυκίνη και η προλίνη διακόπτουν τις α-έλικες και τους δημιουργούν καμπές (bends and turns).

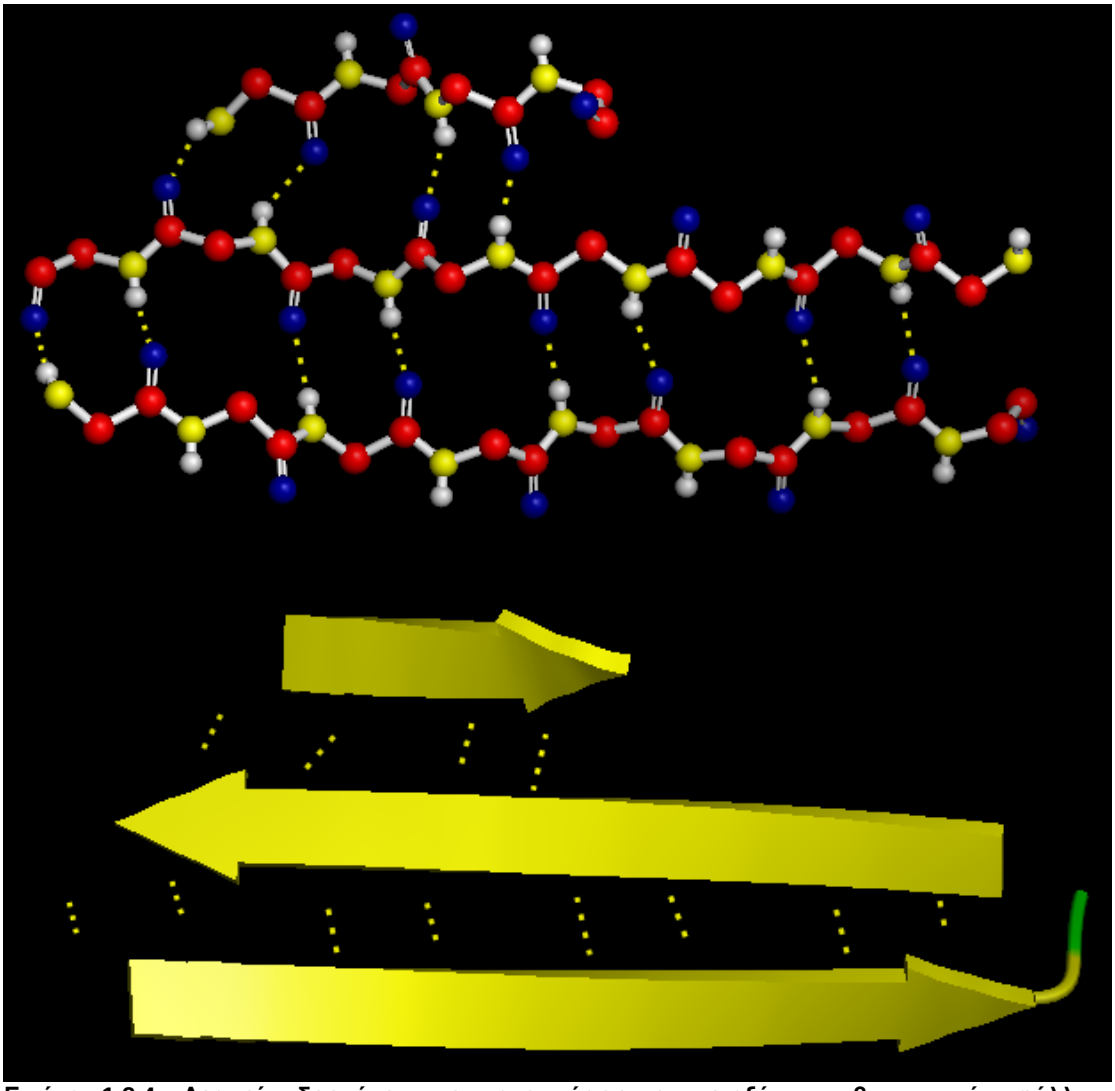
Η ***α-έλικα (alpha-helix)*** είναι μια δομή που μοιάζει με σπειροειδή σκάλα. Σε κάθε πεπτιδικό δεσμό σχηματίζονται δυο δεσμοί υδρογόνου. Ο πρώτος μεταξύ της ομάδας C=O του πεπτιδικού δεσμού και της ομάδας N-H που βρίσκεται τέσσερα αμινοξέα πιο χαμηλά. Ο δεύτερος μεταξύ της ομάδας N-H του πεπτιδικού δεσμού και της ομάδας C=O που βρίσκεται τέσσερα αμινοξέα πιο πάνω. Εκτός από αυτούς τους δεσμούς, οι οποίοι ευθύνονται για τον σχηματισμό της α-έλικας, αναπτύσσονται δεσμοί υδρογόνου και μεταξύ των πλευρικών ομάδων των αμινοξέων, οι οποίοι όσο περισσότεροι είναι τόσο σταθερότερη α-έλικα σχηματίζεται.

Στην πιο συνήθη μορφή της α-έλικας μια πλήρη περιστροφή αποτελείται από 3.6 αμινοξέα (υπάρχουν και α-έλικες με 5.1 αμινοξέα ανά περιστροφή). Για τον λόγο αυτό ορισμένες πλευρικές ομάδες έρχονται πιο κοντά απ’ την συνηθισμένη απόσταση. Αν αυτές οι πλευρικές ομάδες έχουν ίδια πολική συμπεριφορά (πολικές ή μη πολικές) λόγω αποστικών δυνάμεων μπορεί να δημιουργηθεί ένα κενό στην α-έλικα. Αυτά τα κενά είναι πολύ σημαντικά για τις αντιδράσεις της α-έλικας με άλλα τμήματα της πολυπεπτιδικής αλυσίδας, το νερό ή άλλες πολυπεπτιδικές αλυσίδες κατά τον σχηματισμό της τεταρτοταγούς δομής. [6][1]



Εικόνα 1.2.3: Δεσμοί υδρογόνου που αναπτύσσονται στο εσωτερικό του πολυπεπτιδικού σκελετού και σταθεροποιούν την δομή της α-έλικας (χρώματα: άνθρακας – κόκκινο, οξυγόνο – μπλε, άζωτο – κίτρινο, υδρογόνο - άσπρο).

Η διαμόρφωση του ***β-πτυχωτού φύλλου (beta-sheet)*** είναι μια παρατεινόμενη α-έλικα με δυο αμινοξέα ανά περιστροφή. Τα β-πτυχωτά φύλλα σχηματίζονται είτε από αλυσίδες που έχουν τον ίδιο προσανατολισμό, δηλαδή παράλληλες αλυσίδες, είτε από μια αλυσίδα η οποία κάμπτεται και αλλάζει κατεύθυνση, δηλαδή αντιπαράλληλες αλυσίδες. Οι συγκεκριμένες δομές σταθεροποιούνται με τους δεσμούς υδρογόνου που σχηματίζονται μεταξύ διαφορετικών β-πτυχωτών φύλλων της ίδιας πολυπεπτιδικής αλυσίδας ή διαφορετικών.



Εικόνα 1.2.4: Δεσμοί υδρογόνου που αναπτύσσονται μεταξύ των β-πτυχωτών φύλλων (χρώματα: άνθρακας – κόκκινο, οξυγόνο – μπλε, άζωτο – κίτρινο).

Εξετάζοντας την δομή πολλών πρωτεϊνών παρατηρούνται συνεχόμενα τμήματα δευτεροταγούς δομής, τα οποία εμφανίζονται με σημαντική συχνότητα. Τα τμήματα αυτά ονομάζονται ***δομικά μοτίβα (structural motifs)***. Τα μοτίβα μπορεί να είναι απλά όπως α-έλικα – περιστροφή – α-έλικα, β-πτυχωτό φύλλο – περιστροφή – β-πτυχωτό φύλλο, β-πτυχωτό φύλλο – καμπύλη – α-έλικα – καμπύλη - β-πτυχωτό ή πιο σύνθετα. Τα δομικά μοτίβα είναι σημαντικά χαρακτηριστικά των πρωτεϊνών καθώς μπορεί να παρέχουν πληροφορίες για τη λειτουργία τους. Συνήθως, τα δομικά χαρακτηριστικά μιας πρωτεΐνης δίνουν περισσότερες πληροφορίες για τη λειτουργίας της απ' ότι η πρωτοταγής δομή. [7]

Η τρισδιάστατη διαμόρφωση της αλληλουχίας των αμινοξέων περιορίζεται από ασθενείς μη ομοιοπολικούς δεσμούς, οι οποίοι σχηματίζονται είτε μεταξύ των σταθερών ατόμων των αμινοξέων είτε μεταξύ των πλευρικών τους ομάδων. Οι δεσμοί αυτοί είναι δεσμοί υδρογόνου, ιοντικοί δεσμοί και δυνάμεις van der Waals.

Η ***τριτοταγής δομή (tertiary structure)*** μιας πρωτεΐνης συνίσταται από την πλήρη, τρισδιάστατη διαμόρφωση ολόκληρης της πολυπεπτιδικής αλυσίδας και είναι η πιο σημαντική δομή από λειτουργική άποψη, όταν η πρωτεΐνη αποτελείται από μια πολυπεπτιδική αλυσίδα. Η τριτοταγής δομή εξαρτάται αποκλειστικά από τις αλληλεπιδράσεις των πλευρικών ομάδων των αμινοξέων. Οι δεσμοί που αναπτύσσονται είναι μη ομοιπολικοί, όπως δεσμοί υδρογόνου, ιοντικοί και Van der Waals, αλλά και ομοιοπολικοί όπως οι δισουλφιδικοί δεσμοί. Ένας επιπλέον παράγοντας που καθορίζει τη δίπλωση της πρωτεΐνης είναι η κατανομή των πολικών και μη πολικών αμινοξέων στην αλληλουχία της. Οι μη πολικές (υδρόφοβες) πλευρικές αλυσίδες, τείνουν να συναθροίζονται στο εσωτερικό του μορίου ώστε να αποφύγουν την επαφή με το νερό ενώ οι πολικές (υδρόφιλες) πλευρικές ομάδες διατάσσονται κοντά στην εξωτερική επιφάνεια του μορίου, όπου σχηματίζουν δεσμούς υδρογόνου με το νερό και άλλα πολικά μόρια. Η τελική διαμόρφωση της πολυπεπτιδικής αλυσίδας είναι αυτή στην οποία η ελεύθερη ενέργεια του μορίου ελαχιστοποιείται (stable state). [2]

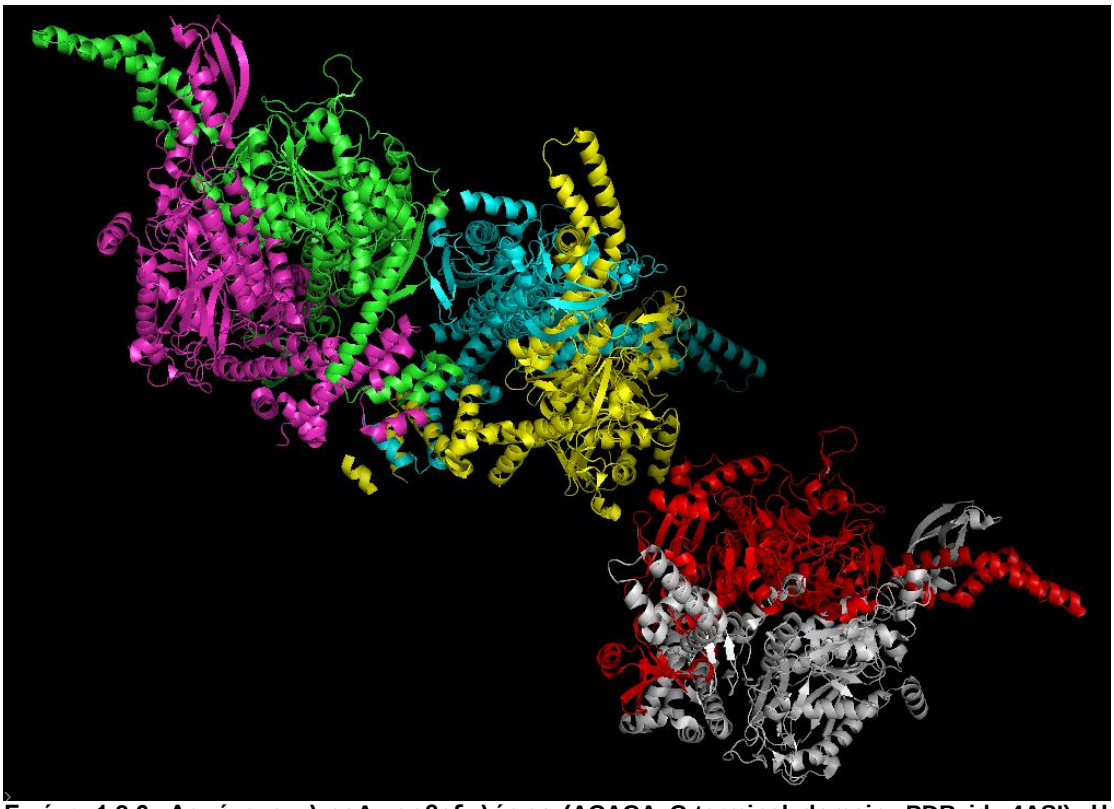
Μέσα στην τρισδιάστατη δομή μιας πρωτεΐνης μπορούν να υπάρχουν ορισμένες ημιαυτόνομες περιοχές, καθοριστικής σημασίας για την λειτουργία του μορίου, οι οποίες ονομάζονται ***πρωτεϊνικές περιοχές (protein domains)***. Μια πρωτεϊνική περιοχή, μήκος 50-350 αμινοξέα, μπορεί να αποτελεί ένα τμήμα της συνολικής τρισδιάστατης δομής του μορίου ή ολόκληρη την πρωτεΐνη. Η πρωτεϊνική περιοχή διαθέτει ένα πυρήνα από υδρόφοβα αμινοξέα και μια εξωτερική επιφάνεια από υδρόφιλα κυρίως αμινοξέα. Η δομή της σταθεροποιείται από τους δεσμούς διπόλου-διπόλου που σχηματίζονται μεταξύ των πολικών αμινοξέων της επιφάνειας και των μορίων νερού του διαλύτη. [6]

Εικόνα 1.2.5: Πρωτεϊνικό σύμπλοκο (TEM-1 BETA-LACTAMASE / BETA-LACTAMASE INHIBITOR). Με πορτοκαλί χρώμα η πρωτεϊνική περιοχή β-λακταμάση (beta-lactamase, UniProt id: P62593), η οποία είναι υδρολάση και σχηματίζεται από τα αμινοξέα των θέσεων 49-259.

Η πρωτεϊνική περιοχή έχει συνήθως σφαιρικό σχήμα και αποτελείται από συναθροιζόμενα τμήματα δευτεροταγούς δομής. Α-έλικες και β-πτυχωτά φύλλα πακετάρονται σχηματίζοντας τις δομές αυτές, ενώ τα αμινοξέα που συμμετέχουν ενώνονται συχνά με τα υπόλοιπα αμινοξέα της πολυπεπτιδικής αλυσίδας. Αυτό έχει σαν αποτέλεσμα τη δημιουργία ελεύθερων περιοχών στο εσωτερικών των πρωτεϊνικών περιοχών. Επιπλέον, παρόλο που η τρισδιάστατη δομή είναι αρκετά πυκνή, υπάρχουν άτομα που ταλαντώνονται και περιστρέφονται δημιουργώντας ελεύθερες περιοχές για την είσοδο μικρών μορίων αλλά και ολόκληρων πρωτεϊνικών περιοχών στο εσωτερικό της πρωτεΐνης. Όπως συμβαίνει και με τα δομικά μοτίβα, η ομοιότητα πρωτεϊνικών περιοχών ίσως δηλώνει και λειτουργική ομοιότητα.

Τεταρτοταγή δομή (quaternary structure) έχουν πρωτεΐνες, οι οποίες αποτελούνται από περισσότερες από μια πολυπεπτιδικές αλυσίδες. Η δομή αυτή αναφέρεται στη διάταξη των πολυπεπτιδικών αλυσίδων, έτσι ώστε να σχηματιστεί το τελικό μόριο της πρωτεΐνης. Σε ορισμένες περιπτώσεις σχηματίζεται και ένα πέμπτο επίπεδο δομής, όπου δυο ή περισσότερες

πρωτεΐνες οργανώνονται σε πολυπρωτεϊνικά σύμπλοκα ***(multiprotein complex)***.



Εικόνα 1.2.6: Δομή ακετυλ-coA-καρβοξυλάσης (ACACA C-terminal domain, PDB id: 4ASI). Η λιγάση αυτή αποτελείται από έξι πολυπεπτιδικές αλυσίδες, οι οποίες είναι ίδιες μεταξύ τους (με πολύ μικρές διαφορές).

1.3. ΛΕΙΤΟΥΡΓΙΕΣ ΠΡΩΤΕΪΝΩΝ

1.3.1 Εισαγωγή

Οι πρωτεΐνες ρυθμίζουν και συμμετέχουν σε όλες τις σημαντικές διεργασίες ενός ζωντανού οργανισμού. Αποτελούν το 50 έως 70% των στερεών του κυττάρου και ανάλογα με την λειτουργία την οποία εκτελούν κατατάσσονται στης εξής κατηγορίες [4]:

- Δομικές: αποτελούν το βασικό υλικό για την κατασκευή του κυτταρικού τοιχώματος και της κυτταρικής μεμβράνης (π.χ. κολλαγόνο, γλυκοπρωτεΐνες).
- Μεταφορικές: μεταφέρουν διάφορες ουσίες όπως το οξυγόνο.
- Προστατευτικές: χρησιμοποιούνται στην άμυνα του οργανισμού (π.χ. αντισώματα).
- Συσταλτικές: αποτελούν μέρος μηχανισμών κίνησης (π.χ. στις βλεφαρίδες, στα μαστίγια).
- Τοξίνες: προκαλούν δηλητηρίαση πάνω από μια ορισμένη συγκέντρωση.
- Ορμόνες: έχουν ρυθμιστικό ρόλο.
- Ένζυμα: είναι οι βιολογικοί καταλύτες καθώς συμμετέχουν στις βιολογικές αντιδράσεις επιταχύνοντας την πραγματοποίησή τους.

1.3.2 Τα Ένζυμα

Τα ένζυμα είναι πολύπλοκα πρωτεΐνικά μόρια ή πρωτεΐνικά μόρια μαζί με μόρια RNA, τα οποία καταλύουν βιοχημικές αντιδράσεις. Ανάλογα με την αντίδραση που καταλύουν κατατάσσονται σε έξι μεγάλες κατηγορίες [3]:

ΠΙΝΑΚΑΣ 1.3.1: Βασικές κατηγορίες ενζύμων

Κατηγορία ενζύμου	Καταλυόμενη αντίδραση
1. Οξειδοαναγωγάσες (Oxidoreductases)	Οξειδοαναγωγή
2. Τρανσφεράσες (Transferases)	Μεταφορά ομάδας
3. Υδρολάσες (Hydrolases)	Διάσπαση δεσμών με την προσθήκη νερού
4. Λυάσες (Lyases)	Διάσπαση δεσμών χωρίς την προσθήκη νερού
5. Ισομεράσες (Isomerases)	Αντίδραση ισομερίωσης
6. Λιγάσες (Ligases)	Σχηματισμός δεσμών με ταυτόχρονη απομάκρυνση νερού

Τα σημαντικότερο χαρακτηριστικό των ενζύμων είναι η μεγάλη τους εξειδίκευση, η οποία οφείλεται στη μοναδικότητα της τρισδιάστατης διαμόρφωσής τους. Τα ένζυμα έχουν την ικανότητα να επιταχύνουν μια βιοχημική αντίδραση από 10^7 έως και 10^{14} φορές, μειώνοντας την ενέργεια ενεργοποίησης της αντίδρασης. Όμως δεν μπορούν να πραγματοποιήσουν μια αντίδραση, η οποία θερμοδυναμικά είναι αδύνατο να γίνει. Για να πραγματοποιηθεί μια αντίδραση θα πρέπει η ελεύθερη ενέργεια κατά Gibbs της αντίδρασης να είναι αρνητική:

$$\boxed{\Delta G_r < 0} \qquad (1.3.1)$$

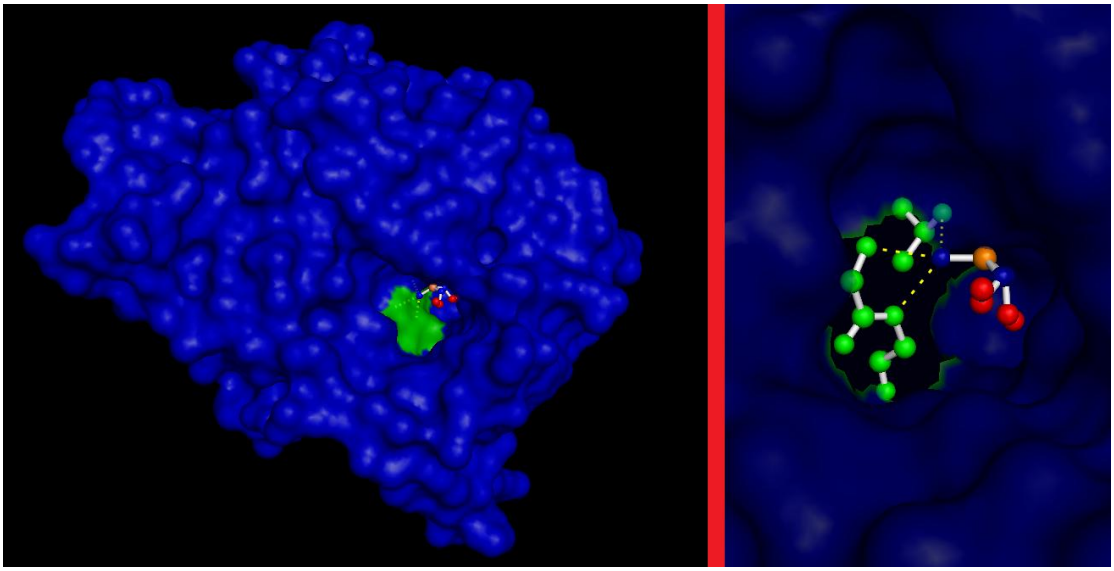
Θα πρέπει δηλαδή, η ελεύθερη ενέργεια κατά Gibbs των προϊόντων να είναι μικρότερη απ' αυτή των αντιδρώντων. [5]

Πριν πραγματοποιηθεί η κατάλυση, το πρωτεϊνικό μόριο ίσως χρειάζεται να συνδεθεί με ένα επιπλέον μικρό μόριο ώστε να γίνει λειτουργικό. Το μόριο αυτό μπορεί να είναι οργανική ένωση, όπως κάποιο συνένζυμο (co-enzyme), μέταλλο, αμμέταλο, μεταλλοειδές ή ιόν και ονομάζεται ***συνδέτης (ligand)*** ή ***προσθετική ομάδα (prosthetic group)***.

Το ένζυμο εμφανίζει συνήθως διάφορες σχισμές στην τρισδιάστατη δομή του, ώστε να επιτρέπει την είσοδο και έξοδο μορίων μέσα σ' αυτό. Το τμήμα του ενζύμου, στο οποίο προσαρμόζεται το ***υπόστρωμα (substrate)*** και πραγματοποιείται η κατάλυση, βρίσκεται εκτεθειμένο μέσα σε σχισμές και ονομάζεται ***ενεργό κέντρο (active site)***. Η επιφάνεια του ενεργού κέντρου αποτελεί περίπου το 5% της συνολικής επιφάνειας του ενζύμου. Σχηματίζεται από αμινοξέα, τα οποία βρίσκονται κοντά λόγω της τριτοταγούς διαμόρφωσης του μορίου. Τα αμινοξέα του ενεργού κέντρου μπορεί να μην εκτελούν την ίδια λειτουργία, για τον λόγο αυτό διακρίνονται στις εξής κατηγορίες:

- αμινοξύ «επαφής»: σχηματίζει δεσμούς με το υπόστρωμα.
- «βοηθητικό» αμινοξύ: δεν συμμετέχει στην αντίδραση, αλλά συμβάλλει στην σταθεροποίηση και τις ιδιότητες του ενεργού κέντρου.
- αμινοξύ «συνεισφοράς»: συμμετέχει έμμεσα στην αντίδραση, καθώς αποτελεί το δομικό υπόβαθρο πάνω στο οποίο στηρίζεται η δραστική διαμόρφωση του ενεργού κέντρου.
- αμινοξύ που δεν έχει καμία συνεισφορά στο ενεργό κέντρο και μπορεί να απομακρυνθεί χωρίς να επηρεαστεί η λειτουργία του ενζύμου.

Στην παρακάτω εικόνα φαίνεται με μπλε χρώμα η τριτοταγής δομή μιας καρβοξυλεστεράσης (PDB id: 2O7V) και το ενεργό της κέντρο με πράσινο. Τα αμινοξέα τα οποία αποτελούν το ενεργό της κέντρο είναι η Gly-91, Gly-92 και Ala-170. Ο συνδέτης είναι το διαιθύλ-όξυ φωσφορώδες (DEP, $C_4H_{11}O_3P$).



Εικόνα 1.3.1: Η τρισδιάστατη δομή καρβοξυλεστεράσης και το ενεργό της κέντρο.

Πριν την πραγματοποίηση της κατάλυσης, το ενεργό κέντρο αναπτύσσει συνήθως ασθενείς δεσμούς (υδρογόνου και ιονικούς) με το υπόστρωμα. Για πολλά έτη υποστηριζόταν η άποψη ότι κατά την κατάλυση, το ενεργό κέντρο και το υπόστρωμα ενώνονται σύμφωνα με το μοντέλο ***κλειδαριάς-κλειδιού (lock-and-key model)***, άποψη η οποία αντικατόπτριζε τη μεγάλη εξειδίκευση των ενζύμων. Όμως, το μοντέλο αυτό δεν έδινε καμία πληροφορία για τον μηχανισμό της κατάλυσης. Για τον λόγο αυτό δαιτυπώθηκε το μοντέλο του ***τεχνητού ταιριάσματος (induced-fit model)***, το οποίο δέχεται πως οι ασθενείς δεσμοί που αναπτύσσονται μεταξύ ενεργού κέντρου και υποστρώματος παραμορφώνουν τόσο το ενεργό κέντρο όσο και το υπόστρωμα. Η παραμόρφωση αυτή καθιστά τους ομοιοπολικούς δεσμούς του υποστρώματος πιο ασθενείς ευνοώντας την πραγματοποίηση της κατάλυσης. Εκτός από τους ασθενείς δεσμούς, το ενεργό κέντρο σχηματίζει και ομοιοπολικούς δεσμούς με το υπόστρωμα, με στόχο να δημιουργήσει το κατάλληλο χημικό περιβάλλον για την πραγματοποίηση της κατάλυσης. [2]

1.3.3 Εξάρτηση των Ενζύμων από τη Θερμοκρασία

Η εξάρτηση των ενζύμων από την θερμοκρασία δεν έχει ιδιαίτερη σημασία για ομοιόθερμους οργανισμούς. Τα ένζυμα των οργανισμών αυτών δρουν σε συγκεκριμένη θερμοκρασία, η οποία δεν επηρεάζεται από το περιβάλλον. Αντίθετα, υπάρχουν μικροοργανισμοί των οποίων οι βιοχημικές διεργασίες επηρεάζονται σημαντικά από την θερμοκρασία του περιβάλλοντος. Ανάλογα με την βέλτιστη θερμοκρασία ανάπτυξης και διαβίωσης οι μικροοργανισμοί κατατάσσονται ως ψυχρόφιλοι ($-5^{o}C<T_{opt}<20^{o}C$), μεσόφιλοι ($20^{o}C<T_{opt}<45^{o}C$), θερμόφιλοι ($45^{o}C<T_{opt}<80^{o}C$) και υπερθερμόφιλοι ($80^{o}C< T_{opt}$).

Η αύξηση της θερμοκρασίας στο περιβάλλον ενός μικροοργανισμού οδηγεί σε αύξηση του ποσοστού απόδοσης ενός ενζύμου, καθώς αυξάνεται η κινητική ενέργεια ενζύμου και υποστρώματος και κατά συνέπεια ο αριθμός των μεταξύ τους συγκρούσεων. Όμως πάνω από μια ορισμένη θερμοκρασία, τα ένζυμα μετουσιώνονται, δηλαδή σπάνε οι δεσμοί υδρογόνου και

καταστρέφεται η τριτοταγής δομή τους. Τα ένζυμα καταστρέφονται και οι βιοχημικές διεργασίες σταματούν.

Η ανακάλυψη των διαφορών μεταξύ θερμοανθεκτικών και μη θερμοανθεκτικών ενζύμων, που καταλύουν ίδιες αντιδράσεις, έχει αρκετά μεγάλο βιοτεχνολογικό ενδιαφέρον. Ένζυμα που δρουν σε υψηλές θερμοκρασίες μπορούν να χρησιμοποιηθούν στην βιομηχανία χωρίς τον περιορισμό της μετουσίωσης σε μέτριες θερμοκρασίες. Για τον λόγο αυτό έχουν ερευνηθεί αλληλουχίες όμοιων ενζύμων που διαφέρουν μόνο ως προς την σταθερότητά τους σε σχέση με τη θερμοκρασία. Έχει βρεθεί πως η σύσταση των πολικών φορτισμένων αμινοξέων (κυρίως Arg, Glu, Lys) αυξάνεται και η σύσταση των μη πολικών (κυρίως Ser, Thr, Asn, Gln) μειώνεται με την αύξηση της θερμοκρασίας. Οι διαφορές αυτές οφείλονται στο γεγονός ότι τα φορτισμένα αμινοξέα αυξάνουν την σταθερότητα του πρωτεϊνικού μορίου, καθώς δημιουργούν περισσότερους ιοντικούς δεσμούς και γέφυρες άλατος. Τα θερμοανθεκτικά ένζυμα έχουν υψηλότερες συγκεντρώσεις υδροφοβικότητας και με τον τρόπο αυτό αποκτούν πιο συμπαγές εσωτερικό και είναι άκαμπτα. Επιπλέον σε μεγάλες θερμοκρασίες η Asn και η Gln υφίστανται απαμίνωση, ενώ η Cys και η Met οξειδώνονται. Για τους λόγους αυτούς οι συγκεντρώσεις τους είναι υψηλότερες στα μη θερμοανθεκτικά ένζυμα. Οι α-έλικες προσδίδουν σταθερότητα στο πρωτεϊνικό μόριο, επομένως συναντώνται σε μεγαλύτερες συγκεντρώσεις στα θερμοανθεκτικά ένζυμα, πιο συγκεκριμένα, μοτίβα αμινοξέων που υπάρχουν συνήθως σε α-έλικες εμφανίζονται περισσότερες φορές στα θερμοανθεκτικά ένζυμα ενώ μοτίβα αμινοξέων συνήθη σε β-πτυχωτά φύλλα έχουν μεγαλύτερες συγκεντρώσεις σε μεσόφιλα ένζυμα. [9][10][11]

2

ΣΤΟΙΧΙΣΗ ΑΛΛΗΛΟΥΧΙΩΝ

«Βιοπληροφορική (Bioinformatics): Ο διεπιστημονικός τομέας που εμπεριέχει την βιολογία, την επιστήμη των υπολογιστών, τα μαθηματικά και την στατιστική και στοχεύει στην ανάλυση των βιολογικών αλληλουχιών και του γονιδιακού περιεχομένου και στην πρόβλεψη της δομής και της λειτουργίας των βιολογικών μακρομορίων.»

David W. Mount "*Bioinformatics, Sequence and Genome Analysis*"

2.1. ΟΡΙΣΜΟΣ ΚΑΙ ΤΡΟΠΟΙ ΣΤΟΙΧΙΣΗΣ

Η ***στοίχιση αλληλουχιών (sequence alignment)*** είναι μια διαδικασία σύγκρισης δυο ή περισσότερων αλληλουχιών, όπως πρωτεΐνες, DNA και RNA, με σκοπό την εύρεση ταυτόσημων ή παρόμοιων περιοχών ή τον εντοπισμό όμοιων μοτίβων. Οι εξεταζόμενες αλληλουχίες κατατάσσονται η μια κάτω από την άλλη με τέτοιο τρόπο, ώστε τα μονομερή ή οι περιοχές, που πρόκειται να εξεταστούν για να διαπιστωθούν ομοιότητες, να βρίσκονται στην ίδια στήλη. Η μια αλληλουχία είναι συνήθως άγνωστη, δηλαδή δεν υπάρχουν πληροφορίες για τη λειτουργίας της ή τις φυλογενετικές της σχέσεις, και σκοπός της στοίχισης είναι να βρεθούν αυτές οι πληροφορίες. Οι υπόλοιπες αλληλουχίες είναι γνωστές και προέρχονται συνήθως από βάσεις δεδομένων, όπου υπάρχουν καταγεγραμμένα πολλά σημαντικά χαρακτηριστικά τους. Οι ομοιότητες στην αλληλουχία, ίσως να συνεπάγονται ομοιότητες στην λειτουργία και την δομή ή φυλογενετικές σχέσεις μεταξύ των βιολογικών μακρομορίων, όπως αναφέρθηκε και στο Κεφάλαιο 1.

Οι δυο κατηγορίες της στοίχισης αλληλουχιών είναι η ***στοίχιση κατά ζεύγη (pair-wise alignment)*** και η ***πολλαπλή στοίχιση (multiple alignment)***. Η στοίχιση κατά ζεύγη πραγματοποιείται μόνο με δυο αλληλουχίες, όπως φαίνεται στην εικόνα 2.1.1.

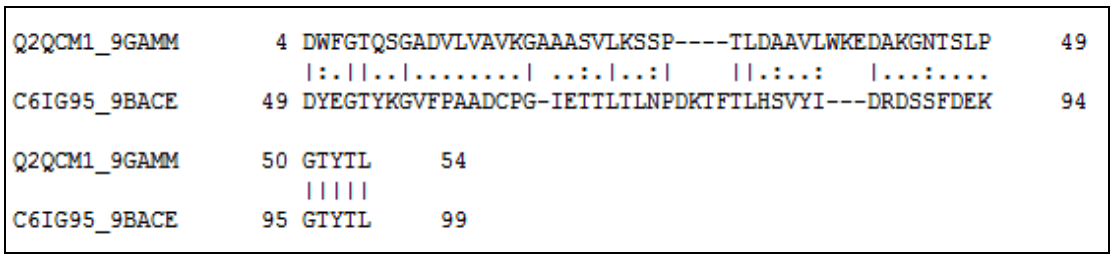
```
>TR:A4YGL6_METS5 A4YGL6 Transcriptional regulator-like protein OS=Metallosphaera
           sedula (strain ATCC 51363 / DSM 5348) GN=Msed_1410 PE=4
           SV=1
          Length = 116

 Score =  114 bits (285), Expect = 4e-31
 Identities = 55/116 (47%), Positives = 84/116 (72%), Gaps = 5/116 (4%)

Query: 10  SKKIQAPCETSIKHVIPAIKSILAQELIKRGYSQKDIGELLGISVAEVNYLIKGKRGDLE 69
           S+K+  PCET+++ VIPAIK++L +EL+K+G SQ     LLG++ AEV+Y +KGKR + E
Sbjct: 5   SRKLMIPCETAMREVIPAIKALLVKELVKQGESQSHTASLLGLTPAEVSYYLKGKRAEGE 64

Query: 70  IKKELENDEDFVDLINSFVNKLVNEKDSEDIVISLCPLCSYARRKVLKQDSACPYD 125
            K  LENDE+F+++I  + ++L +E D     +++CPLCS AR+K+   D +CPYD
Sbjct: 65  YKTILENDEEFMEMIRHYTSRL-HEADR----VNICPLCSLARKKLGIMDYSCPYD 115
```

Εικόνα 2.1.1: Στοίχιση κατά ζεύγη. Η άγνωστη αλληλουχία (πάνω) και η γνωστή A4YGL_METS5 (κάτω).

Αντίθετα η πολλαπλή στοίχιση πραγματοποιείται με τρεις ή περισσότερες αλληλουχίες και χρησιμοποιείται για την εύρεση μοτίβων ή πρωτεϊνικών περιοχών σε μια οικογένεια αλληλουχιών, αλλά και για έλεγχο ομοιότητας μιας άγνωστης αλληλουχίας με μια οικογένεια (εικόνα 2.1.2).

Εικόνα 2.1.2: Πολλαπλή στοίχιση πρωτεϊνικών αλληλουχιών σε ένα συγκεκριμένο τμήμα της κάθε μιας.

Υπάρχουν δυο διαφορετικοί τρόποι στοίχισης αλληλουχιών, η ***τοπική (local alignment)*** και η ***συνολική στοίχιση (global alignment)***. Κατά την τοπική στοίχιση το αποτέλεσμα είναι η εύρεση μίας ή περισσοτέρων περιοχών υψηλής ομοιότητας μεταξύ των αλληλουχιών. Για τον λόγο αυτό, η τοπική στοίχιση είναι χρήσιμη για την εξέταση αλληλουχιών με μεγάλο μήκος, οι οποίες έχουν μικρά ποσοστά ομοιότητας ή για την εξέταση μεγάλου αριθμού αλληλουχιών, όπως γίνεται όταν χρησιμοποιούνται βάσεις δεδομένων. [12][13]



Εικόνα 2.1.3: Τοπική στοίχιση. Η στοίχιση πραγματοποιείται μεταξύ των 4-54 αμινοξέων της πρωτεΐνης Q2QCM1_9GAMM και των 49-99 αμινοξέων της C6IG95_9BACE. Τα υπόλοιπα τμήματα αμινοξέων δεν συμπεριλαμβάνονται στην στοίχιση καθώς δεν εμφανίζουν σημαντικές ομοιότητες μεταξύ τους.

Αντίθετα, κατά την συνολική στοίχιση, το αποτέλεσμα είναι η στοίχιση των αλληλουχιών σε όλο τους το μήκος, δηλαδή εκτός από περιοχές ομοιότητας υπάρχουν και περιοχές όπου οι αλληλουχίες δεν έχουν καμία ομοιότητα. Η μέθοδος της συνολικής στοίχισης είναι πιο αργή και για τον λόγο αυτό δεν χρησιμοποιείται όταν αναζητούνται ομοιότητες άγνωστων αλληλουχιών σε μια βάση δεδομένων. Προτιμάται για την εξέταση αλληλουχιών, οι οποίες είναι γνωστό πως έχουν αυξημένα ποσοστά ομοιότητας ή για την σύγκριση μιας άγνωστης αλληλουχίας με ένα μικρό αριθμό γνωστών.

```
Q2QCM1_9GAMM       1 IFSDWFGTQSGADVLVAVKGAAASVLKSSPTLDAAVLWKEDAKGNTSLPG     50
                     :..                 ....:|..|..|.||.........:||...
C6IG95_9BACE       1 MIM-----------------RKIIILVCSCFLLAACGNSAKTNNSTSADS     33

Q2QCM1_9GAMM      51 TYTLGVASHDSFLTKLEAEQFLPQEQKDALAAAFGTGTPLNGLAQITQVY    100
                     |.|..|..|::. |.|:.|.    ..|....||...|..........:.:
C6IG95_9BACE      34 TATEVVDIHNAE-TSLDYEG----TYKGVFPAADCPGIETTLTLNPDKTF     78

Q2QCM1_9GAMM     101 TGAVKLPYFLSSPATAGSWDKAKTQSWHG--    129
                     |  :...|....    .|:|:..|.:..|
C6IG95_9BACE      79 T--LHSVYIDRD----SSFDEKGTYTLEGNI    103
```

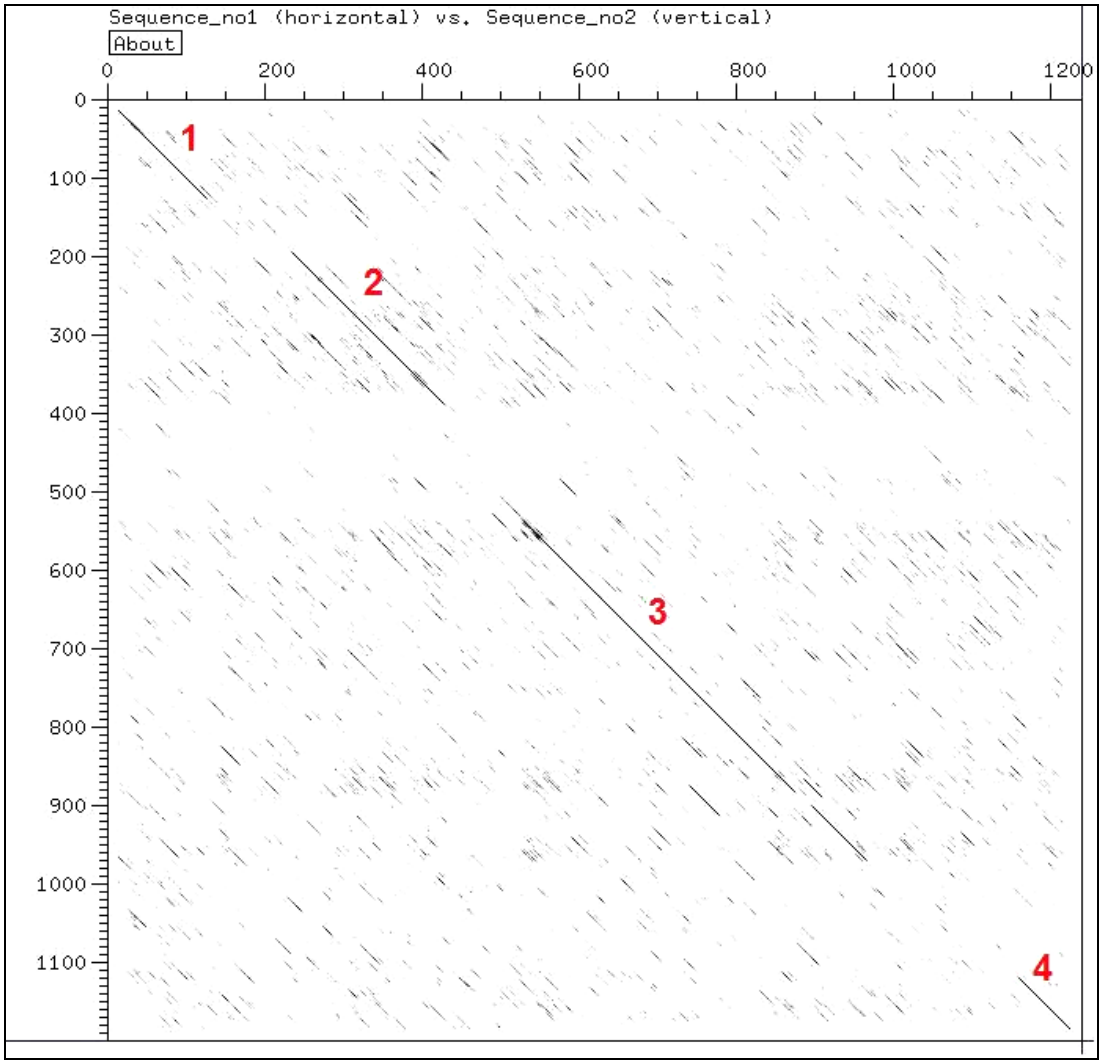
Εικόνα 2.1.4: Συνολική στοίχιση. Οι δυο αλληλουχίες, Q2QCM1_9GAMM και C6IG95_9BACE, στοιχίζονται από την αρχή μέχρι το τέλος τους.

2.2. ΣΤΟΙΧΙΣΗ ΚΑΤΑ ΖΕΥΓΗ

2.2.1. Πίνακας Σημείων

Όταν είναι άγνωστο το ποσοστό ομοιότητας μεταξύ δυο αλληλουχιών ο σχηματισμός του πίνακα σημείων μπορεί να αποτελέσει μια πρώτη εξέταση για τυχόν ομοιότητες. Η δομή του πίνακα σημείων είναι ένα απλό διάγραμμα δυο διαστάσεων (x,y). Στην οριζόντια και άνω πλευρά του διαγράμματος τοποθετείται η αλληλουχία Α και στον αριστερό και κάθετο άξονα η αλληλουχία Β. Η εξέταση ξεκινά με τον πρώτο χαρακτήρα της αλληλουχίας Β, (Β1). Εξετάζοντας έναν-έναν τους χαρακτήρες της αλληλουχίας Α, όπου ο Β1 είναι όμοιος με τον Α τοποθετείται μια τελεία στον πίνακα. Έτσι στο τέλος του ελέγχου του χαρακτήρα Β1, η πρώτη γραμμή του πίνακα έχει ορισμένες τελείες, όπου ο Β1 είναι όμοιος με τους Α. Η διαδικασία αυτή πραγματοποιείται για όλους του χαρακτήρες της Β αλληλουχίας. Στο τέλος ο πίνακας σημείων απεικονίζει με την μορφή τελείων όλα τα πιθανά ταιριάσματα μεταξύ των χαρακτήρων των δυο αλληλουχιών. Οι ***περιοχές υψηλής ομοιότητας (maximal segment pairs, MSP)*** παριστάνονται ως διαγώνιες γραμμές (εικόνα 2.2.1). Απομονωμένες τελείες, οι οποίες δεν βρίσκονται στις διαγώνιους, απεικονίζουν τυχαία ταιριάσματα χαρακτήρων Α και Β. [12][13]

Το πλεονέκτημα αυτής της μεθόδου είναι η αποκάλυψη όλων των περιοχών ομοιότητας είτε αυτές είναι μεγάλες είτε μικρές. Όμως τα μειονεκτήματα είναι πολλά. Ο πίνακας σημείων δεν έχει σαν αποτέλεσμα την στοίχιση των δυο αλληλουχιών αλλά ένα διάγραμμα σημείων, όπου οι περιοχές ομοιότητες παριστάνονται με διαγώνιες ευθείες χωρίς περισσότερες λεπτομέρειες. Επιπλέον, οι τυχόν ομοιότητες δεν ποσοτικοποιούνται έτσι ώστε να διαπιστωθεί αν είναι σημαντικές ή όχι. Τέλος η μέθοδος αυτή δεν μπορεί να χρησιμοποιηθεί σε βάσεις δεδομένων ή όταν ο όγκος των άγνωστων αλληλουχιών που πρόκειται να εξεταστούν είναι μεγάλος. [13]



Εικόνα 2.2.1: Πίνακας σημείων. Οι αλληλουχίες 1 (οριζόντια) και 2 (κάθετη) έχουν τέσσερις περιοχές υψηλής ομοιότητας (MSP).

2.2.2. Αλγόριθμος Δυναμικού Προγραμματισμού

2.2.2.1 Εισαγωγή

Ο δυναμικός προγραμματισμός είναι μια υπολογιστική μέθοδος, η οποία δημιουργεί την βέλτιστη στοίχιση μεταξύ δυο αλληλουχιών και την ποσοτικοποιεί υπολογίζοντας ένα ***σκορ στοίχισης (alignment score)***. Η στοίχιση περιλαμβάνει ομοιότητες και ανομοιότητες μεταξύ των δυο αλληλουχιών και διαστήματα με κενά. Το σκορ που υπολογίζεται εξαρτάται από τις παραμέτρους που χρησιμοποιούνται. Η τεχνική του δυναμικού προγραμματισμού μπορεί να εφαρμοστεί και στους δυο τρόπους στοίχισης

(τοπική και συνολική). Κατά την τοπική στοίχιση χρησιμοποιείται ο αλγόριθμος Smith-Waterman ενώ κατά την συνολική στοίχιση ο Needleman-Wunsch.

Ο υπολογισμός της ποιότητας της στοίχισης, δηλαδή του σκορ στοίχισης, γίνεται με την χρήση συγκεκριμένων πινάκων, γνωστοί ως πίνακες αντικατάστασης. Οι πίνακες αντικατάστασης περιέχουν ένα συγκεκριμένο σκορ στοίχισης για κάθε ζεύγος αμινοξέων και συνήθως ένα σκορ για τα κενά. Το τελικό σκορ είναι το άθροισμα των επιμέρους σκορ των ζευγών αμινοξέων και των κενών που εντοπίζονται.

Η επιλογή του συστήματος υπολογισμού μιας στοίχισης είναι πολύ σημαντική παράμετρος. Ο χρήστης θα πρέπει να αποφασίσει με βάση του τί θέλει να πετύχει με την στοίχιση των αλληλουχιών. Πιθανοί στόχοι είναι η εύρεση δομικών και λειτουργικών ομοιοτήτων, ο εντοπισμός νέων μελών μιας πρωτεϊνικής οικογένειας ή η φυλογενετική σχέση δυο αλληλουχιών. Το σύστημα υπολογισμού της στοίχισης περιλαμβάνει τον τρόπο στοίχισης, τον πίνακα αντικατάστασης και την χρήση ή μη κενών σε περιοχές ανομοιότητας.

2.2.2.2. Αλγόριθμος Smith-Waterman

Ο αλγόριθμος των Smith-Waterman χρησιμοποιείται για την εύρεση περιοχών υψηλής ομοιότητας μεταξύ δυο αλληλουχιών. Είναι ο πιο χρησιμοποιημένος αλγόριθμος δυναμικού προγραμματισμού για την τοπική στοίχιση δυο αλληλουχιών. Η λογική του αλγορίθμου παρουσιάζεται παρακάτω [14]:

Έστω οι δυο αλληλουχίες: $\underset{\sim}{A} = a_1 a_2 a_3 .....a_n$ και $\underset{\sim}{B} = b_1 b_2 b_3 .....b_m$.

Για να βρεθεί ένα τμήμα των αλληλουχιών με μεγάλα ποσοστά ομοιότητας κατασκευάζεται πίνακας H, του οποίου το κάθε στοιχείο είναι μια τιμή για ένα ζεύγος αμινοξέων και ισχύει:

- $H_{i0} = H_{0j} = 0$ για $0 \le i \le n$ και $0 \le j \le m$
- Τα υπόλοιπα στοιχεία του πίνακα H λαμβάνουν μια τιμή σύμφωνα με την σχέση:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j) \\ H_{i-1,j} - g \\ H_{i,j-1} - g \\ 0 \end{cases} \qquad (2.2.1)$$

όπου $s(a_i,b_j)$ είναι το σκορ για το ταίριασμα ή μη ταίριασμα δυο αμινοξέων, g είναι συνήθως σταθερά (σκορ κενού) και το μηδέν υπάρχει ώστε όλα τα στοιχεία να έχουν μη αρνητική τιμή. Η τιμή του $s(a_i,b_j)$ ορίζεται αυθαίρετα, για παράδειγμα +1 αν τα αμινοξέα ταιριάζουν και -0.5 αν δεν ταιριάζουν, ενώ το σκορ κενού πρέπει να είναι τουλάχιστον ίσο με την διαφορά αυτών των τιμών, δηλαδή 1+0.5=1.5.

Για να βρεθεί το τμήμα των δυο αλληλουχιών με την μεγαλύτερη ομοιότητα, αρχικά εντοπίζεται το στοιχείο του πίνακα H, $h_{k,l}$, με την μεγαλύτερη τιμή, από το οποίο ξεκινά ένα μονοπάτι. Για να βρεθεί το επόμενο στοιχείο του μονοπατιού συγκρίνονται τα εξής στοιχεία: $h_{k-1,l-1}$ $h_{k-1,l}$ $h_{k,l-1}$ και επιλέγεται αυτό με την μεγαλύτερη τιμή. Η διαδικασία αυτή ακολουθείται και στα νέα στοιχεία του μονοπατιού και δημιουργείται ένα μονοπάτι προς τα πίσω (traceback) μέχρι να βρεθεί στοιχείο, του οποίου η τιμή είναι μηδέν. Το ζητούμενο τμήμα των δυο αλληλουχιών εντοπίζεται με την διαδρομή που έχει χαραχτεί πάνω στον πίνακα H. Παρακάτω παρουσιάζεται ένα μικρό παράδειγμα, ώστε να κατανοηθεί καλύτερα ο αλγόριθμος των Smith-Waterman:

Έστω οι δυο αλληλουχίες αμινοξέων a και b:

a: A-E-R-A-S-C-I-R-A-F-W-F

b: E-R-A-S-C-R-V-F-F

Θεωρούνται οι τιμές s(a,b)=1 για ταίριασμα, s(a,b)=-0.5 για μη ταίριασμα και g=2. Αρχικά κατασκευάζεται ο πίνακας H:

		A	**E**	**R**	**A**	**S**	**C**	**I**	**R**	**A**	**F**	**W**	**F**
	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0												
R	0												
A	0												

S	0												
C	0												
R	0												
V	0												
F	0												
F	0												

Υπολογίζονται τα στοιχεία $H_{2,2}$, $H_{2,3}$:

$$H_{2,2} = \max\begin{cases} 0+(-0.5) = -0.5 \\ 0-2 = -2 \\ 0-2 = -2 \\ 0 \end{cases} \Rightarrow H_{2,2} = 0, \quad H_{2,3} = \max\begin{cases} 0+1 = 1 \\ 0-2 = -2 \\ 0-2 = -2 \\ 0 \end{cases} \Rightarrow H_{2,3} = 1$$

Μετά από όλους τους υπολογισμούς εντοπίζεται το ζητούμενο μονοπάτι:

		A	**E**	**R**	**A**	**S**	**C**	**I**	**R**	**A**	**F**	**W**	**F**
	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0	0	0	0	0	0
R	0	0	0	2	0	0	0	0	1	0	0	0	0
A	0	1	0	0	3	1	0	0	0	2	0	0	0
S	0	0	0.5	0	1	4	2	0	0	0	1.5	0	0
C	0	0	0	0	0	2	5	3	1	0	0	1	0
R	0	0	0	1	0	0	3	4.5	4	2	0	0	0.5
V	0	0	0	0	0	0	1	2.5	4	3.5	1.5	0	0
F	0	0	0	0	0	0	0	0.5	2	3.5	4	2	1
F	0	0	0	0	0	0	0	0	0	1.5	4.5	2.5	3

Τελικά, το τμήμα με την μεγαλύτερη ομοιότητα είναι:

A E R A S C I R A F W F

\- E R A S C R V F F - -

Στις δυο αλληλουχίες, υπάρχουν και άλλα σημεία-περιοχές ομοιότητας, όμως το σκορ στοίχισης τους είναι μικρότερο από το τμήμα που αναφέρεται παραπάνω.

2.2.3. Μέθοδος k-γραμμάτων

Η μέθοδος k-γραμμάτων είναι μια τεχνική διαχωρισμού μιας αλληλουχίας πρωτεϊνών, DNA ή RNA ανά k γράμματα με σκοπό την πιο αποτελεσματική αναζήτηση όμοιων αλληλουχιών σε μια μεγάλη βάση δεδομένων.

Αν υποθέσουμε την αλληλουχία αμινοξέων AEFQVE και ορίσουμε k=3 τότε προκύπτουν οι εξής λέξεις: AEF, EFQ, FQV, QVE. Κατά την ανάλυση της αλληλουχίας σε μια βάση δεδομένων, ουσιαστικά αναζητούνται τμήματα στις αλληλουχίες της βάσης όμοια με τις τέσσερις λέξεις. Αν υπάρξει κάποια ομοιότητα, τότε ξεκινά η σύγκριση της υπόλοιπης αλληλουχίας. Με τον τρόπο αυτό απορρίπτονται πολύ γρήγορα αλληλουχίες που δεν έχουν καμία ομοιότητα με την εξεταζόμενη. Η μέθοδος αυτή είναι πολύ σημαντική όταν χρησιμοποιούνται βάσεις δεδομένων με δεκάδες χιλιάδες αλληλουχίες, για τον λόγο αυτό, βιοπληροφορικά εργαλεία όπως το BLAST την χρησιμοποιούν.

Η τεχνική των k-γραμμάτων είναι συνήθως πιο αποτελεσματική από τον απλό δυναμικό προγραμματισμό, ενώ για την αξιολόγηση της ποιότητας της στοίχισης που πραγματοποιείται κάθε φορά, χρησιμοποιούνται οι πίνακες αντικατάστασης.

2.2.4. Πίνακες Αντικατάστασης

2.2.4.1 Εισαγωγή

Η λογική αξιολόγησης ενός ζεύγους αμινοξέων που χρησιμοποιείται στον αλγόριθμο Smith-Waterman είναι απλή, υπάρχει ένα σκορ αν τα αμινοξέα ταιριάζουν και μια ποινή αν δεν ταιριάζουν. Όμως με τον τρόπο αυτό, ζεύγη διαφορετικών αμινοξέων, με παρόμοιες χημικές και φυσικοχημικές ιδιότητες, των οποίων τα κωδικόνια μπορεί να διαφέρουν μόνο ως προς ένα νουκλεοτίδιο, ισοδυναμούν με ζεύγη αμινοξέων με τελείως διαφορετικές ιδιότητες. Η ποιότητα της στοίχισης μπορεί να βελτιωθεί, αν εκτός από τα ταυτόσημα αμινοξέα, και ζεύγη «συγγενικών» αμινοξέων

χαρακτηρίζονται με θετικά σκορ. Για τον λόγο αυτό χρησιμοποιείται ένα βελτιωμένο σύστημα αξιολόγησης των ζευγών των αμινοξέων, οι ***πίνακες αντικατάστασης (substitution matrices).***

Υπάρχουν διάφορα είδη πινάκων αντικατάστασης, των οποίων η βασική διαφορά είναι τα δεδομένα με τα οποία υπολογίστηκαν στα σκορ στοίχισης. Το σκορ ενός ζεύγους αμινοξέων, s(a,b), στον πίνακα αντικατάστασης, είναι μια εκτίμηση για τον αν η στοίχιση στο συγκεκριμένο ζεύγος συμβαίνει λόγω ομοιότητας των δυο αλληλουχιών ή κατά τύχη. Είναι δηλαδή το αποτέλεσμα ενός ελέγχου υπόθεσης, σύμφωνα με τον οποίο η μηδενική υπόθεση (H_0) δηλώνει πως οι δυο αλληλουχίες είναι ομόλογες ή έχουν κάποια σχέση ομοιότητας μεταξύ τους και η εναλλακτική υπόθεση (H_1) δηλώνει πως οι δυο αλληλουχίες δεν έχουν καμία σχέση μεταξύ τους και τυχόν ομοιότητες κατά την στοίχιση είναι τυχαία γεγονότα. Η τεχνική αυτή παρουσιάζεται αναλυτικά παρακάτω [16]:

Έστω ότι εξετάζεται το ζεύγος αμινοξέων a και b σε μια στοίχιση δυο αλληλουχιών. Αρχικά ορίζονται οι εξής πιθανότητες:

- ***q_{ab}, πιθανότητα στόχος (target frequency)*** : η πιθανότητα τα αμινοξέα a και b να στοιχίζονται στις δυο αλληλουχίες, γιατί οι αλληλουχίες είναι ομόλογες (πιθανότητα μηδενικής υπόθεσης).
- ***p_a, πιθανότητα υπόβαθρου (background frequency)*** : η πιθανότητα τυχαίας εμφάνισης του αμινοξέος a στην αλληλουχία του παραδείγματος (ουσιαστικά είναι η πιθανότητα εμφάνισης του αμινοξέος a, τυχαία, σε μια μεγάλη ομάδα (block) αλληλουχιών), ομοίως η p_b .
- **$p_a \cdot p_b$**: η πιθανότητα τα αμινοξέα a και b να στοιχίζονται στις δυο αλληλουχίες, οι οποίες δεν είναι ομόλογες και δεν παρουσιάζουν ομοιότητες μεταξύ τους (πιθανότητα εναλλακτικής υπόθεσης).

Το σκορ στοίχισης ή αλλιώς σκορ αντικατάστασης δυο αμινοξέων, s(a,b), έχει τη μορφή λογαριθμικής πιθανότητας ***(log odds scores)***, δηλαδή λογαρίθμου του λόγου δυο πιθανοτήτων, οι οποίες εξετάζουν αν ισχύει μια υπόθεση ή όχι. Στη συγκεκριμένη περίπτωση, το σκορ στοίχισης είναι ο λογάριθμος του λόγου της πιθανότητας να συμβαίνει η στοίχιση του ζεύγους a και b λόγω ομοιλογίας των δυο αλληλουχιών προς την πιθανότητα η στοίχιση να συμβαίνει κατά τύχη, πολλαπλασιαζόμενος με τον αντίστροφο μιας

σταθεράς λ. Η εξίσωση που δίνει το σκορ στοίχισης για κάθε ζεύγος αμινοξέων a,b είναι:

$$s(a,b) = 1/\lambda \times \log \frac{q_{ab}}{p_a \times p_b} \qquad (2.2.2)$$

Η τιμή λ είναι μια σταθερά της κατανομής των σκορ του πίνακα αντικατάστασης, η οποία δεν επηρεάζει το συνολικό αποτέλεσμα της στοίχισης αλληλουχιών καθώς πολλαπλασιάζεται με όλα τα σκορ. Οι τιμές που λαμβάνει είναι συνήθως λ=1, λ=ln2 και λ=2. Επιπλέον, η βάση του λογαρίθμου είναι συνήθως ο e.

Η τιμή του σκορ στοίχισης μπορεί να είναι θετική ή αρνητική. Όταν τα δυο αμινοξέα a και b στοιχίζονται σε ομόλογες αλληλουχίες με συχνότητα μεγαλύτερη από αυτή της τυχαίας τους στοίχισης σε μη σχετικές αλληλουχίες, δηλαδή $q_{ab} > p_a \cdot p_b$, τότε η τιμή είναι θετική. Όταν όμως η στοίχιση συμβαίνει με μεγαλύτερη συχνότητα κατά τύχη, τότε η τιμή είναι αρνητική, καθώς ο λόγος των πιθανοτήτων είναι μικρότερος της μονάδας.

Όμως το σκορ στοίχισης δυο αμινοξέων είναι μια καθαρά στατιστική τιμή και δεν μπορεί να χρησιμοποιηθεί για την εξαγωγή συμπερασμάτων για τις βιοχημικές και δομικές ιδιότητες των αμινοξέων. Για παράδειγμα, στον πίνακα αντικατάστασης BLOSUM62, το ζεύγος αμινοξέων αλανίνη/λευκίνη (A/L) έχει την τιμή -1 ενώ το ζεύγος γλουταμινικό οξύ/λυσίνη (E/K) έχει +1. Επομένως, το ζεύγος E/K παρουσιάζεται πιο συχνό σε ομόλογες αλληλουχίες. Από βιοχημική άποψη, η αλανίνη και η λυσίνη είναι υδρόφοβα, μη πολικά μόρια και οι πλευρικές τους ομάδες είναι παρόμοιες. Το γλουταμινικό οξύ και η λυσίνη είναι υδρόφιλα, πολικά μόρια αλλά η λυσίνη είναι βάση σε αντίθεση με το γλουταμικό οξύ. Επιπλέον οι πλευρικές τους ομάδες διαφέρουν σημαντικά. Φαίνεται λοιπόν ότι τα σκορ στοίχισης των δυο ζευγών δεν αντικατοπτρίζουν τη βιοχημική ομοιότητα των αμινοξέων. Αν αναλυθούν περισσότερο τα δυο ζεύγη, διαπιστώνεται ότι: $q_{AL}=0.0044$ και $q_{EK}=0.0041$. Οι πιθανότητες δηλαδή τα δυο ζεύγη να βρεθούν σε ομόλογες αλληλουχίες και όχι κατά τύχη είναι παρόμοιες. Όμως $p_A \cdot p_L=0.0073$, και $p_K \cdot p_E=0.0031$, επομένως το ζεύγος A/L εμφανίζεται πιο συχνά, κατά τύχη,

απ’ ότι το E/K. Αυτός είναι ο λόγος της διαφοράς των σκορ s(A,L) και s(E,K). [16]

Εκτός από τα σκορ στοίχισης ζευγών αμινοξέων, συχνά χρησιμοποιούνται και ***ποινές για τα κενά (gap penalties)***. Η χρήση κενών στην τοπική στοίχιση αλληλουχιών επιτρέπει την επέκταση της στοίχισης σε περιοχές, όπου δεν υπάρχει ομοιότητα και πιθανότατα έχουν προστεθεί ή αφαιρεθεί αμινοξέα σε μια από τις δυο αλληλουχίες. Με τον τρόπο αυτό μπορεί να ενωθούν περιοχές υψηλής ομοιότητας, οι οποίες χωρίζονται από μικρή ανόμοια αλληλουχία. Η είσοδος ενός κενού αντιστοιχεί με μια ποινή, η οποία μειώνει στο συνολικό σκορ στοίχισης. Η ποινή για το πρώτο κενό είναι συνήθως μεγάλος αριθμός (-9, -11), ενώ για τα κενά που ακολουθούν είναι μικρότερη (-2, -3). Ουσιαστικά τα κενά χρησιμοποιούνται για την κάλυψη μικρών ανομοιοτήτων στις δυο αλληλουχίες, καθώς όσο περισσότερα είναι τόσο μικρότερο ποσοστό ομοιότητας έχουν οι δυο αλληλουχίες και το σκορ στοίχισης μειώνεται σημαντικά. Για το λόγο αυτό, όταν η στοίχιση έχει σαν σκοπό την εύρεση περιοχών υψηλής ομοιότητας, χρησιμοποιούνται μεγάλες ποινές, ενώ αντίθετα, όταν η στοίχιση αποσκοπεί στην εύρεση απομακρυσμένων συγγενικά αλληλουχιών, χρησιμοποιούνται μικρές σχετικά ποινές.

Ένα βασικό ερώτημα για την κατασκευή ενός πίνακα αντικατάστασης είναι τί δεδομένα πρέπει να χρησιμοποιηθούν για τον υπολογισμού της πιθανότητας στόχου (target frequency) κάθε ζεύγους αμινοξέων. Η βασική ιδέα είναι η συλλογή ενός μεγάλου όγκου όμοιων-σχετικών αλληλουχιών από τις οποίες, με στοίχιση κατά ζεύγη, θα προκύψουν οι ζητούμενες πιθανότητες. Με τον τρόπο αυτό, διάφορες ερευνητικές ομάδες έχουν κατασκευάσει πίνακες αντικατάστασης, από τους οποίους οι πιο ευρέως χρησιμοποιημένοι είναι οι PAM και οι BLOSUM πίνακες.

2.2.4.2. PAM Πίνακες Αντικατάστασης

Οι **PAM πίνακες** βασίζονται στην ιδέα της ***αποδεκτής σημειακής μετάλλαξης (Point Accepted Mutation)***. Η σημειακή μετάλλαξη πραγματοποιείται στο DNA (αντικατάσταση ενός νουκλεοτιδίου με ένα άλλο)

και μπορεί να οδηγήσει, μέσα από τις διαδικασίες της μεταγραφής και μετάφρασης, στην αντικατάσταση ενός αμινοξέος με ένα άλλο σε μια πρωτεϊνική αλληλουχία. Για να είναι όμως αποδεκτή θα πρέπει να γίνει αφομοίωση της μετάλλαξης στα κύτταρα-απογόνους. Συνήθως, η αποδεκτή σημειακή μετάλλαξη αφορά αμινοξέα που έχουν παρόμοιες χημικές και φυσικοχημικές ιδιότητες. Για να διαπιστωθεί η συχνότητα με την οποία μεταλλάσσεται ένα αμινοξύ, αλλά και από ποια αμινοξέα αντικαθίσταται, πρέπει να εξεταστεί η εξελικτική πορεία διαφόρων οικογενειών πρωτεϊνών. [18][20]

Με βάση την λογική αυτή, ερευνητική ομάδα κατασκεύασε 71 φυλογενετικά δέντρα πρωτεϊνών, οι οποίες προέρχονταν από 34 υπεροικογένειες. Στην συνέχεια, οι πρωτεΐνες του ίδιου φυλογενετικού δέντρου συγκρίθηκαν με την μέθοδο της συνολικής στοίχισης, ώστε να βρεθούν τυχόν αλλαγές στα αμινοξέα. Με σκοπό την ελαχιστοποίηση του ευρισκόμενου αριθμού των μεταλλάξεων κατά την σύγκριση των αλληλουχιών, οι αλληλουχίες ενός φυλογενετικού δέντρου ήταν όμοιες σε ποσοστό πάνω από 85%. Το τελικό αποτέλεσμα από την εξέταση όλων των φυλογενετικών δέντρων ήταν ο αριθμός των 1572 αποδεκτών σημειακών μεταλλάξεων. [18]

Με τον τρόπο αυτό, βρέθηκαν, για μια μεγάλη βάση ομόλογων αλληλουχιών, σημαντικές πληροφορίες για την πιθανότητα αντικατάστασης ενός αμινοξέος a από ένα b. Το αποτέλεσμα αυτών των υπολογισμών ήταν η εύρεση της πιθανότητας στόχου για κάθε ζεύγος αμινοξέων. Στην συνέχεια κατασκευάστηκε ο PAM1 πίνακας.

Κάθε στοιχείο του PAM1 είναι το σκορ αντικατάστασης ενός αμινοξέος a από ένα αμινοξύ b σε μια αλληλουχία 100 αμινοξέων, όταν πραγματοποιείται μία μόνο αποδεκτή σημειακή μετάλλαξη στο γονίδιο. Όταν ο PAM1 πολλαπλασιαστεί N φορές με τον εαυτό του, τότε κατασκευάζεται ο PAM N. Έτσι κατασκευάστηκαν οι πίνακες PAM120, PAM250 και PAM500, οι οποίοι είναι οι πιο ευρέως χρησιμοποιημένοι αυτής της οικογένειας. [18][12]

```
# PAM 250 substitution matrix, scale = ln(2)/3 = 0.231049
#
# Expected score = -0.844, Entropy = 0.354 bits
#
# Lowest score = -8, Highest score = 17
#
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
A  2 -2  0  0 -2  0  0  1 -1 -1 -2 -1 -1 -3  1  1  1 -6 -3  0
R -2  6  0 -1 -4  1 -1 -3  2 -2 -3  3  0 -4  0  0 -1  2 -4 -2
N  0  0  2  2 -4  1  1  0  2 -2 -3  1 -2 -3  0  1  0 -4 -2 -2
D  0 -1  2  4 -5  2  3  1  1 -2 -4  0 -3 -6 -1  0  0 -7 -4 -2
C -2 -4 -4 -5 12 -5 -5 -3 -3 -2 -6 -5 -5 -4 -3  0 -2 -8  0 -2
Q  0  1  1  2 -5  4  2 -1  3 -2 -2  1 -1 -5  0 -1 -1 -5 -4 -2
E  0 -1  1  3 -5  2  4  0  1 -2 -3  0 -2 -5 -1  0  0 -7 -4 -2
G  1 -3  0  1 -3 -1  0  5 -2 -3 -4 -2 -3 -5  0  1  0 -7 -5 -1
H -1  2  2  1 -3  3  1 -2  6 -2 -2  0 -2 -2  0 -1 -1 -3  0 -2
I -1 -2 -2 -2 -2 -2 -2 -3 -2  5  2 -2  2  1 -2 -1  0 -5 -1  4
L -2 -3 -3 -4 -6 -2 -3 -4 -2  2  6 -3  4  2 -3 -3 -2 -2 -1  2
K -1  3  1  0 -5  1  0 -2  0 -2 -3  5  0 -5 -1  0  0 -3 -4 -2
M -1  0 -2 -3 -5 -1 -2 -3 -2  2  4  0  6  0 -2 -2 -1 -4 -2  2
F -3 -4 -3 -6 -4 -5 -5 -5 -2  1  2 -5  0  9 -5 -3 -3  0  7 -1
P  1  0  0 -1 -3  0 -1  0  0 -2 -3 -1 -2 -5  6  1  0 -6 -5 -1
S  1  0  1  0  0 -1  0  1 -1 -1 -3  0 -2 -3  1  2  1 -2 -3 -1
T  1 -1  0  0 -2 -1  0  0 -1  0 -2  0 -1 -3  0  1  3 -5 -3  0
W -6  2 -4 -7 -8 -5 -7 -7 -3 -5 -2 -3 -4  0 -6 -2 -5 17  0 -6
Y -3 -4 -2 -4  0 -4 -4 -5  0 -1 -1 -4 -2  7 -5 -3 -3  0 10 -2
V  0 -2 -2 -2 -2 -2 -2 -1 -2  4  2 -2  2 -1 -1 -1  0 -6 -2  4
```

Εικόνα 2.2.2: PAM 250.

2.2.4.3. BLOSUM Πίνακες Αντικατάστασης

Οι ***πίνακες BLOSUM (BLOck SUbstitution Matrix)*** βασίζονται στον ίδιο κανόνα με τους PAM, ότι δυο αλληλουχίες, οι οποίες προέρχονται από τον ίδιο απόγονο, έχουν ομοιότητες μεταξύ τους (εξελικτική σχέση μεταξύ των αλληλουχιών). Όμως οι BLOSUM πίνακες κατασκευάστηκαν με στόχο να μπορούν να χρησιμοποιηθούν επιτυχώς, για τον εντοπισμό μακρινών ομόλογων αλληλουχιών. Οι PAM πίνακες, λόγω του ότι κατασκευάστηκαν από την ανάλυση όμοιων αλληλουχιών (σε ποσοστό άνω του 85%) παρουσιάζουν προβλήματα στο να αξιολογήσουν σωστά μια πιο μακρινή ομοιότητα μεταξύ δυο αλληλουχιών, π.χ. σε ποσοστό 50%. Το πρόβλημα αυτό προσπάθησε να λύσει η ερευνητική ομάδα που κατασκεύασε τους πίνακες BLOSUM. [19]

Το αρχικό βήμα στην κατασκευή των BLOSUM πινάκων ήταν η εύρεση κατάλληλων αλληλουχιών με σκοπό να αναλυθούν. Για τον λόγο αυτό συγκεντρώθηκε ένας μεγάλος όγκος γνωστών αλληλουχιών, οι οποίες είχαν σχέσεις υψηλής αλλά και χαμηλής ομοιότητας μεταξύ τους. Για την εύρεση των πιθανοτήτων στόχων, οι αλληλουχίες που ήταν όμοιες μέχρι ένα συγκεκριμένο ποσοστό (π.χ. 62%) συγκεντρώνονταν σε μια βάση, η οποία ονομαζόταν BLOCK. Πραγματοποιώντας πολλαπλή στοίχιση (multiple alignment) στο BLOCK, υπολογίζονταν οι ζητούμενες πιθανότητες και κατασκευαζόταν ο αντίστοιχος πίνακας (π.χ. BLOSUM62). Με τον τρόπο αυτό έχουν κατασκευαστεί όλοι οι πιθανοί πίνακες BLOSUM1 έως BLOSUM100. Οι πίνακες που χρησιμοποιούνται περισσότερο είναι ο BLOSUM62, BLOSUM50 και BLOSUM80. [16][19]

```
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 62
#  Entropy =   0.6979, Expected =  -0.5209
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4 -1 -3 -3 -1  0 -1 -4 -3 -3
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1
Q -1  1  0  0 -3  5  2 -2  0 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2
E -1  0  0  2 -4  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4 -2 -3 -3 -2  0 -2 -2 -3 -3
H -2  0  1 -1 -3  0  0 -2  8 -3 -3 -1 -2 -1 -2 -1 -2 -2  2 -3
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2 -3  1  0 -3 -2 -1 -3 -1  3
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5 -1 -3 -1  0 -1 -3 -2 -2
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5  0 -2 -1 -1 -1 -1  1
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5 -2 -2  0
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11  2 -3
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4
```

Εικόνα 2.2.3: BLOSUM 62.

2.2.4.4. Σχετική Εντροπία

Όπως αναφέρθηκε, οι BLOSUM πίνακες κατασκευάστηκαν με σκοπό να αντιμετωπιστεί καλύτερα το πρόβλημα της εύρεσης μακρινών ομόλογων αλληλουχιών. Η βασική λοιπόν διαφορά των PAM και BLOSUM πινάκων είναι η βάση δεδομένων των αλληλουχιών, από την οποία υπολογίστηκαν οι πιθανότητες στόχοι. Επιπλέον, οι PAM πίνακες κατασκευάστηκαν με βάση την συνολική ανάλυση των αλληλουχιών, δηλαδή την πραγματοποίηση συνολικής στοίχισης, ενώ στους BLOSUM πίνακες εξετάσθηκαν μόνο περιοχές ομοιότητας μεταξύ των αλληλουχιών, οι οποίες διέφεραν σε μέγεθος ανάλογα με το συνολικό ποσοστό ομοιότητας. Για τους λόγους αυτούς, οι BLOSUM πίνακες έχουν συνήθως καλύτερη απόδοση από τους PAM, όταν αναζητούνται ομοιότητες και πιθανές ομολογίες μεταξύ γνωστών και άγνωστων αλληλουχιών με τοπική στοίχιση.

Όμως, παρότι οι δυο οικογένειες πινάκων έχουν διαφορές μεταξύ τους ως προς τον τρόπο με τον οποίο κατασκευάστηκαν, υπάρχει ένα μέτρο το οποίο μπορεί να εξισώσει έναν πίνακα PAM με έναν BLOSUM ως προς την σημαντικότητα της πληροφορίας, την οποία παρέχουν σε κάθε στοίχιση δυο αμινοξέων. Το μέτρο αυτό ονομάζεται ***σχετική εντροπία (relative entropy)***, η μονάδα μέτρησης του οποίου εξαρτάται από την σταθερά κατανομής λ των σκορ αντικατάστασης (συνήθως είναι το 1bit => λ=ln2), και ορίζεται ως εξής:

$$H = \sum_{a=1}^{20} \sum_{b=1}^{20} (q_{ab} \times s_{ab}) \qquad (2.2.3)$$

όπου q_{ab} είναι η πιθανότητα στόχος για το ζεύγος αμινοξέων a,b και s_{ab} το αντίστοιχο σκορ αντικατάστασης. [12][16]

Θα πρέπει να διευκρινιστεί πως τα στοιχεία των πινάκων αντικατάστασης κατά την σύγκριση δυο αλληλουχιών ερμηνεύονται ως εξής: το θετικό σκορ δηλώνει πιθανή ομοιότητα των δυο αλληλουχιών και το αρνητικό ανομοιότητα. Η ερμηνεία αυτή προκύπτει από τον ορισμό του σκορ αντικατάστασης. Επομένως, για να αποδειχτεί ομολογία πρέπει να τελικό σκορ να είναι όσο το δυνατόν μεγαλύτερο. Όμως κάθε πίνακας της ίδιας

οικογένειας δίνει διαφορετικό σκορ για ένα συγκεκριμένο ζεύγος. Τίθενται λοιπόν τα ερωτήματα, ποιο είναι το κατάλληλο μήκος στοίχισης που εξασφαλίζει ασφαλή συμπεράσματα και ποιός πίνακας είναι κατάλληλος κάθε φορά. Για να απαντηθούν, χρησιμοποιείται η σχετική εντροπία.

Η σχετική εντροπία μεταφράζεται ως η μέση ποσότητα της πληροφορίας που παρέχει κάθε μια στοίχιση στη σύγκριση δυο αλληλουχιών. Επομένως, είναι ένας τρόπος να βρεθεί αν το μέγεθος της πληροφορίας που δίνει η ολική στοίχιση είναι αρκετό, ώστε να εξαχθούν ασφαλή συμπεράσματα για την ομοιότητα/ανομοιότητα των δυο αλληλουχιών. Όπως προκύπτει, αν έχουμε να συγκρίνουμε μικρές αλληλουχίες, θέλουμε έναν πίνακα ο οποίος σε κάθε στοίχιση θα δίνει μεγάλο όγκο πληροφορίας, δηλαδή θα έχει μεγάλη σχετική εντροπία. Αντίθετα, οι μεγάλες αλληλουχίες χρειάζονται μικρή σχετική εντροπία. Για να κατανοηθεί καλύτερα η έννοια της σχετικής εντροπίας, παρουσιάζονται στον παρακάτω πίνακα οι σχετικές εντροπίες για ορισμένους PAM πίνακες:

ΠΙΝΑΚΑΣ 2.2.1: Σχετική εντροπία PAM πινάκων

PAM	H(bits)	Ελάχιστο απαιτούμενο μήκος (30 bits)	PAM	H(bits)	Ελάχιστο απαιτούμενο μήκος (30 bits)
10	3.43	9	220	0.45	68
50	2.00	15	250	0.36	83
100	1.18	26	300	0.27	113
120	0.98	31	320	0.24	127
160	0.70	43	350	0.20	149

[17]

Αν θεωρηθεί ότι 30 bits πληροφορίας είναι αρκετά, για τον πίνακα PAM10 χρειάζονται τουλάχιστον 9 θέσεις στοίχισης. Άρα ο PAM10 μπορεί να χρησιμοποιηθεί σε αλληλουχίες 10-15 αμινοξέων, όπου μια περιοχή 9 αμινοξέων υψηλής ομοιότητας αρκεί για να αποδειχτεί ομολογία των δυο αλληλουχιών. Με την ίδια λογική, ο PAM350 μπορεί να χρησιμοποιηθεί για αλληλουχίες 400+ αμινοξέων, όπου αν συγκριθούν τουλάχιστον τα 149, τα αποτελέσματα που προκύπτουν θεωρούνται ασφαλή.

Σύμφωνα με το EBI (European Bioinformatics Institute) οι πίνακες αντικατάστασης κατά την ανάλυση άγνωστων αλληλουχιών, μπορούν να χρησιμοποιηθούν ως εξής:

ΠΙΝΑΚΑΣ 2.2.2: Συσχέτιση πινάκων αντικατάστασης

Μήκος αλληλουχίας αμινοξέων	**Κατάλληλος πίνακας αντικατάστασης**	
>300	BLOSUM 50	PAM 250
85-300	BLOSUM 62	PAM 120
50-85	BLOSUM 80	

[21]

Θα πρέπει να σημειωθεί πως η επιλογή του πίνακα αντικατάστασης δεν εξαρτάται μόνο από το μήκος των εξεταζόμενων αλληλουχιών αλλά και τον σκοπό της εξέτασης. Το EBI προτείνει τους παραπάνω πίνακες για την σύγκριση άγνωστων αλληλουχιών με ένα μεγάλο όγκο δεδομένων, όπου θα πρέπει να υπάρχει η δυνατότητα να βρεθούν και μικρές ομοιότητες (20%) ή μακρινές φυλογενετικές σχέσεις. Αν ο χρήστης επιθυμεί την εύρεση ταυτόσημων αλληλουχιών με τις άγνωστες, δηλαδή ποσοστό ομοιότητας >85%, τότε θα πρέπει να χρησιμοποιηθούν πιο «ευαίσθητοι» πίνακες. Στην περίπτωση των BLOSUM, χρειάζονται πίνακες που έχουν κατασκευαστεί από αλληλουχίες υψηλής ομοιότητας (π.χ. BLOSUM90) και στην περίπτωση των PAM από τον ισχυρισμό μικρού αριθμού σημειακών μεταλλάξεων (π.χ. PAM 60).

2.3. ΠΟΛΛΑΠΛΗ ΣΤΟΙΧΙΣΗ

Η πολλαπλή στοίχιση, δηλαδή η στοίχιση τριών ή περισσότερων αλληλουχιών, χρησιμοποιείται κυρίως με σκοπό τη εύρεση ομοιοτήτων σε έναν πλήθος αλληλουχιών. Στην περίπτωση των πρωτεϊνών, οι ομοιότητες μπορεί να αναζητούνται σε συγκεκριμένες θέσεις αμινοξέων, σε πρωτεϊνικές περιοχές που έχουν δομική και λειτουργική σημασία για το μόριο (domains) ή σε ολόκληρες τις αλληλουχίες. Συνήθως οι αλληλουχίες που στοιχίζονται έχουν φυλογενετικές σχέσεις και αναζητούνται οι κοινές τους περιοχές ή οι μεταλλάξεις που έχουν συμβεί κατά την εξέλιξη. Επιπλέον η πολλαπλή στοίχιση μπορεί να χρησιμοποιηθεί για ανάλυση αλληλουχιών όμοιας λειτουργίας, με σκοπό την εύρεση των περιοχών που ορίζουν την συγκεκριμένη λειτουργία και την μετέπειτα χρήση των αποτελεσμάτων για ταξινόμηση άγνωστων αλληλουχιών. Το αποτέλεσμα της πολλαπλής στοίχισης είναι μια νέα αλληλουχία, η οποία αποτελεί την πιο πιθανή αλληλουχία-πρότυπο για όσες στοιχίζονται.

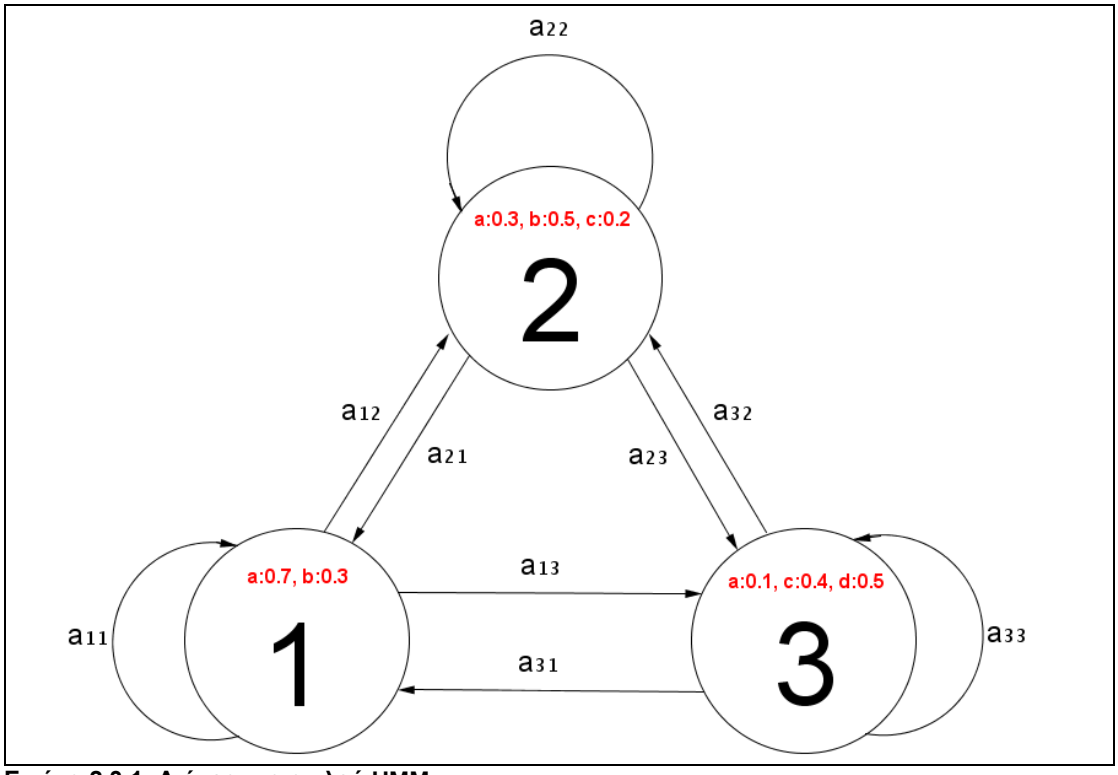
Μια μαθηματική μέθοδος που χρησιμοποιείται, μεταξύ άλλων, στην κατασκευή πολλαπλών στοιχίσεων και την εύρεση πρωτεϊνικών περιοχών ή μοτίβων είναι τα ***Κρυφά Μαρκοβιανά Μοντέλα (Hidden Markov Models)***. Ως Μαρκοβιανή αλυσίδα καλείται η στοχαστική ανέλιξη $\{X(t)\}, t \in T$, για την οποία ισχύσει η σχέση:

$$\begin{aligned} &P\left(X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}, X(t_{n-2}) = x_{n-2}, \ldots, X(t_1) = x_1\right) = \\ &= P\left(X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}\right) \quad \text{για κάθε } t \in T,\ x \in \Omega \text{ και } n > 2. \end{aligned} \tag{2.3.1}$$

δηλαδή η τιμή της $\{X(t)\}$, σε μια τυχαία χρονική στιγμή n, εξαρτάται μόνο από την τιμή της την στιγμή n-1 και όχι από την διαδρομή μέχρι την κατάσταση $X(t_{n-1})$ [22]. Τα Κρυφά Μαρκοβιανά Μοντέλα περιλαμβάνουν μια Μαρκοβιανή αλυσίδα $\{X(t)\}, t \in T$ η οποία λαμβάνει διακριτές τιμές $\{x_1, x_2, \ldots x_n\} = \Omega$. Ονομάζονται κρυφά γιατί η Μαρκοβιανή αλυσίδα δεν είναι

ορατή κατά την εξέλιξή της. Αυτό που είναι ορατό είναι μια στοχαστική ανέλιξη $\{Y(t)\}, t \in T$, της οποίας η κατανομή των τιμών, κάθε στιγμή t, εξαρτάται από την τιμή της $\{X(t)\}$. [23]

Ένα γενικό παράδειγμα Κρυφού Μαρκοβιανού Μοντέλου είναι το παρακάτω. Η Μαρκοβιανή αλυσίδα $\{X(t)\}$ λαμβάνει τρεις διαφορετικές τιμές (καταστάσεις, states), 1,2 και 3. Η πιθανότητα μετάβασης από μια κατάσταση i σε μια j συμβολίζεται με a_{ij}. Οι πιθανές τιμές της $\{Y(t)\}$ είναι a,b,c, και d. Η κάθε κατάσταση της $\{X(t)\}$ αντιστοιχεί σε μια συγκεκριμένη κατανομή των τιμών της $\{Y(t)\}$.



Εικόνα 2.3.1: Διάγραμμα απλού HMM.

Μια πιθανή αλυσίδα καταστάσεων θα ήταν η εξής: $1 \to 3 \to 3 \to 2 \to 2$. Η διαδικασία όμως αυτή δεν θα ήταν εμφανής. Οι καταστάσεις 1, 2 και 3 θα εκφράζονταν μέσα από τις τιμές a,b,c, και d. Επομένως πιθανή ορατή διαδικασία θα ήταν η: $a \to c \to d \to b \to c$. Δυο χρήσεις των Κρυφών Μαρκοβιανών Μοντέλων είναι η αποκρυπτογράφηση μιας αλληλουχίας,

δηλαδή η εύρεση της πιο πιθανής αλληλουχίας καταστάσεων της Μαρκοβιανής αλυσίδας και η εκτίμηση της πιο πιθανής αλληλουχίας τιμών της $\{Y(t)\}$.

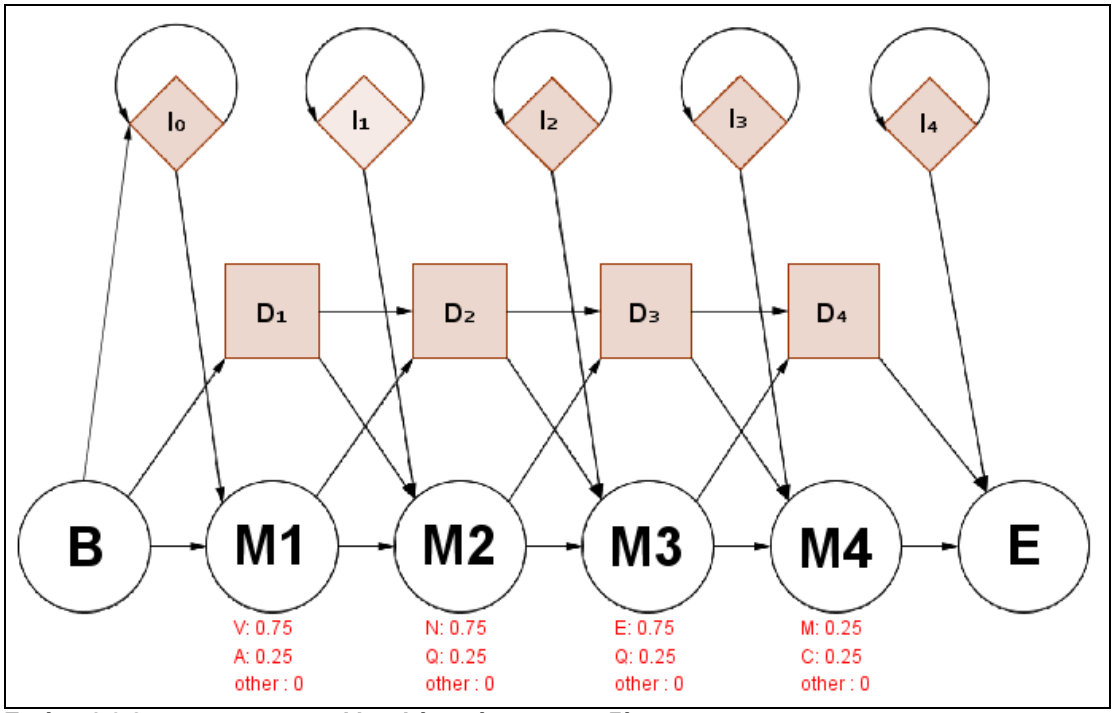
Στην περίπτωση των πρωτεϊνικών αλληλουχιών, τα στοιχεία του Κρυφού Μαρκοβιανού Μοντέλου είναι τα εξής [25]:

- Τύποι Κρυφών καταστάσεων (states): 1) ταίριασμα (match), 2) προσθήκη αμινοξέος (insertion), 3) διαγραφή αμινοξέων (deletion)
- Πίνακας πιθανοτήτων μετάβασης: $A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ ... & ... & ... \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix}$
- Πιθανότητες αρχικής κατάστασης: p_{01}, p_{02}, p_{03}
- Σύνολο ορατών συμβόλων: τα 20 αμινοξέα
- Πιθανότητες εκπομπής (emission), δηλαδή η κατανομή των πιθανοτήτων των αμινοξέων σε κάθε κατάσταση: $e_{st} = P(\text{amino_acid} = X_t \mid \text{state}_i)$

Ένα απλό μοντέλο πολλαπλής στοίχισης αλληλουχιών είναι το παρακάτω:

Πεπτίδιο 1	V	N	E	.
Πεπτίδιο 2	V	Q	E	M
Πεπτίδιο 3	V	N	Q	.
Πεπτίδιο 4	A	N	E	C

Το μοντέλο HMM που αναπτύσσεται με σκοπό την στοίχιση των τεσσάρων πεπτιδίων έχει την εξής μορφή:



Εικόνα 2.3.2: HMM για την πολλαπλή στοίχιση πεπτιδίων.

Το μονοπάτι ξεκινά από την κατάσταση B (Beginning) και με βάση τις αρχικές πιθανότητες μεταβαίνει στην επόμενη κατάσταση (σχεδόν πάντα στη M1). Κάθε κατάσταση M (match) ορίζει μια κατανομή των αμινοξέων. Για παράδειγμα η κατάσταση M1 θα εκπέμψει κατά 75% το αμινοξύ βαλίνη, V. Το μονοπάτι που περνά μόνο από τις καταστάσεις M, δημιουργεί μια νέα αλληλουχία χωρίς κενά και διαγραφές στο μήκος των μεγαλύτερων εξεταζόμενων αλληλουχιών. Όμως ορισμένες φορές η νέα αλληλουχία περιέχει κενά, προστιθέμενα αμινοξέα που δεν εκπέμπονται από τις καταστάσεις M ή το μήκος της δεν είναι το μέγιστο δυνατό. Η κατάσταση I (insertion) ορίζει και αυτή μια κατανομή αμινοξέων με την οποία ένα νέο αμινοξύ προστίθεται στην αλληλουχία. Η κατανομή αυτή είναι συνήθως διαφορετική από αυτή της αντίστοιχης κατάστασης M και η λογική υπολογισμού της, στους αλγόριθμους HMMs που έχουν κατασκευαστεί, είναι διαφορετική. Η κατάσταση D (deletion) αφαιρεί αμινοξέα από την νέα αλληλουχία. Επιπλέον ορισμένοι αλγόριθμοι τοποθετούν και κενά στην νέα αλληλουχία. Συμπερασματικά, το μονοπάτι υπολογισμού της πολλαπλής στοίχισης δεν είναι πάντα μοναδικό και ο τρόπος υπολογισμού του διαφέρει από αλγόριθμος σε αλγόριθμο. Έχουν αναπτυχθεί πολλά λογισμικά (BLOCKS, META-MEME, profile HMM, HMMER) τα οποία χρησιμοποιούν τα

HMMs με διαφορετικό σχεδιασμό το καθένα [24]. Στην παρούσα εργασία δεν κατασκευάστηκε κάποιο HMM με σκοπό την πολλαπλή στοίχιση. Για τον λόγο αυτό δεν θα αναφερθούν οι αλγόριθμοι υπολογισμού πολλαπλών στοιχίσεων.

Κατά την πολλαπλή στοίχιση αλληλουχιών, όπως είναι γνωστό, προκύπτει μια νέα αλληλουχία, η οποία περιέχει τις περιοχές υψηλής ομοιότητας των εξεταζόμενων. Όμως, κάθε θέση στη νέα αλληλουχία δεν αντιστοιχεί αποκλειστικά σε ένα μόνο αμινοξύ. Κάθε θέση χαρακτηρίζεται από μια κατανομή αμινοξέων, η οποία έχει υπολογιστεί κατά την πολλαπλή στοίχιση. Επομένως για κάθε πολλαπλή στοίχιση, κατασκευάζεται ένας ***Πίνακας Τιμών Συγκεκριμένης Θέσης (Position Specific Scoring Matrix, PSSM)***, του οποίου οι στήλες αναφέρονται στα 20 διαφορετικά αμινοξέα και οι γραμμές στις συγκεκριμένες θέσεις του μοντέλου πολλαπλής στοίχισης. Το κάθε σκορ (λογαριθμική πιθανότητα) δηλώνει την πιθανότητα το αμινοξύ να βρίσκεται στην συγκεκριμένη θέση λόγω ομολογίας ή κατά τύχη. Τα λογισμικά που χρησιμοποιούν Κρυφά Μαρκοβιανά Μοντέλα, κατά την ανάλυση άγνωστων αλληλουχιών με βάσεις δεδομένων που έχουν προκύψει από πολλαπλές στοιχίσεις αλληλουχιών, χρησιμοποιούν τους συγκεκριμένους πίνακες στην στοίχιση.

3

ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

«Ότι κάθε γνώση μας αρχίζει από την εμπειρία, αυτό δεν επιδέχεται καμία αμφιβολία· γιατί με τί άλλο θα μπορούσε να αφυπνιστεί η γνωστική μας δύναμη για να ασκησεί το έργο της, αν όχι με αντικείμενα που ερεθίζουν τις αισθήσεις μας και που πότε προκαλούν από μόνα τους τη γέννηση παραστάσεων και πότε βάζουν τη νοητική μας ενέργεια σε κίνηση να τις συγκρίνει, να τις συνδέσει ή να τις χωρίσει και έτσι να κατεργαστεί το άμορφο υλικό των κατ' αίσθηση εντυπώσεων για το σχηματισμό μιας γνώσης των αντικειμένων που ονομάζεται εμπειρία;»

Immanuel Kant, «*Κριτική του Καθαρού Λόγου*»

3.1. ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

3.1.1. Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

Ο ανθρώπινος εγκέφαλος, όργανο εξαιρετικά πολύπλοκο, αποτελεί το σημαντικότερο τμήμα του νευρικού συστήματος, καθώς ενεργοποιεί, ελέγχει και ρυθμίζει τις λειτουργίες όλων των υπόλοιπων οργάνων και τμημάτων του σώματος. Πέρα από τη ρύθμιση βιολογικών και κινητικών λειτουργιών, μπορεί να εκτελεί με επιτυχία μια πληθώρα διαφορετικών λειτουργιών, το σύνολο των οποίων αποτελεί τη ***νοημοσύνη (intelligence)*** του. Τέτοιες λειτουργίες είναι η αναγνώριση εικόνων, η αναγνώριση και παραγωγή ομιλίας, η κατάστρωση σχεδίου για την εύρεση της βέλτιστης επιλογής, ο προσανατολισμός, η ανάπτυξη λόγου και επιχειρημάτων, η λογική σκέψη, η μάθηση και η προσαρμογή σε νέα περιβάλλοντα. [28]

Η ***μάθηση (learning)*** είναι ίσως μια από τις σημαντικότερες λειτουργίες του εγκεφάλου (όχι μόνο του ανθρώπινου, αλλά και των ζώων). Αποτελεί κλειδί της νοημοσύνης, γιατί οι περισσότερες λειτουργίες που εκτελεί ο εγκέφαλος «μαθαίνονται» κατά της διάρκεια της ζωής. Επομένως η μάθηση είναι απαραίτητη για την βελτίωση της νοημοσύνης του ανθρώπου και των ζώων.

Στο καθημερινό λεξιλόγιο η έννοια της μάθησης μπορεί να χρησιμοποιηθεί για διάφορες διαδικασίες. Για παράδειγμα, δεχόμενοι ότι αναφερόμαστε στον άνθρωπο, ως μάθηση μπορεί να χαρακτηριστεί η απόκτηση γνώσης μέσα από την μελέτη, την εμπειρία, την εκπαίδευση ή την παρατήρηση, η ενημέρωση μέσω της ανταλλαγής πληροφοριών, η ανάπτυξη επιδεξιότητας και τεχνικής κατάρτισης και η επιτυχής προσαρμογή σε νέες συνθήκες. Το αποτέλεσμα της μάθησης είναι η βελτίωση του ανθρώπου σε ένα συγκεκριμένο τομέα ή σε μια λειτουργία. Εκτός όμως από τον άνθρωπο, λειτουργίες εκτελούν και οι υπολογιστικές μηχανές.

Για την εκτέλεση των λειτουργιών που έχουν αναφερθεί παραπάνω και την βελτίωση της νοημοσύνης, ο άνθρωπος και τα ζώα διαθέτουν το νευρικό σύστημα. Αντίθετα, οι υπολογιστικές μηχανές, με σκοπό να αναπτύξουν την

τεχνητή νοημοσύνη (artificial intelligence) διαθέτουν κατάλληλα υλικά όπως ημιαγωγά στοιχεία και τρανζίστορ [28]. Σκοπός της τεχνητής νοημοσύνης είναι η ανάπτυξη αλγορίθμων, οι οποίοι θα μιμούνται τις λειτουργίες του ανθρώπινου εγκεφάλου, όπως για παράδειγμα την μάθηση.

Ως ***μηχανική μάθηση (machine learning)***, η οποία αποτελεί σημαντικό τμήμα της τεχνητής νοημοσύνης, περιγράφεται η εκπαίδευση μιας υπολογιστικής μηχανής, με την χρήση δεδομένων, με σκοπό να αποκτήσει εμπειρία και να βελτιώσει την αποτελεσματικότητά της σε μια συγκεκριμένη λειτουργία. Τέτοιες λειτουργίες μπορεί να είναι η παραγωγή ομιλίας, η αυτόματη πλοήγηση ενός ρομπότ, η εύρεση επιτυχίας/αποτυχίας μιας επένδυσης, η πρόγνωση του καιρού, ο σχεδιασμός ενός ηλεκτρονικού παιχνιδιού στρατηγικής κ.α. Η μηχανική μάθηση έχει χρησιμοποιηθεί και στον χώρο της γενομικής-πρωτεομικής ανάλυσης. Ενδεικτικά, τεχνικές μηχανικής μάθησης έχουν εφαρμοστεί για την ταξινόμηση ενζύμων με βάση τα InterPro χαρακτηριστικά τους [29], τον προσδιορισμό οντολογιών γονιδίων (GO terms) χρησιμοποιώντας ετερογενή χαρακτηριστικά πρωτεϊνών (ομοιότητες αλληλουχίας, πρωτεϊνικές περιοχές, δομικά μοτίβα, βιοχημικές ιδιότητες) [30] και τον προσδιορισμό ενζυμικής λειτουργίας με την εξόρυξη δεδομένων από διάφορα χαρακτηριστικά (ομοιότητες αλληλουχίας, βιοχημικές ιδιότητες, φυλογενετικές σχέσεις) [31]. Όπως προκύπτει, η τεχνική της μηχανικής μάθησης μπορεί να εφαρμοστεί για εντελώς διαφορετικές λειτουργίες.

Η εκπαιδευμένη υπολογιστική μηχανή μπορεί να χαρακτηριστεί ως ένα έμπειρο σύστημα. Ένα έμπειρο σύστημα θεωρείται σαν η ενσωμάτωση στον υπολογιστή μιας βασισμένης στη γνώση συνιστώσας κατά τέτοιο τρόπο ώστε να μπορεί να δίνει μια νοήμονα συμβουλή ή να παίρνει ευφυείς αποφάσεις σχετικά με κάποια λειτουργία επεξεργασίας. Ένα επιπλέον απαραίτητο χαρακτηριστικό του έμπειρου συστήματος είναι η ικανότητα να αιτιολογεί κατανοητά τον τρόπο σκέψης του. [32]

Όμως, τίθεται το ερώτημα, γιατί θα πρέπει να χρησιμοποιηθεί η μηχανική μάθηση και δεν κατασκευάζονται από την αρχή συστήματα, τα οποία θα έχουν καλή απόδοση; Οι λόγοι είναι σημαντικοί [32][33]:

- Πολλές φορές υπάρχουν “κρυφές” πληροφορίες μέσα σε ένα μεγάλο όγκο δεδομένων και τα έμπειρα συστήματα μπορούν να τις εξάγουν (data mining).

- Για ορισμένες εργασίες ή διαδικασίες δεν υπάρχουν γνωστές συναρτήσεις ή θεωρίες. Για να προβλεφθεί το αποτέλεσμα της εργασίας, πρέπει αρχικά να γίνει μελέτη όμοιων, γνωστών παραδειγμάτων, ώστε να αποκτηθεί εμπειρία.
- Όταν τα δεδομένα περιέχουν θόρυβο ή η επεξεργασία τους είναι δύσκολη, το έμπειρο σύστημα μπορεί να ξεχωρίσει ποια δεδομένα είναι σημαντικά.
- Με την μηχανική μάθηση αποφεύγεται ο συνεχής ανασχεδιασμός μιας εργασίας όταν αλλάζουν οι συνθήκες και τα δεδομένα εκτέλεσής της. Οι αλγόριθμοι που αναπτύσσονται μπορούν να προσαρμόζουν τα έμπειρα συστήματα στις αλλαγές που συμβαίνουν.

3.1.2. Ταξινόμηση με Χρήση Μηχανικής Μάθησης

Η ***ταξινόμηση (classification)*** είναι μια διαδικασία κατηγοριοποίησης ενός ***παραδείγματος (example)***, με βάση τις ***τιμές (values)*** του σε συγκεκριμένα ***χαρακτηριστικά (attributes / features)***. Ένα χαρακτηριστικό μπορεί να λαμβάνει αριθμητικές (numerical) ή/και ονοματικές (nominal) τιμές, ενώ οι αριθμητικές μπορεί να είναι διακριτές (discrete) ή συνεχείς (continuous). Η ταξινόμηση γίνεται σε ***κλάσεις (classes)***, οι οποίες μπορεί να είναι δυο ή περισσότερες και οι τιμές τους είναι αριθμητικές (π.χ. -1, 0, +1) ή ονοματικές (π.χ. ΝΑΙ, ΟΧΙ). [33]

Για να είναι σε θέση μια υπολογιστική μηχανή να ταξινομήσει άγνωστα παραδείγματα, θα πρέπει να εκπαιδευτεί με ήδη γνωστά, ώστε να αποκτήσει εμπειρία. Για τον σκοπό αυτό, χρησιμοποιούνται οι αλγόριθμοι της μηχανικής μάθησης. Ένα παράδειγμα με γνωστή κλάση ονομάζεται ***πρότυπο*** ή ***παράδειγμα εκπαίδευσης (training example)***. Το κάθε πρότυπο είναι ένα διάνυσμα n διαστάσεων: $\vec{p} = [p_1, p_2, ..., p_n]$ και ανήκει σε μια κλάση y_i , όπου p_i η τιμή του προτύπου για το i χαρακτηριστικό. Όλα τα πρότυπα συνιστούν την ***πληροφορία εκπαίδευσης (training data)*** η οποία περιγράφεται ως εξής: $D = \left\{ (\vec{p_1}, y_1), ..., (\vec{p_n}, y_n) \right\}$.

Η μηχανική μάθηση διακρίνεται σε δυο κατηγορίες, ανάλογα με την γνώση ή μη των κλάσεων των πρότυπων παραδειγμάτων. Όταν στην

πληροφορία εκπαίδευσης εμπεριέχονται οι κλάσεις των πρότυπων, η μάθηση χαρακτηρίζεται ως ***επιτηρούμενη (supervised learning)***. Αντίθετα, όταν η πληροφορία εκπαίδευσης περιέχει μόνο πρότυπα χωρίς να προσδιορίζεται η κλάση του καθενός, τότε η μηχανή προσπαθεί, αναλύοντας τις τιμές των χαρακτηριστικών κάθε παραδείγματος, να τα διαχωρίσει σε κλάσεις. Η διαδικασία αυτή αναφέρεται ως ***μη επιτηρούμενη μάθηση (unsupervised learning)*** [33]. Στην συνέχεια θα ασχοληθούμε μόνο με αλγορίθμους επιτηρούμενης μηχανικής μάθησης.

Το πρώτο στάδιο σε μια εργασία ταξινόμησης με την τεχνική της μηχανικής μάθησης είναι η εκπαίδευση (training) της μηχανής. Η πληροφορία εκπαίδευσης, που δέχεται η μηχανή, έχει συγκεκριμένη μορφή:

Πρότυπα	**Χαρακτηριστικό 1**	**Χαρακτηριστικό 2**	**. . .**	**Χαρακτηριστικό n**	**Κλάση**
Πρότυπο 1			. . .		
Πρότυπο 2			. . .		
Πρότυπο 3			. . .		
. . .	. . .	. . .		. . .	. . .
Πρότυπο N			. . .		

Εικόνα 3.1.1: Πληροφορία εκπαίδευσης.

Σκοπός στο στάδιο αυτό είναι η σταδιακή ανάπτυξη ενός αλγορίθμου με την ρύθμιση διαφόρων παραμέτρων, ο οποίος θα μπορεί να διαχωρίζει τα πρότυπα ανάλογα με την κλάση τους. Όπως γίνεται αντιληπτό, η πληροφορία εκπαίδευσης πρέπει να περιέχει πρότυπα παραδείγματα όλων των κλάσεων. Όμως, η χρήση μεγάλου όγκου πληροφορίας μπορεί να δυσκολέψει την διαδικασία της μάθησης καθώς καταναλίσκεται μεγάλος χώρος στη μνήμη του υπολογιστή και οι υπολογισμοί γίνονται πολύπλοκοι.

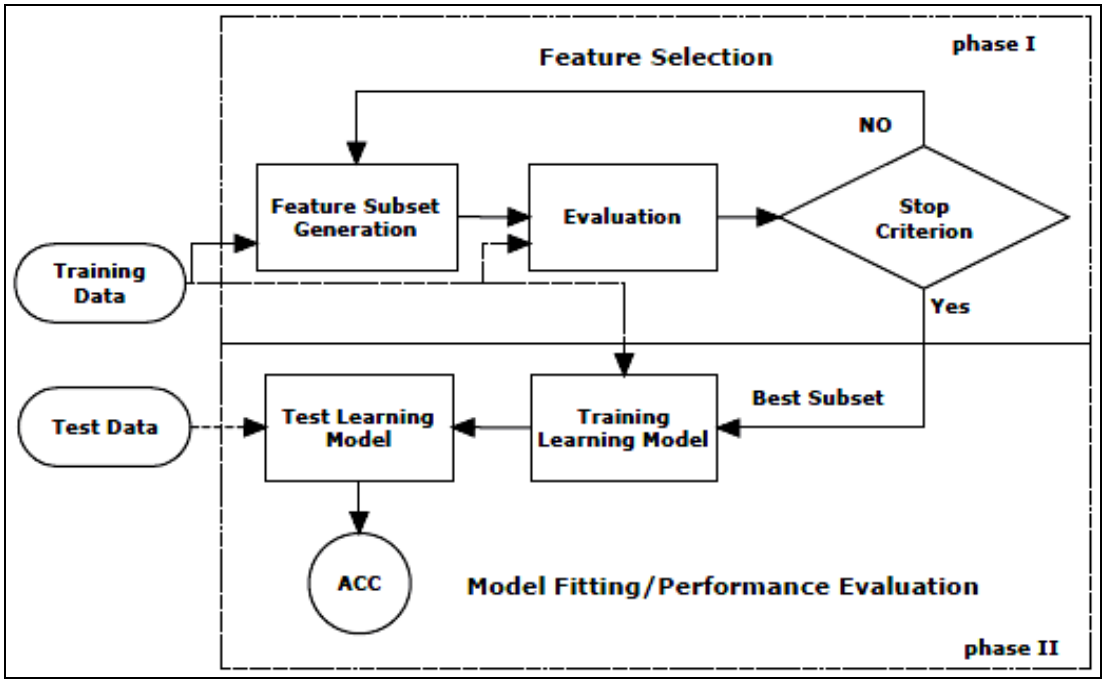
Ο αλγόριθμος, στο τέλος της μάθησης, θα έχει αναπτύξει μια συνάρτηση, η οποία ονομάζεται ***ταξινομητής (classifier).*** Ο ταξινομητής είναι αυτός που θα χρησιμοποιηθεί στο μέλλον για την ταξινόμηση άγνωστης κλάσης παραδειγμάτων. Για να διαπιστωθεί αν η μηχανή έχει εκπαιδευτεί

σωστά, δηλαδή αν ο ταξινομητής μπορεί να κατατάξει ένα άγνωστο δείγμα στη σωστή κλάση, γίνεται ***αξιολόγησή (validation)*** του με τα πρότυπα παραδείγματα.

Μια ευρέως χρησιμοποιούμενη μέθοδος μάθησης-αξιολόγησης είναι η **cross-validation**. Κατά την διαδικασία αυτή, η πληροφορία εκπαίδευσης χωρίζεται σε k ίσα μέρη (συνήθως 10). Στη συνέχεια πραγματοποιούνται k διαδικασίες μάθησης-αξιολόγησης. Σε κάθε διαδικασία τα k-1 τμήματα της πληροφορίας εκπαίδευσης χρησιμοποιούνται για την μηχανική μάθησης, ενώ ένα τμήμα χρησιμοποιείται για την αξιολόγηση. Μια μονή διαδικασία αξιολόγησης-μάθησης μπορεί να οριστεί ως **split validation,** κατά την οποία η πληροφορία εκπαίδευσης χωρίζεται σε δυο μέρη (με βάση συγκεκριμένη αναλογία, π.χ. 30%-70%) και το μεγάλο μέρος χρησιμοποιείται για την μάθηση, ενώ το μικρό για την αξιολόγηση. [40]

Μια τεχνική που χρησιμοποιείται με σκοπό την βελτίωση της απόδοσης και τον καλύτερο διαχωρισμό της πληροφορίας εκπαίδευσης είναι η ***επιλογή χαρακτηριστικών (feature selection)***. Ενδέχεται η πληροφορία εκπαίδευσης να έχει πολλή μεγάλη διάσταση ή να περιέχει πληροφορία που δημιουργεί θόρυβο. Η επιλογή χαρακτηριστικών απομακρύνει θορυβώδη και περιττά χαρακτηριστικά, μειώνοντας τη διάσταση της πληροφορίας και αυξάνοντας ταυτόχρονα την ταχύτητα πραγματοποίησης της μάθησης, την απόδοση και την δυνατότητα εύκολης κατανόησης του ταξινομητή.

Ο τυπικός αλγόριθμος επιλογής χαρακτηριστικών αποτελείται από δυο φάσεις (εικόνα 3.1.2), 1) την επιλογή χαρακτηριστικών και 2) την μάθηση και αξιολόγηση του ταξινομητή. Η πρώτη φάση περιλαμβάνει τρία στάδια: Αρχικά επιλέγεται ένα τμήμα της πληροφορίας εκπαίδευσης που περιέχει μια ομάδα χαρακτηριστικών. Στη συνέχεια αξιολογείται η χρησιμότητα κάθε χαρακτηριστικού της ομάδας. Με βάση την αξιολόγηση κάποια χαρακτηριστικά απομακρύνονται και άλλα επιλέγονται ως χρήσιμα. Τέλος ελέγχεται αν η ομάδα των χρήσιμων χαρακτηριστικών είναι αποτελεσματική με βάση ένα κριτήριο διακοπής. Αν είναι αποτελεσματική τότε ο αλγόριθμος έχει βρει τα χρήσιμα χαρακτηριστικά και η πρώτη φάση έχει ολοκληρωθεί, αλλιώς επαναλαμβάνεται. Κατά την δεύτερη φάση πραγματοποιείται η μηχανική μάθηση και η αξιολόγηση του ταξινομητή χρησιμοποιώντας την ομάδα των επιλεγμένων χαρακτηριστικών. [35]



Εικόνα 3.1.2: Τυπικός αλγόριθμος επιλογής χαρακτηριστικών.

Η απόδοση ενός ταξινομητή περιγράφεται στον ***πίνακα σύγχυσης (confusion matrix)*** και με τον υπολογισμό ορισμένων δεικτών. Όταν το πρόβλημα ταξινόμησης αφορά δυο κλάσεις τότε η μια αναφέρεται ως ***θετική κλάση (positive class)*** και η άλλη ως ***αρνητική (negative class)***. Ο πίνακα σύγχυσης έχει την εξής μορφή:

		Κλάση στην οποία ταξινομήθηκαν	
		negative	positive
Πραγματική κλάση	negative	True Negatives	False Positives
	positive	False Negatives	True Positives

Εικόνα 3.1.3: Πίνακας σύγχυσης δυο κλάσεων (θετική και αρνητική κλάση).

Κάθε στοιχείο, A_{ij}, του πίνακα είναι το πλήθος των πρότυπων που ανήκουν στην κλάση i και ο ταξινομητής τα κατέταξε στην κλάση j. Το στοιχείο A_{21} είναι το πλήθος των θετικών παραδειγμάτων, τα οποία ταξινομήθηκαν στα αρνητικά (false negatives, FN). Στην συγκεκριμένη περίπτωση ο ταξινομητής υπέπεσε σε σφάλμα τύπου 1 (type error 1). Αντίστοιχα, το στοιχείο A_{12} είναι το πλήθος των αρνητικών παραδειγμάτων τα οποία ταξινομήθηκαν στα θετικά

(false positives, FP) και ο ταξινομητής υπέπεσε σε σφάλμα τύπου 2 (type error 2). Γενικά η απόδοση του ταξινομητή μεγιστοποιείται όταν μηδενίζονται τα FS και FN. Με βάση αυτούς τους ορισμούς υπολογίζονται οι εξής δείκτες:

- ***Ακρίβεια (accuracy)***: $\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$ (3.1.1)
- ***Ευστοχία (precision)***: $\text{Precision} = \frac{TP}{TP + FP}$ (3.1.2)
- ***Επιστροφή (recall)***: $\text{Recall} = \frac{TP}{TP + FN}$ (3.1.3)

Η ακρίβεια είναι το ποσοστό των παραδειγμάτων, τα οποία ταξινομήθηκαν σωστά και αποτελεί το βασικό μέτρο της ***απόδοσης (performance)*** του ταξινομητή. Η ευστοχία δείχνει την ικανότητα του ταξινομητή να διακρίνει ποια παραδείγματα είναι θετικά και η επιστροφή δηλώνει την ευαισθησία του ταξινομητή να κατατάσσει σωστά τα θετικά παραδείγματα. Οι παραπάνω δείκτες μπορούν να υπολογιστούν και για ταξινομήσεις παραδειγμάτων σε τρεις ή περισσότερες κλάσεις.

Η επιστροφή μπορεί να συμβολιστεί ως TP_{rate}. Το ποσοστό κατάταξης των λανθασμένων αρνητικών παραδειγμάτων, περιγράφεται από το ποσοστό $FP_{rate} = \frac{FP}{TN + FP}$. Χρησιμοποίωντας τα δυο αυτά ποσοστά κατασκευάζονται ***τα διαγράμματα ROC (Receiver Operating Characteristics)***. Η θεωρία των διαγραμμάτων ROC βασίζεται στην υπόθεση πως ο ταξινομητής ορίζει μια τιμή κατώφλι (π.χ. 0.5) και όσα παραδείγματα έχουν τιμή πάνω από το κατώφλι ταξινομούνται ως θετικά, αντιθέτως αρνητικά. Έτσι αν το κατώφλι διαχωρίζει σωστά τις δυο κλάσεις, μηδενίζεται το ποσοστό FP_{rate} και TP_{rate} =1. Ένα πολύ σημαντικό στατιστικό μέγεθος των διαγραμμάτων ROC είναι η ***περιοχή κάτω από το διάγραμμα (area under ROC, AUC)***. Το εμβαδόν αυτής της περιοχής ισούται με την πιθανότητα ο ταξινομητής να αξιολογήσει ένα τυχαίο θετικό παράδειγμα με σκορ μεγαλύτερο από ένα τυχαίο αρνητικό. Αν AUC=1 τότε ο ταξινομητής δίνει σε όλα τα θετικά παραδείγματα μεγαλύτερα σκορ απ' ότι στα αρνητικά και επομένως οι δυο κλάσεις διαχωρίζονται πλήρως. [41]

3.1.3. Στοιχεία Γραμμικής Άλγεβρας

3.1.3.1. Υπερεπίπεδο

Ένα υποσύνολο $U \neq \varnothing$ ενός διανυσματικού χώρου W, λέγεται υποχώρος του W αν είναι και αυτό, ένας διανυσματικός χώρος και περιέχει το σημείο μηδέν [28]. Ο υποχώρος ονομάζεται και ***υπερεπίπεδο (hyperplane)***. Επομένως αν ο διανυσματικός χώρος W είναι n διαστάσεων, τότε το υπερεπίπεδο U μπορεί να είναι το πολύ n διαστάσεων. Για παράδειγμα, τα υπερεπίπεδα του διανυσματικού χώρου $\Re^2$ είναι δυο διαστάσεων (ο ίδιος ο $\Re^2$), μίας διάστασης (ευθείες που διέρχονται από την αρχή των αξόνων) ή μηδενικής διάστασης (το σημείο μηδέν). Ο σκοπός σε ορισμένους αλγορίθμους της μηχανικής μάθησης είναι η εύρεση του βέλτιστου υπερεπιπέδου διαχωρισμού δυο κλάσεων ($y_i = \pm 1$). Στις περιπτώσεις αυτές, ο όρος υπερεπίπεδο αφορά το διανυσματικό χώρο n-1 διαστάσεων. Τα υπερεπίπεδα n-1 διαστάσεων έχουν την ιδιότητα να διαχωρίζουν τον χώρο n διαστάσεων σε δυο ίσους χώρους.

Ένα πολύ σημαντικό μέτρο στον υπολογισμό του βέλτιστου υπερεπιπέδου, είναι το ***εσωτερικό γινόμενο (dot product)*** δυο διανυσμάτων. Αν δυο διανύσματα $\vec{x} = [x_1, x_2, ..., x_n]$ και $\vec{y} = [y_1, y_2, ..., y_n]^T$, τότε το εσωτερικό τους γινόμενο είναι:

$$\boxed{\left\langle \vec{x}, \vec{y} \right\rangle = x_1 y_1 + x_2 y_2 + ... + x_n y_n} \qquad (3.1.4)$$

Η εξίσωση κάθε υπερεπιπέδου μπορεί να γραφτεί ως εξής [36]:

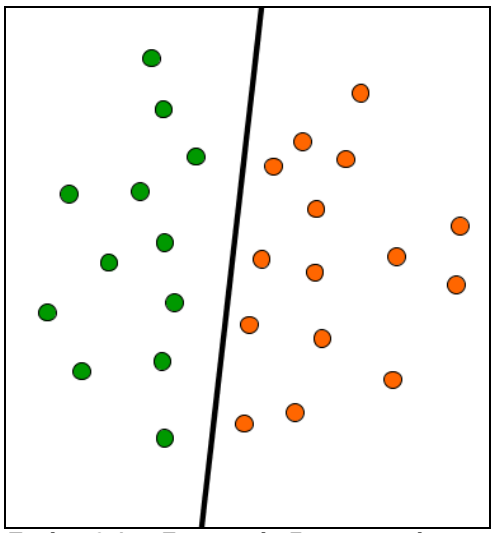
$$\boxed{\left\langle \vec{w}, \vec{x} \right\rangle + b = 0} \qquad (3.1.5)$$

δηλαδή είναι το άθροισμα του εσωτερικού γινομένου ενός ***διανύσματος βαρών (weight vector)*** $\vec{w} = [w_1, w_2, ..., w_n]^T$ με ένα διάνυσμα

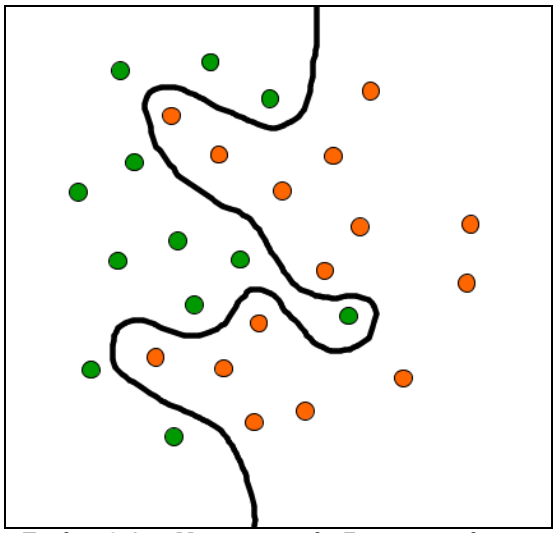
$\vec{x} = [x_1, x_2, ..., x_n]$, και ενός ***κατωφλιού (threshold)*** b. Το διάνυσμα $\vec{w}$ είναι κάθετο στο υπερεπίπεδο και ορίζει τη διεύθυνσή του, ενώ το κατώφλι b ορίζει την θέση του στο χώρο.

3.1.3.2. Γραφική Απεικόνιση της Πληροφορίας Εκπαίδευσης

Στους αλγορίθμους μηχανικής μάθησης, η πληροφορία εκπαίδευσης, $D = \left\{(\vec{p}_1, y_1), ..., (\vec{p}_n, y_n)\right\}$ μπορεί να απεικονιστεί σε γράφημα. Έστω ότι μια πληροφορία αποτελείται από n πρότυπα παραδείγματα δυο διαστάσεων, $\vec{p}_i = [p_1, p_2]$ με $i = 1,2,...,n$, τα οποία ταξινομούνται σε δυο κλάσεις, $y = \pm 1$. Το γράφημα της πληροφορίας θα είναι ένα διάγραμμα δυο διαστάσεων, όπου κάθε πρότυπο απεικονίζεται με ένα σημείο. Όταν οι δυο κλάσεις μπορούν να διαχωριστούν με ένα υπερεπίπεδο, στο συγκεκριμένο παράδειγμα με μια ευθεία γραμμή, τότε η πληροφορία εκπαίδευσης χαρακτηρίζεται ως ***γραμμικώς διαχωρίσιμη (linearly separable)*** (εικόνα 3.1.4). Αντίθετα, όταν ο διαχωρισμός με ένα υπερεπίπεδο δεν είναι εφικτός, η πληροφορία εκπαίδευσης είναι ***γραμμικώς μη διαχωρίσιμη (linearly inseparable)*** (εικόνα 3.1.5).



Εικόνα 3.1.4: Γραμμικός διαχωρισμός.



Εικόνα 3.1.5: Μη γραμμικός διαχωρισμός.

Οι αλγόριθμοι μηχανικής μάθησης έχουν αναπτυχθεί και βελτιωθεί σημαντικά στην ανάλυση γραμμικώς διαχωρίσιμων πληροφοριών εκπαίδευσης [37]. Όμως, σε πραγματικά δεδομένα οι πληροφορίες εκπαίδευσης είναι συνήθως μη γραμμικώς διαχωρίσιμες. Επομένως απαιτούνται μη γραμμικές μέθοδοι διαχωρισμού των κλάσεων, ώστε να γίνει με επιτυχία η μάθηση της μηχανής.

3.1.3.3. Επεξεργασία μη γραμμικώς διαχωρίσιμης πληροφορίας εκπαίδευσης

Μια μέθοδος εύρεσης του βέλτιστου υπερεπιπέδου για τον διαχωρισμό μη γραμμικώς διαχωρίσιμων προτύπων είναι η απεικόνιση τους σε χώρο μεγαλύτερων διαστάσεων. Η μετατροπή των διαστάσεων γίνεται με τη χρήση μιας ***συνάρτησης μη γραμμικού μετασχηματισμού (feature map)*** $\Phi(.)$, και ο νέος χώρος ονομάζεται ***μετασχηματισμένος χώρος (feature space)***. Στον χώρο αυτό η πληροφορία εκπαίδευσης είναι γραμμικώς διαχωρίσιμη. Η συγκεκριμένη μέθοδος αποτελείται από δυο στάδια, τον μετασχηματισμό της πληροφορίας εκπαίδευσης και το τέχνασμα του πυρήνα.

Ένα πρότυπο παράδειγμα δυο διαστάσεων είναι το $\vec{p} = [p_1, p_2]$ και η γραφική του απεικόνιση είναι ένα σημείο στον δισδιάστατο χώρο. Αν εφαρμοστεί ένας μη γραμμικός μετασχηματισμός $\Phi(.): \Re^2 \to \Re^3$ τότε το πρότυπο $\vec{p}$ μετασχηματίζεται στο $\Phi\left(\vec{p}\right) = \left[p_1^2, \sqrt{2}p_1p_2, p_2^2\right]$ και παριστάνεται με ένα σημείο στο χώρο τριών διαστάσεων [38]. Τα στοιχεία του μετασχηματισμένου διανύσματος είναι μονώνυμα $2^{ου}$ βαθμού (degree). Με τον τρόπο αυτό μετασχηματίζεται η πληροφορία εκπαίδευσης και οι κλάσεις μπορούν να διαχωριστούν γραμμικά στο χώρο τριών διαστάσεων.

Υπάρχουν πολλές συναρτήσεις μετασχηματισμού $\Phi(.)$ που μπορούν να εφαρμοστούν ώστε να καταστεί γραμμικός ο διαχωρισμός των δυο κλάσεων. Για παράδειγμα, το μετασχηματισμένο διάνυσμα θα μπορούσε να είναι: $\Phi\left(\vec{p}\right) = [p_1, p_1p_2, p_2]$ ή $\Phi\left(\vec{p}\right) = \left[p_1, \sqrt{p_1p_2}, p_1p_2, p_2\right]$. Όμως ορισμένες

συναρτήσεις αυξάνουν κατά πολύ τις διαστάσεις των πρότυπων (η διάσταση του μετασχηματισμένου προτύπου μπορεί να γίνει και άπειρη [28]), ενώ περιέχουν και μεγάλο αριθμό παραμέτρων. Τέτοιες συναρτήσεις πρέπει να αποφεύγονται γιατί δημιουργούν πολύπλοκη μετασχηματισμένη πληροφορία εκπαίδευσης και η επεξεργασία της είναι υπολογιστικά δαπανηρή. [38]

Το εσωτερικό γινόμενο του μετασχηματισμένου προτύπου $\Phi\left(\vec{p}\right)=\left[p_1^{\,2},\sqrt{2}p_1p_2,p_2^{\,2}\right]$ και ενός άγνωστου μετασχηματισμένου παραδείγματος, το οποίο πρόκειται να ταξινομηθεί, $\Phi\left(\vec{x}\right)=\left[x_1^{\,2},\sqrt{2}x_1x_2,x_2^{\,2}\right]$ είναι:

$$\boxed{\left\langle \Phi\left(\vec{p}\right),\Phi\left(\vec{x}\right)\right\rangle = p_1^{\,2}x_1^{\,2}+2p_1x_1p_2x_2+p_2^{\,2}x_2^{\,2}=\left(p_1x_1+p_2x_2\right)^2=\left\langle \vec{p},\vec{x}\right\rangle^2} \qquad (3.1.6)$$

Από την εξίσωση 3.1.6 βλέπουμε πως το εσωτερικό γινόμενο των δυο μετασχηματισμένων παραδειγμάτων, με στοιχεία $2^{ου}$ βαθμού, ισούται με το εσωτερικό γινόμενο των δυο παραδειγμάτων στον αρχικό τους χώρο, υψωμένο στο τετράγωνο. Η ισότητα αυτή οφείλεται στην συνάρτηση μετασχηματισμού $\Phi(.)$ που χρησιμοποιήθηκε. Γενικεύοντας, για δυο παραδείγματα $\vec{x}$ και $\vec{y}$ n διαστάσεων, δεχόμενοι πως τα στοιχεία των μετασχηματισμένων παραδειγμάτων είναι μονώνυμα d βαθμού, αποδεικνύεται [38] πως για τον μη γραμμικό μετασχηματισμό $\Phi(.):\Re^n\to\Re^{n^d}$ ισχύει:

$$\boxed{\left\langle \Phi\left(\vec{x}\right),\Phi\left(\vec{y}\right)\right\rangle = \left\langle \vec{x},\vec{y}\right\rangle^d} \qquad (3.1.7)$$

δηλαδή το εσωτερικό γινόμενο στον μετασχηματισμένο χώρο είναι ο ίσο με το εσωτερικό γινόμενο των παραδειγμάτων στον πραγματικό χώρο, υψωμένο στο d. Η συνάρτηση $k\left(\vec{x},\vec{y}\right)=\left\langle \vec{x},\vec{y}\right\rangle^d$ είναι μια μορφή πολυωνυμικής

συνάρτησης d βαθμού και υπάγεται σε μια ειδική κατηγορία συναρτήσεων, τις ***συναρτήσεις πυρήνα (kernel functions)***. Όπως ειπώθηκε παραπάνω, ο υπολογισμός του εσωτερικού γινομένου των μετασχηματισμένων παραδειγμάτων μπορεί να είναι δαπανηρή και χρονοβόρα διαδικασία. Όμως υπάρχουν συναρτήσεις μετασχηματισμού $\Phi(.)$, για τις οποίες το εσωτερικό γινόμενο των μετασχηματισμένων διανυσμάτων ισούται και μπορεί να αντικατασταθεί με μια συνάρτηση πυρήνα, επομένως ο μετασχηματισμός δεν χρειάζεται να πραγματοποιηθεί. Η μέθοδος αυτή ονομάζεται ***το τέχνασμα του πυρήνα (the kernel trick)***.

Έχουν βρεθεί πολλές συναρτήσεις πυρήνα οι οποίες χρησιμοποιούνται στους αλγορίθμους μηχανικής μάθησης με σκοπό την ανάλυση μη γραμμικώς διαχωρίσιμης πληροφορίας εκπαίδευσης. Στον παρακάτω πίνακα αναφέρονται μερικές από αυτές:

Πίνακας 3.1.1: Συναρτήσεις πυρήνα (kernel functions)

Όνομα συνάρτησης	Τύπος συνάρτησης
dot	$k\left(\vec{x},\vec{y}\right)=\left\langle \vec{x},\vec{y}\right\rangle$
radial	$k\left(\vec{x},\vec{y}\right)=\exp\left(-g\left\|\vec{x}-\vec{y}\right\|^2\right)$
polynomial	$k\left(\vec{x},\vec{y}\right)=\left(a\left\langle \vec{x},\vec{y}\right\rangle+c\right)^d$
gaussian	$k\left(\vec{x},\vec{y}\right)=\exp\left(\frac{\left\|\vec{x}-\vec{y}\right\|^2}{2\sigma^2}\right)$
anova	$k\left(\vec{x},\vec{y}\right)=\sum_{i=1}^{n}\exp\left(-g\left(x_i-y_i\right)\right)^d$
neural (sigmoid)	$k\left(\vec{x},\vec{y}\right)=\tanh\left(a\left\langle \vec{x},\vec{y}\right\rangle+b\right)$
multiquadric	$k\left(\vec{x},\vec{y}\right)=\sqrt{\left\|\vec{x}-\vec{y}\right\|^2+c^2}$

[40][39]

3.1.4. Γενικός Αλγόριθμος Ταξινόμησης

Ένας γενικός αλγόριθμος ταξινόμησης είναι ο εξής: το άγνωστο παράδειγμα $\vec{x}$ θα ταξινομηθεί σε μια από τις κλάσεις $y = \pm 1$ με βάση την γειτνίασή του με το μέσο πρότυπο παράδειγμα $\vec{p}_{mean}$ κάθε κλάσης. Ο ταξινομητής, αν η πληροφορία εκπαίδευσης είναι γραμμικώς διαχωρίσιμη, είναι η παρακάτω εξίσωση:

$$\boxed{y = \mathrm{sgn}\left(\left\langle \vec{x}, \vec{p}_{mean:y=+1} \right\rangle - \left\langle \vec{x}, \vec{p}_{mean:y=-1} \right\rangle + b \right)} \qquad (3.1.8)$$

όπου η συνάρτηση sgn είναι η συνάρτηση σημείο (signum), $\vec{p}_{mean:y_i} = \frac{1}{n_{y_i}} \times \sum_{i:y_i}^{n_{yi}} \vec{p}_i$ είναι το μέσο παράδειγμα της κλάσης y_i στον πραγματικό χώρο, και $b = \frac{1}{2}\left(\left\| \vec{p}_{mean:y=-1} \right\|^2 - \left\| \vec{p}_{mean:y=+1} \right\|^2 \right)$. Αν η πληροφορία εκπαίδευσης δεν είναι γραμμικώς διαχωρίσιμη τότε ο αλγόριθμος χρησιμοποιεί τα μετασχηματισμένα διανύσματα $\Phi\left(\vec{x}\right)$ και $\Phi\left(\vec{p}_{mean}\right)$, και αντικαθιστά τα εσωτερικά τους γινόμενα με συνάρτηση πυρήνα: [37]:

$$\boxed{y = \mathrm{sgn}\left(\frac{1}{n_{y=+1}} \times \sum_{i:y=+1}^{n_{y=+1}} k\left(p_i, x\right) - \frac{1}{n_{y=-1}} \times \sum_{i:y=-1}^{n_{y=-1}} k\left(p_i, x\right) + b \right)} \qquad (3.1.9)$$

Ο παραπάνω αλγόριθμος αποτελεί μια γενική εικόνα των περισσότερων αλγορίθμων ταξινόμησης με χρήση μηχανικής μάθησης. Στην συνέχεια θα αναφερθούν πέντε συγκεκριμένοι ταξινομητές, οι οποίοι χρησιμοποιήθηκαν στο τεχικό μέρος της εργασίας: ο Μπεϋζιανός ταξινομητής Πυρήνα (Naive Bayes Kernel classifier), τα Δέντρα Απόφασης (Decision Trees), ο κ-Κοντινότερος-Γείτονας (k-Nearest-Neighbor), οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) και ο ταξινομητής Perceptron.

3.2. Μπεϋζιανός Ταξινομητής

3.2.1. Εισαγωγή

Ο ***μπεϋζιανός ταξινομητής (Bayesian classifier)*** είναι στατιστικός ταξινομητής. Η βάση του συγκεκριμένου ταξινομητή είναι το ***θεώρημα του Bayes (Bayes' theorem).*** Έστω τα γεγονότα X και Y. Η πιθανότητα $P(X)$ ονομάζεται ***εκ των προτέρων πιθανότητα (prior probability)*** και είναι η πιθανότητα να συμβεί το γεγονός X, ομοίως για το γεγονός Y. Η πιθανότητα $P(X|Y)$ ονομάζεται ***δεσμευμένη πιθανότητα (conditional probability)*** ή ***εκ των υστέρων πιθανότητα (posterior probability)*** του X ως προς το Y και δηλώνει την πιθανότητα να συμβεί το γεγονός X δεδομένου ότι έχει συμβεί το Y. Το θεώρημα του Bayes ορίζει ότι:

$$P(X|Y) = \frac{P(Y|X) \times P(X)}{P(Y)} \qquad (3.2.1)$$

με $P(Y)>0$. Η πιθανότητα $P(X|Y)$ είναι ίση με την $P(X)$ όταν τα δυο γεγονότα, X και Y, είναι ανεξάρτητα μεταξύ τους, δηλαδή αν η πραγματοποίηση του ενός δεν επηρεάζει την πραγματοποίηση του άλλου.

3.2.2. Μάθηση-Ταξινόμηση

Ο μπεϋζιανός ταξινομητής, για ένα άγνωστο παράδειγμα $\vec{x} = [x_1, x_2, ..., x_n]$, υπολογίζει την δεσμευμένη πιθανότητα της κάθε κλάσης y_i, ως προς το παράδειγμα $\vec{x}$ [43]:

$$P\left(y_i\middle|\vec{x}\right)=\frac{P\left(\vec{x}\middle|y_i\right)\times P(y_i)}{P\left(\vec{x}\right)} \qquad (3.2.2)$$

δηλαδή δεδομένου ότι υπάρχει το παράδειγμα $\vec{x}$, η $P\left(y_i\middle|\vec{x}\right)$ είναι η πιθανότητα η y_i να είναι η κλάση του $\vec{x}$. Ο μπεϋζιανός ταξινομητής θα ταξινομήσει το παράδειγμα $\vec{x}$ στην κλάση y_i, αν και μόνο αν για κάθε $j\neq i$ ισχύει: $P\left(y_i\middle|\vec{x}\right)>P\left(y_j\middle|\vec{x}\right)$. Επομένως το ζητούμενο στην μπεϋζιανή ταξινόμηση είναι να βρεθεί η μέγιστη δεσμευμένη πιθανότητα $P\left(y_i\middle|\vec{x}\right)$. Ο παρονομαστής της εξίσωσης 3.2.2 είναι ίδιος για όλες τις κλάσεις, επομένως δεν επηρεάζει την τελική απόφαση του ταξινομητή. Το πρόβλημα περιορίζεται στην μεγιστοποίηση της σχέσης $P\left(y_i\middle|\vec{x}\right)\times P(y_i)$.

Η πιθανότητα $P(y_i)$ μπορεί να υπολογιστεί από την πληροφορία εκπαίδευσης. Για παράδειγμα, αν στην πληροφορία υπάρχουν συνολικά N πρότυπα παραδείγματα, από τα οποία τα N_1 έχουν κλάση y_1 και τα N_2 έχουν κλάση y_2, τότε $P(y_1)=N_1/N$ και $P(y_2)=N_2/N$. Ο υπολογισμός όμως της $P\left(\vec{x}\middle|y_i\right)$ είναι πιο δύσκολος. Επειδή υπολογιστικά είναι αρκετά δαπανηρός, γίνεται η παραδοχή ότι τα χαρακτηριστικά των παραδειγμάτων είναι ανεξάρτητα ανά δυο [44][43], επομένως:

$$P\left(\vec{x}\middle|y_i\right)=\prod_{k=1}^{n}P\left(x_k\middle|y_i\right)=P\left(x_1\middle|y_i\right)\times P\left(x_2\middle|y_i\right)\times\ldots\times P\left(x_n\middle|y_i\right) \qquad (3.2.3)$$

όπου x_k η τιμή του $\vec{x}$ για το χαρακτηριστικό A_k. Η παραπάνω παραδοχή διευκολύνει τους υπολογισμούς, όμως σε πραγματικά δεδομένα ισχύει ελάχιστες φορές, καθώς τα χαρακτηριστικά δεν είναι ανεξάρτητα μεταξύ τους.

Η εξίσωση 3.2.3 είναι ο ***απλός μπεϋζιανός ταξινομητής (Naive Bayes classifier)***.

Αν οι πιθανές κλάσεις ταξινόμησης είναι δυο, $y_i = \pm 1$, τότε η εξίσωση ταξινόμησης μπορεί να γραφτεί ως εξής:

$$y = \mathrm{sgn}\left[P(y_{+1}) \times \prod_{k=1}^{n} P(x_k | y_{+1}) - P(y_{-1}) \times \prod_{k=1}^{n} P(x_k | y_{-1}) \right] \qquad (3.2.4)$$

Για τον υπολογισμό του απλού μπεϋζιανού ταξινομητή κάθε κλάσης, θα πρέπει να είναι γνωστή η $P(x_k | y_i)$ για κάθε χαρακτηριστικό A_k. Αν το χαρακτηριστικό A_k είναι διακριτή μεταβλητή, δηλαδή οι τιμές που μπορεί να πάρει είναι πεπερασμένες, τότε υπάρχει συνάρτηση $F_{A_k}(x_k)$, η οποία δίνει την πιθανότητα το χαρακτηριστικό A_k να πάρει την τιμή x_k, $P(A_k = x_k)$, και ονομάζεται ***συνάρτηση πιθανότητας, σ.π. (probability mass function, pmf)*** [42]. Δεδομένου ότι η κλάση είναι y_i, η ζητούμενη δεσμευμένη πιθανότητα είναι $P(A_k = x_k | y_i)$. Η πιθανότητα αυτή υπολογίζεται εύκολα από την πληροφορία εκπαίδευσης.

Όταν το χαρακτηριστικό A_k είναι συνεχής μεταβλητή, δηλαδή οι τιμές που μπορεί να πάρει είναι άπειρες, τότε η πιθανότητα $P(A_k = x_k | y_i)$ είναι μηδέν. Υπάρχει όμως συνάρτηση $F(x_k)$, η οποία υπολογίζει την πιθανότητα η τιμή του A_k, να βρίσκεται σε ένα συγκεκριμένο διάστημα τιμών $[a,b)$, $P(a \le A_k < b)$. Η συνάρτηση αυτή ονομάζεται ***συνάρτηση πυκνότητας πιθανότητας, σ.π.π. (probability density function, pdf)*** [42]. Για τον υπολογισμό της $P(a \le A_k < b)$ πρέπει να είναι γνωστή η κατανομή της σ.π.π. και ορισμένες στατιστικές συναρτήσεις όπως για παράδειγμα η μέση τιμή (mean value) και η διασπορά (variance) των τιμών του A_k.

Τις περισσότερες φορές, η κατανομή της σ.π.π. ή οι στατιστικές συναρτήσεις δεν είναι γνωστές και πρέπει να εκτιμηθούν από την πληροφορία εκπαίδευσης. Η εκτίμηση της σ.π.π. μπορεί να γίνει με διάφορες μεθόδους (παραμετρικές και μη παραμετρικές). Στις παραμετρικές μεθόδους,

γίνεται η παραδοχή ή είναι γνωστό, πως το σχήμα της σ.π.π. έχει πολλές ομοιότητες με το σχήμα γνωστής κατανομής. Επομένως υπολογίζονται οι σχετικές στατιστικές συναρτήσεις με διάφορες μεθόδους όπως η μέθοδος μέγιστης πιθανοφάνειας (maximum likelihood method) και η μέθοδος των ροπών (generalized method of moments). Στις μη παραμετρικές μεθόδους, το σχήμα της σ.π.π. δεν είναι γνωστό και υπολογίζεται από την διαθέσιμη πληροφορία εκπαίδευσης. Η πιο γνωστή μη παραμετρική μέθοδος εκτίμησης της σ.π.π. είναι το ιστόγραμμα.

3.2.3. Το Ιστόγραμμα και οι Συναρτήσεις Πυρήνα

Το ***ιστόγραμμα (histogram)*** είναι ένα διάγραμμα δυο διαστάσεων και απεικονίζει την κατανομή των συχνοτήτων ενός χαρακτηριστικού A_k. Το εύρος των τιμών του A_k είναι $r = A_{k,max} - A_{k,min}$. Η σημαντική παράμετρος του ιστογράμματος είναι το ***πλάτος διαστήματος (bin width, b)*** και κάθε διάστημα, i $(i = 1,2,...,n)$, του εύρους μπορεί να γραφτεί ως εξής: $\left[A_{k,min} + (i-1)\times b, A_{k,min} + i \times b\right)$.

Από τις τιμές των πρότυπων παραδειγμάτων για το A_k κατασκευάζεται το ιστόγραμμα και ο εκτιμητής της σ.π.π. του χαρακτηριστικού A_k στο i διάστημα υπολογίζεται ως εξής [46]:

$$\boxed{\hat{F}_{hist}\left(A_k\right) = \frac{N_i}{N \times b}} \qquad (3.2.5)$$

όπου N ο αριθμός των πρότυπων παραδειγμάτων και N_i το πλήθος των τιμών του A_k στο διάστημα i.

Ένας εναλλακτικός τρόπος ορισμού των διαστημάτων ενός ιστογράμματος είναι ο εξής: αν $\bar{x}$ η μέση τιμή κάθε διαστήματος (τάξης) και h η διασπορά, συνήθως ίδια για όλα τα διαστήματα, τότε το κάθε διάστημα μπορεί να γραφτεί ως: $\left[\bar{x} - h, \bar{x} + h\right)$ [46]. Από τον ορισμό της σ.π.π., η

πιθανότητα του χαρακτηριστικού A_k να πάρει τιμή $x_k \in \left[\bar{x}_i - h, \bar{x}_i + h \right)$ είναι : $P\left(\bar{x}_i - h \le x_k < \bar{x}_i + h \right) = \int_{\bar{x}_i - h}^{\bar{x}_i + h} f(t)dt \approx 2h \times f(x)$. Επομένως ο εκτιμητής της σ.π.π. δίνεται από την σχέση:

$$\boxed{\hat{F}(x_k) = \frac{1}{2h} \times P\left(\bar{x}_i - h \le x_k < \bar{x}_i + h \right)} \qquad (3.2.6)$$

Όμως η πιθανότητα $P\left(\bar{x}_i - h \le x_k < \bar{x}_i + h \right)$ είναι ίση με την συχνότητα του διαστήματος i, δηλαδή [47]:

$$\boxed{P\left(\bar{x}_i - h \le x_k < \bar{x}_i + h \right) = \frac{N_{[\bar{x}_i - h, \bar{x}_i + h]}}{N} = \frac{1}{N} \times \sum_{k=1}^{N} 1\left(\left| x_k - \bar{x}_i \right| < h \right)} \qquad (3.2.7)$$

Τελικά, από τις εξισώσεις 3.2.6 και 3.2.7 προκύπτει ο εκτιμητής:

$$\boxed{\hat{F}(x_k) = \frac{1}{Nh} \times \sum_{k=1}^{N} \frac{1}{2} \times 1\left(\left| x_k - \bar{x}_i \right| < h \right)} \qquad (3.2.8)$$

Μια γενική μορφή της εξίσωσης 3.2.8, είναι η εξής [45]:

$$\boxed{\hat{F}(x_k) = \frac{1}{Nh} \times \sum_{k=1}^{N} k\left(\frac{x_k - \bar{x}_i}{h} \right)} \qquad (3.2.9)$$

όπου η συνάρτηση $k\left(\frac{x_k - \bar{x}_i}{h} \right)$ είναι ***συνάρτηση πυρήνα (kernel function)*** και δηλώνει το σχήμα του εκτιμητή της σ.π.π. ενώ η παράμετρος h ονομάζεται ***εύρος (bandwidth)*** ή ***σταθερά εξομάλυνσης (smoothing constant)*** και δείχνει τον βαθμό εξομάλυνσης στην εκτίμηση της σ.π.π. Είναι γνωστό πως η επιλογή της συνάρτησης πυρήνα δεν παίζει σημαντικό ρόλο

στην εκτίμηση της σ.π.π. Η σημαντική παράμετρος είναι το εύρος, το οποίο όσο πιο μικρό είναι τόσο πιο λεπτομερής είναι η καμπύλη του εκτιμητή, πολύ μικρές τιμές όμως δημιουργούν προβλήματα. [45]

Ο μπεϋζιανός ταξινομητής, ο οποίος χρησιμοποιείται για συνεχής μεταβλητές (χαρακτηριστικά) και υπολογίζει τον εκτιμητή της σ.π.π. κάθε χαρακτηριστικού με την χρήση συναρτήσεων πυρήνων ονομάζεται ***απλός μπεϋζιανός ταξινομητής πυρήνα (Naive Bayes Kernel classifier).*** Οι πιο γνωστές συναρτήσεις πυρήνα παρουσιάζονται στον παρακάτω πίνακα:

ΠΙΝΑΚΑΣ 3.2.1: Συναρτήσεις πυρήνα

Όνομα συνάρτησης	Τύπος
Rectangular	$K(t)=\begin{cases} 1/2, & \text{για} \quad \lvert t\rvert<1 \\ 0 & \text{αλλου} \end{cases}$
Epanechnikov	$K(t)=\begin{cases} \frac{3}{4}\left(1-\frac{1}{5}t^2\right)}{\sqrt{5}}, & \text{για} \quad \lvert t\rvert<\sqrt{5} \\ 0 & \text{αλλου} \end{cases}$
Biweight	$K(t)=\begin{cases} \frac{15}{16}\left(1-t^2\right)^2, & \text{για} \quad \lvert t\rvert<1 \\ 0 & \text{αλλου} \end{cases}$
Triangular	$K(t)=\begin{cases} 1-\lvert t\rvert, & \text{για} \quad \lvert t\rvert<1 \\ 0 & \text{αλλου} \end{cases}$
Gaussian	$K(t)=\frac{1}{\sqrt{2\pi}}e^{-(1/2)t^2}$

3.3. Δέντρα Απόφασης

3.3.1. Εισαγωγή

Τα ***δέντρα απόφασης (decision trees)*** είναι μια κατηγορία αλγορίθμων μηχανικής μάθησης, οι οποίοι έχουν τη μορφή δέντρου. Στις αρχές τις δεκαετίας του ’80, ο ερευνητής J.Ross Quinlan ανέπτυξε τον πρώτο αλγόριθμο δέντρου απόφασης, γνωστό ως ID3, ενώ λίγο αργότερα παρουσίασε τον διάδοχό του με το όνομα C4.5. Το 1984 οι στατιστικολόγοι L.Breiman, J.Friedman, R. Olshen και C.Stone εξέδωσαν το βιβλίο «Δέντρα Ταξινόμησης και Παλινδρόμησης» (Classification and Regression Trees, CART) όπου περιέγραφαν τη γενιά των δυαδικών δέντρων απόφασης. Παρόλο που οι αλγόριθμοι ID3 και CART αναπτύχθηκαν ξεχωριστά, από διαφορετικούς επιστήμονες, παρουσιάζουν κοινές προσεγγίσεις του ζητήματος της μηχανικής μάθησης μέσω των δέντρων απόφασης. Όπως οι πρώτοι αλγόριθμοι της δεκαετίας του ’80, έτσι και οι περισσότεροι σημερινοί κατασκευάζουν τα δέντρα απόφασης με κατεύθυνση από πάνω προς τα κάτω, δηλαδή ξεκινώντας από ολόκληρη την πληροφορία εκπαίδευσης και διαχωρίζοντάς τη σε μικρότερα τμήματα καθώς αναπτύσσεται το δέντρο.

3.3.2. Μάθηση-Ταξινόμηση

3.3.2.1 Δομή Δέντρου – Αλγόριθμος του Hunt

Όπως έχει αναφερθεί, το δέντρο απόφασης έχει την μορφή δέντρου, το οποίο αποτελείται από κόμβους, κλαδιά και φύλλα. Η αρχή ενός δέντρου απόφασης είναι ένας κόμβος, ο οποίος ονομάζεται ***ρίζα (root)***. Στην ρίζα δεν εισέρχεται καμία πληροφορία, μόνο εξέρχεται με την μορφή ***διακλαδώσεων (branches)***. Η ρίζα αποτελεί την πρώτη εξέταση ενός άγνωστου παραδείγματος, το οποίο επρόκειτο να ταξινομηθεί με το δέντρο απόφασης. Η εξέταση αφορά ένα συγκεκριμένο χαρακτηριστικό του παραδείγματος και

ανάλογα με τις ιδιότητες του χαρακτηριστικού, οι διακλαδώσεις της ρίζας μπορεί να είναι δυο ή περισσότερες. Όλοι οι υπόλοιποι ***κόμβοι (nodes)*** του δέντρου έχουν ακριβώς ένα εισερχόμενο άκρο. Οι κόμβοι, στους οποίους πραγματοποιείται μια εξέταση, ονομάζονται ***εσωτερικοί κόμβοι (internal nodes)*** ή ***κόμβοι εξέτασης (test nodes)*** και φέρουν διακλαδώσεις, οι οποίες οδηγούν σε νέους κόμβους. Οι κόμβοι, οι οποίοι δεν έχουν διακλαδώσεις, ονομάζονται ***φύλλα (leafs)*** ή ***τελικοί κόμβοι (terminal nodes)*** και καθορίζουν την κλάση του εξεταζόμενο παραδείγματος. Συμπερασματικά, το άγνωστο παράδειγμα οδηγείται από την ρίζα στους εσωτερικούς κόμβους και ανάλογα με τα αποτελέσματα στις εξετάσεις, καταλήγει σε ένα φύλλο, το οποίο καθορίζει την κλάση του. Για την ανάπτυξη του παραπάνω αλγορίθμου τίθενται ερωτήματα όπως ποιά χαρακτηριστικά των παραδειγμάτων θα εξετάζονται, πόσα και σε ποιόν κόμβο το καθένα [50].

Για την μηχανική μάθηση και την κατασκευή του δέντρου απόφασης χρησιμοποιείται κυρίως ***ο αλγόριθμος του Hunt (Hunt's algorithm)*** [49]. Με βάση τον συγκεκριμένο αλγόριθμο, το δέντρο αναπτύσσεται με τέτοιο τρόπο ώστε η πληροφορία εκπαίδευσης, $D = \left\{(\vec{p_1}, y_1),...,(\vec{p_n}, y_n)\right\}$, να διαχωρίζεται σε μικρότερα τμήματα (subsets), έως ότου το κάθε τμήμα, P, να αποτελείται από πρότυπα ίδιας κλάσης, $P_{y_m} = \left\{(\vec{p_{1m}}, y_m),...,(\vec{p_{nm}}, y_m)\right\}$. Παρακάτω παρουσιάζονται τα βασικά βήματα του αλγόριθμου του Hunt:

- Βήμα 1: Εξέταση του τμήματος πληροφορίας εκπαίδευσης P στον κόμβο N (αν $P = D$ τότε ο κόμβος είναι η ρίζα).
- Βήμα 2: Αν όλα τα πρότυπα του P ανήκουν στην ίδια κλάση, $y = m$, τότε ο κόμβος N μετατρέπεται σε φύλλο με κλάση m. Αν όμως περιέχει πρότυπα δυο ή περισσότερων κλάσεων τότε γίνεται περαιτέρω έλεγχος.
- Βήμα 3: Αν δεν υπάρχουν διαθέσιμα χαρακτηριστικά με τα οποία μπορεί να γίνει διαχωρισμός του P, τότε ο N μετατρέπεται σε φύλλο και η κλάση του ορίζεται αυτή με τα περισσότερα πρότυπα στο P. Αν υπάρχουν διαθέσιμα χαρακτηριστικά, τότε το P διαχωρίζεται.
- Βήμα 4: Ο διαχωρισμός του P πραγματοποιείται με βάση ένα ***κριτήριο διαχωρισμού (splitting criterion)***, το οποίο καθορίζει ποιό είναι το καλύτερο χαρακτηριστικό που μπορεί να διαχωρίσει το P σε μικρότερα τμήματα, ώστε

τα υποσύνολα που θα προκύψουν να είναι πιο ομοιογενή από το P. Επιπλέον το κριτήριο διαχωρισμού ορίζει και τον αριθμό των διακλαδώσεων του N, δηλαδή σε πόσα τμήματα θα διαχωριστεί το P. Με τον τρόπο αυτό ο N γίνεται κόμβος εξέτασης.

- Βήμα 5: Για το κάθε τμήμα P_i που προκύπτει, ελέγχεται αν περιέχει πρότυπα. Αν το τμήμα P_i είναι κενό, τότε η διακλάδωση i καταλήγει σε φύλλο, η κλάση του οποίου είναι αυτή με τα περισσότερα πρότυπα στο P. Αν το P_i δεν είναι κενό, τότε ο αλγόριθμος επιστρέφει στο βήμα 1.

Ο αλγόριθμος ολοκληρώνεται όταν το κάθε τμήμα της πληροφορίας εκπαίδευσης που έχει δημιουργηθεί αποτελείται από πρότυπα ίδιας κλάσης ή δεν υπάρχουν άλλα χαρακτηριστικά για τη δημιουργία νέων κόμβων εξέτασης.

3.3.2.2. Κριτήρια Διαχωρισμού

Ο διαχωρισμός ενός τμήματος P της πληροφορίας εκπαίδευσης εξαρτάται από το χαρακτηριστικό, A_k, που επιλέγεται. Το A_k μπορεί να είναι διακριτή ή συνεχής μεταβλητή. Στη διακριτή μεταβλητή (π.χ. $A_k = [$κόκκινο, πράσινο, κίτρινο, μπλε$]$) η κάθε διακλάδωση του κόμβου N αντιστοιχεί σε μια τιμή του A_k, ενώ αν έχει οριστεί πως κάθε κόμβος εξέτασης μπορεί να έχει μόνο δυο διακλαδώσεις, η κάθε διακλάδωση του N αντιστοιχεί σε ένα υποσύνολο των τιμών του A_k (π.χ. $\Delta 1 = [$κόκκινο, κίτρινο$]$ και $\Delta 2 = [$πράσινο, μπλε$]$. Στη συνεχή μεταβλητή (π.χ. $A_k \in [0,100]$) οι διακλαδώσεις ορίζονται είτε από διαστήματα αριθμητικών τιμών (π.χ. $\Delta 1 \in [0,30)$, $\Delta 2 \in [30,70)$ και $\Delta 3 \in [70,100]$) είτε από ανισότητες (π.χ. $\Delta 1 : A_k < 30$ και $\Delta 2 : A_k \geq 30$). Ανεξάρτητα από το είδος του χαρακτηριστικού, το βασικό πρόβλημα για τον διαχωρισμό των προτύπων είναι η επιλογή του κατάλληλου κριτηρίου.

Τα κριτήρια διαχωρισμού του P που έχουν αναπτυχθεί, υπολογίζουν το καταλληλότερο χαρακτηριστικό έτσι ώστε τα υποσύνολα P_i που θα προκύψουν να έχουν μεγάλο βαθμό ομοιογένειας-καθαρότητας (purity degree). Ορισμένα από τα χρησιμοποιούμενα κριτήρια διαχωρισμού είναι: το

κέρδος πληροφορίας, ο λόγος κέρδους, ο δείκτης gini και η ακρίβεια του δέντρου. [44]

Το ***κέρδος πληροφορίας (information gain)*** χρησιμοποιήθηκε για πρώτη φορά στον αλγόριθμο ID3. Το κριτήριο αυτό βασίζεται στη θεωρία πληροφορίας και ορίζεται ως η ποσότητα της πληροφορίας που αποκτάται για την ταξινόμηση ενός παραδείγματος στο σύνολο P, αν το υποσύνολο διαχωριστεί με βάση το χαρακτηριστικό A_k. Η απαιτούμενη ποσότητα ταξινόμησης ενός πρότυπου του συνόλου P ονομάζεται εντροπία του P και δίνεται από την εξίσωση:

$$\boxed{\inf o(P) = -\sum_{i=1}^{m} p_i \log_2 (p_i)} \qquad (3.3.1)$$

όπου $p_i = \left|P_{y_i}\right| / \left|P\right|$, η πιθανότητα ένα πρότυπο του P να αντιστοιχεί στην κλάση y_i. Έστω ότι πραγματοποιείται ο διαχωρισμός με το χαρακτηριστικό A_k. Ένα υποσύνολο P_i μπορεί να περιέχει πρότυπα δυο ή περισσότερων κλάσεων, δηλαδή να είναι ανομοιογενές. Ορίζεται ένα επιπλέον μέτρο, $\inf o_{A_k}(P)$, το οποίο δηλώνει την ποσότητα πληροφορίας που απαιτείται να αποκτηθεί για την ταξινόμηση ενός παραδείγματος του P, μετά τον διαχωρισμό με βάση το A_k. Όπως είναι λογικό, όσο μικρότερη είναι αυτή η πληροφορία, τόσο πιο ομοιογενές είναι το P_i που προέκυψε.

$$\boxed{\inf o_{A_k}(P) = \sum_{i=1}^{r} \frac{\left|P_i\right|}{\left|P\right|} \times \inf o(P_i)} \qquad (3.3.2)$$

Το κέρδος πληροφορίας ορίζεται ως εξής:

$$\boxed{gain(A_k) = \inf o(P) - \inf o_{A_k}(P)} \qquad (3.3.3)$$

Δηλαδή το κέρδος πληροφορίας είναι η διαφορά της ποσότητας πληροφορίας που απαιτείται για την ταξινόμηση ενός προτύπου του P με την ποσότητα της

πληροφορίας που θα χρειάζεται μετά τον διαχωρισμό με βάση το A_k. Όσο μεγαλύτερη η τιμή του $\text{gain}(A_k)$ τόσο πιο ομοιογενή υποσύνολα δημιουργούνται και τόσο λιγότερη περαιτέρω επεξεργασία χρειάζεται για την ολοκλήρωση του δέντρου. Επομένως επιλέγεται το χαρακτηριστικό με την μεγαλύτερη $\text{gain}(A_k)$.

Έστω ένα χαρακτηριστικό που λαμβάνει πολλές διαφορετικές τιμές, n στο σύνολό τους. Ο διαχωρισμός με βάση το συγκεκριμένο χαρακτηριστικό σε n διακλαδώσεις ίσως να δημιουργεί ομοιογενή υποσύνολα. Με το κέρδος πληροφορίας επιλέγονται συνήθως τέτοια χαρακτηριστικά και η πληροφορία P διαχωρίζεται σε πολλά υποσύνολα. Όμως ο τρόπος αυτός διαχωρισμού είναι μη χρηστικός στα δέντρα απόφασης και πρέπει να αποφεύγεται.

Για τον λόγο αυτό, ο διάδοχος του ID3, ο αλγόριθμος C4.5 χρησιμοποιεί ένα διαφορετικό κριτήριο διαχωρισμού, τον ***λόγο κέρδους (gain ratio)***. Η πληροφορία διαχωρισμού του συνόλου P δίνεται από την εξίσωση:

$$\text{split_info}_{A_k}(P) = -\sum_{i=1}^{r} \frac{|P_i|}{|P|} \times \log_2\left(\frac{|P_i|}{|P|}\right) \qquad (3.3.4)$$

Το μέτρο αυτό δηλώνει την πληροφορία που γίνεται διαθέσιμη διαχωρίζοντας το P σε r υποσύνολα. Όσο μεγαλύτερο είναι το r τόσο μεγαλύτερη πληροφορία διαχωρισμού προκύπτει. Το μέτρο αυτό χρησιμοποιείται για την κανονικοποίηση του κέρδους πληροφορίας σύμφωνα με την εξίσωση:

$$\text{gain_ratio}(A_k) = \frac{\text{gain}(A_k)}{\text{split_info}_{A_k}(P)} \qquad (3.3.5)$$

Η εξίσωση 3.3.5 ορίζει το λόγο κέρδους και το καταλληλότερο χαρακτηριστικό για τον διαχωρισμό του P είναι αυτό με το μεγαλύτερο $\text{gain_ratio}(A_k)$.

Ο ***δείκτης gini (gini index)***, ο οποίος χρησιμοποιείται στους αλγόριθμους CART, περιγράφει τον βαθμό ανομοιογένειας του συνόλου P σύμφωνα με την εξίσωση:

$$\boxed{\text{gini}(P) = 1 - \sum_{i=1}^{m} p_i^2} \qquad (3.3.6)$$

Όπου p_i η πιθανότητα ένα πρότυπο του P να αντιστοιχεί στην κλάση y_i. Όπως έχει προαναφερθεί, στους αλγόριθμους CART ένας κόμβος διακλαδίζεται μόνο σε δυο νέους. Ο πιο κατάλληλος συνδυασμός των τιμών του A_k, για τον διαχωρισμό του P είναι αυτός που ελαχιστοποιεί την εξίσωση:

$$\boxed{\text{gini}_{A_k}(P) = \frac{|P_1|}{|P|}\text{gini}(P_1) + \frac{|P_2|}{|P|}\text{gini}(P_2)} \qquad (3.3.7)$$

Η εξίσωση 3.3.7 είναι ένα μέτρο της συνολικής ανομοιογένειας των δυο υποσυνόλων που προκύπτουν από τον διαχωρισμό του P με βάση το A_k. Επομένως, όμως μικρότερος είναι αυτός ο δείκτης, τόσο πιο ομοιογενή υποσύνολα δημιουργούνται. Από τις εξισώσεις 3.3.6 και 3.3.7 προκύπτει πως η μείωση της ανομοιογένειας του συνόλου P είναι η διαφορά του βαθμού ανομοιογένειας του P με την συνολική ανομοιογένεια των δυο υποσυνόλων P_1, P_2:

$$\boxed{\Delta\text{gini}(A_k) = \text{gini}(P) + \text{gini}_{A_k}(P)} \qquad (3.3.8)$$

Το χαρακτηριστικό A_k για το οποίο μεγιστοποιείται η εξίσωση 3.3.8 επιλέγεται για τον διαχωρισμό της πληροφορίας σε αυτό τον κόμβο.

Ένα επιπλέον κριτήριο διαχωρισμού είναι η συνολική ***ακρίβεια (accuracy)*** του δέντρου απόφασης. Με βάση το κριτήριο αυτό, επιλέγονται τα χαρακτηριστικά τα οποία μεγιστοποιούν την ακρίβεια του δέντρου κατά την αξιολόγησή του (validation).

3.3.2.3. Κλάδεμα Δέντρου

Το ***κλάδεμα δέντρου (tree pruning)*** είναι μια τεχνική απλοποίησης του δέντρου απόφασης με σκοπό να απομακρυνθούν κλάδοι που δεν χρησιμεύουν, να μειωθεί η πολυπλοκότητα του και να καταστεί πιο κατανοητό. Ο αλγόριθμος κατασκευής ενός δέντρου μπορεί να δημιουργήσει ένα υπερπροσαρμοσμένο (overfitted) δέντρο για συγκεκριμένη πληροφορία εκπαίδευσης. Η τεχνική του κλαδέματος σκοπεύει στην μείωση της υπερπροσαρμογής του δέντρου. Υπάρχουν δυο τεχνικές κλαδέματος ενός δέντρου, το ***προ-κλάδεμα (prepruning)*** και το ***μετα-κλάδεμα (postpruning)***. Κατά το προ-κλάδεμα παύει η ανάπτυξη του δέντρου μετά από έναν αριθμό κόμβων. Με βάση ορισμένα στατιστικά μεγέθη υπολογίζεται η χρησιμότητα της επιπλέον διακλάδωσης. Αν τα νέα υποσύνολα δεν ικανοποιούν μια τιμή κατωφλιού ο αλγόριθμος σταματά και οι τελευταίοι κόμβοι μετατρέπονται σε φύλλα. Όταν το κλάδεμα πραγματοποιείται μετά την ανάπτυξη του δέντρου, τότε η τεχνική ονομάζεται μετα-κλάδεμα. Με βάση ένα κριτήριο (π.χ. τον αριθμό των φύλλων) ο αλγόριθμος μετα-κλαδέματος μειώνει την πολυπλοκότητα του δέντρου μετατρέποντας κλάδους σε απλά φύλλα. Το κλάδεμα του δέντρου χρησιμοποιείται συχνά καθώς καθιστά τα δέντρα πιο κατανοητά και βελτιώνει την επίδοσή τους κατά την αξιολόγηση. [44]

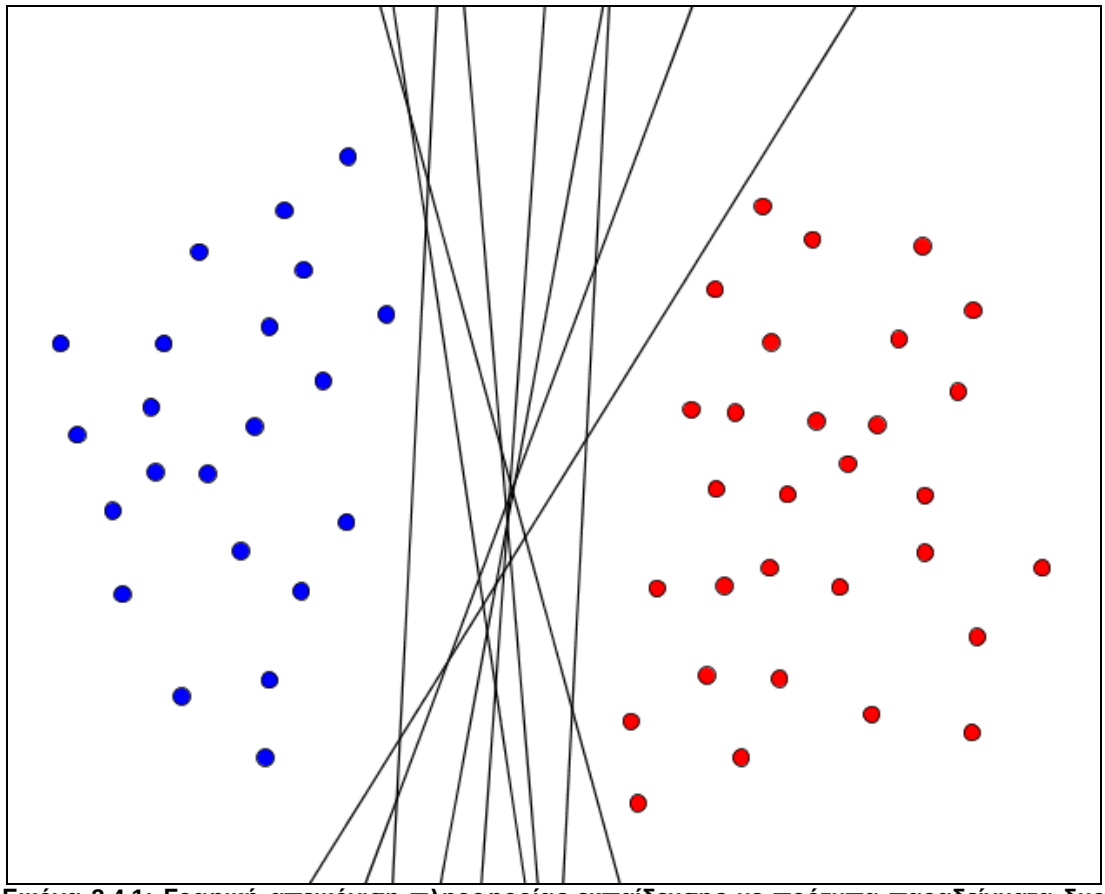
3.4. Μηχανές Διανυσμάτων Υποστήριξης

3.4.1. Εισαγωγή

Οι ***μηχανές διανυσμάτων υποστήριξης (support vector machines)*** αναφέρθηκαν για πρώτη φορά το 1992 από τους Vladimir Vapnik, Bernhard Boser και Isabelle Guyon, ενώ οι μελέτες γύρω από τη συγκεκριμένη μέθοδο είχαν αρχίσει από την δεκαετία του '60. Οι μηχανές διανυσμάτων υποστήριξης χρησιμοποιούνται για ταξινόμηση και παλινδρόμηση (classification and regression) δεδομένων. Η βασική τους αρχή συνοψίζεται στην παρακάτω πρόταση: σε μια πληροφορία εκπαίδευσης δυο κλάσεων, η μηχανή διανυσμάτων υποστήριξης υπολογίζει ένα υπερεπίπεδο, το οποίο διαχωρίζει τα πρότυπα των δυο κλάσεων, έτσι ώστε το περιθώριο μεταξύ τους και η απόστασή τους από το υπερεπίπεδο να μεγιστοποιούνται. Οι μηχανές διανυσμάτων υποστήριξης αναπτύχθηκαν για γραμμικώς διαχωρίσιμη πληροφορία εκπαίδευσης, όμως με την χρήση διαφόρων μεθόδων μπορούν να χρησιμοποιηθούν και για μη γραμμικώς διαχωρίσιμες.

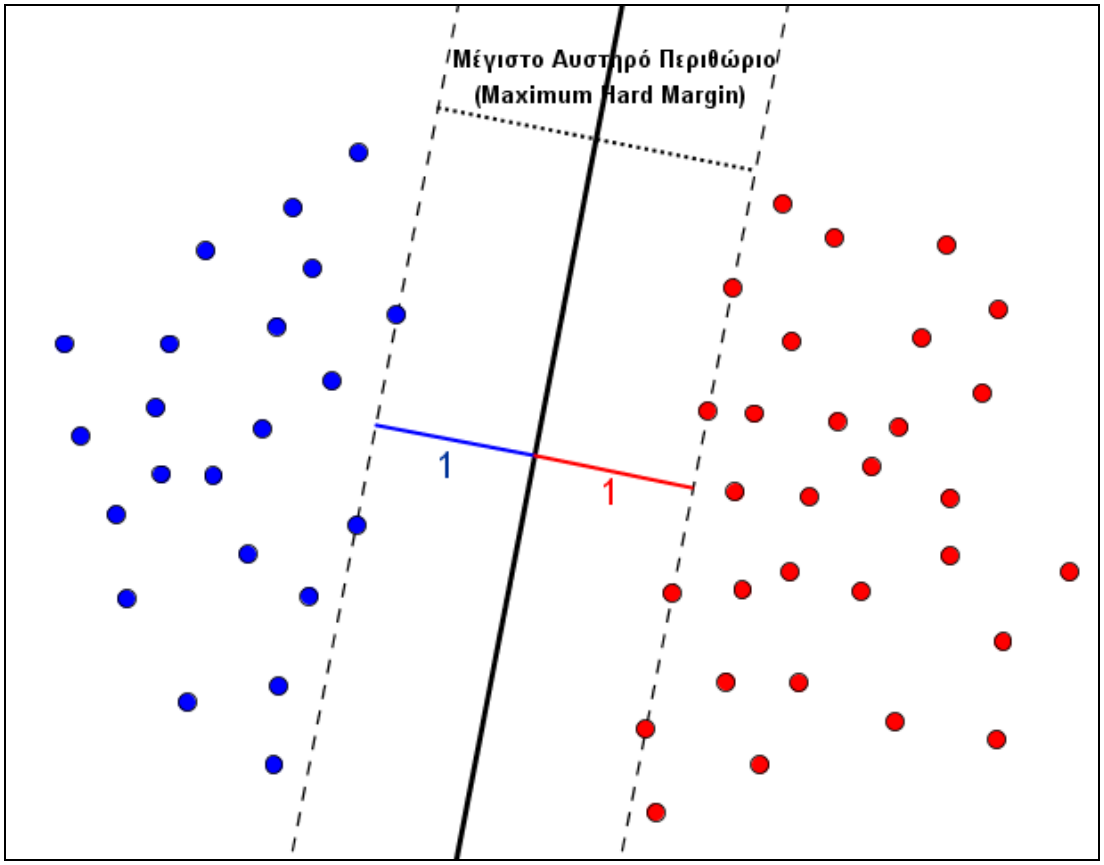
3.4.2. Μάθηση-Ταξινόμηση

Ο αλγόριθμος των μηχανών διανυσμάτων υποστήριξης δημιουργεί μια γραφική απεικόνιση της πληροφορία εκπαίδευσης, $D = \left\{ (\vec{p}_1, y_1), ..., (\vec{p}_n, y_n) \right\}$, όπου κάθε πρότυπο παράδειγμα, $\vec{p} = [p_1, p_2, ..., p_n]$, παριστάνεται με ένα σημείο. Έστω ότι τα πρότυπα παραδείγματα είναι δυο διαστάσεων ότι οι δυο κλάσεις της πληροφορίας εκπαίδευσης είναι $y = \pm 1$. Η γραφική απεικόνιση θα είναι διάγραμμα δυο διαστάσεων, επομένως το υπερεπίπεδο θα είναι μίας διάστασης, δηλαδή μια ευθεία γραμμή. Ο αριθμός των ευθειών, οι οποίες μπορούν να διαχωρίσουν τις δυο κλάσεις, είναι άπειρος (εικόνα 3.4.1) .



Εικόνα 3.4.1: Γραφική απεικόνιση πληροφορίας εκπαίδευσης με πρότυπα παραδείγματα δυο διαστάσεων. Με μπλε χρώμα τα παραδείγματα της κλάσης -1 και με κόκκινο της κλάσης +1. Οι ευθείες είναι πιθανά υπερεπίπεδα που διαχωρίζουν τις δυο κλάσεις.

Όλα τα πιθανά υπερεπίπεδα μπορούν να ταξινομήσουν τα πρότυπα σωστά. Όμως, κατά την ταξινόμηση άγνωστων παραδειγμάτων, τα οποία ίσως να βρίσκονται κοντά σε πρότυπα και των δυο κλάσεων, μπορεί να προκύψουν σφάλματα. Η μηχανή διανυσμάτων υποστήριξης υπολογίζει το υπερεπίπεδο εκείνο, για το οποίο μεγιστοποιείται ο χώρος στον οποίο μπορεί να υπάρξει παράδειγμα για κάθε κλάση y_i. Για τον σκοπό αυτό υπολογίζει το ***μέγιστο αυστηρό περιθώριο (maximum hard margin)*** μεταξύ των δυο κλάσεων. Το συγκεκριμένο περιθώριο είναι το μέγιστο διάστημα που διαχωρίζει τις κλάσεις και το ***βέλτιστο υπερεπίπεδο (optimal hyperplane)*** είναι η παράλληλη ευθεία που ισαπέχει από τα άκρα του περιθωρίου (εικόνα 3.4.2). [51]



Εικόνα 3.4.2: Βέλτιστο υπερεπίπεδο.

Όπως είναι γνωστό από την ενότητα 3.1.3.1 η εξίσωση κάθε υπερεπιπέδου μπορεί να γραφτεί ως εξής:

$$\left\langle \vec{w}, \vec{p} \right\rangle + b = 0 \qquad (3.4.1)$$

Για ένα πρότυπο διάνυσμα $\vec{p} = [p_1, p_2]$ το οποίο βρίσκεται πάνω στο υπερεπίπεδο, η εξίσωση 3.4.1 γράφεται: $w_1 p_1 + w_2 p_2 + b = 0$. Επομένως, με δεδομένο b<0, εάν ένα σημείο βρίσκεται αριστερά από το υπερεπίπεδο ικανοποιεί την σχέση $w_1 p_1 + w_2 p_2 + b < 0$, ενώ εάν βρίσκεται δεξιά ικανοποιεί την σχέση $w_1 p_1 + w_2 p_2 + b > 0$. Προσαρμόζοντας το διάνυσμα βαρών για οποιοδήποτε πρότυπο διάνυσμα $\vec{p}$ ισχύει μια από τις δυο σχέσεις [44]:

$$\boxed{w_1 p_1 + w_2 p_2 + b \geq +1} \quad \text{για} \quad y_i = +1 \tag{3.4.2}$$

$$\boxed{w_1 p_1 + w_2 p_2 + b \leq -1} \quad \text{για} \quad y_i = -1 \tag{3.4.3}$$

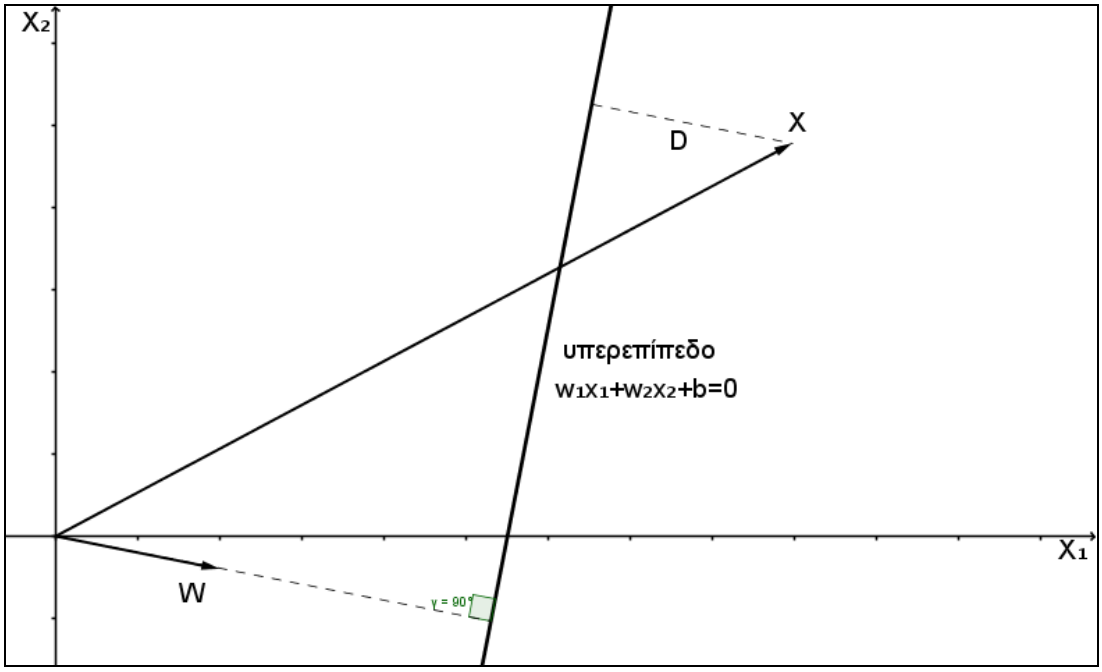
Οι ισότητες στις δυο σχέσεις ορίζουν τα άκρα του περιθωρίου. Τα πρότυπα που ικανοποιούν τις ισότητες βρίσκονται πάνω στα άκρα του περιθωρίου και ανήκουν οριακά στην κλάση +1 ή στην κλάση -1 και ονομάζονται ***διανύσματα υποστήριξης (support vectors)***.

Η εξίσωση του υπερεπιπέδου εξαρτάται από το διάνυσμα βάρους $\vec{w}$ και το κατώφλι b. Όπως προκύπτει από τις εξισώσεις 3.4.2 και 3.4.3, οι παράμετροι w και b περιορίζονται από την εξίσωση:

$$\boxed{\min_i \left| \left\langle \vec{w}, \vec{p}_i \right\rangle + b \right| = 1} \tag{3.4.4}$$

Γενικά, η απόσταση ενός διανύσματος $\vec{x}$ από μια ευθεία $\left(\vec{w}, b\right)$ (εικόνα 3.4.3), δίνεται από την εξίσωση [50]:

$$\boxed{D\left(\vec{w}, b; \vec{x}\right) = \frac{\left| \left\langle \vec{w}, \vec{x}_i \right\rangle + b \right|}{\left\| \vec{w} \right\|} = \frac{\left| w_1 x_1 + w_2 x_2 + b \right|}{\sqrt{w_1^{\,2} + w_2^{\,2}}}} \tag{3.4.5}$$



Εικόνα 3.4.3: Απόσταση D διανύσματος $\vec{X}$ από την ευθεία $w_1x_1 + w_2x_2 + b = 0$.

Επομένως, το περιθώριο $\rho\left(\vec{w}, b\right)$ μεταξύ των κλάσεων +1 και -1 υπολογίζεται ως εξής:

$$\rho\left(\vec{w}, b\right) = \min_{x_i : y_i = -1} D\left(\vec{w}, b; \vec{p_i}\right) + \min_{p_i : y_i = +1} D\left(\vec{w}, b; \vec{p_i}\right) = \min_{p_i : y_i = -1} \frac{\left|\left\langle \vec{w}, \vec{p_i} \right\rangle + b\right|}{\left\|\vec{w}\right\|} + \min_{p_i : y_i = +1} \frac{\left|\left\langle \vec{w}, \vec{p_i} \right\rangle + b\right|}{\left\|\vec{w}\right\|} \Rightarrow$$

$$\rho\left(\vec{w}, b\right) = \frac{2}{\left\|\vec{w}\right\|}$$

Από τον παραπάνω υπολογισμό προκύπτει πως το περιθώριο ταξινόμησης είναι αντιστρόφως ανάλογο της ευκλείδειας νόρμας του διανύσματος βαρών, άρα το πρόβλημα μεγιστοποίησης του περιθωρίου είναι ισοδύναμο με το πρόβλημα ελαχιστοποίησης την νόρμας $\left\|\vec{w}\right\|$. Το ζητούμενο διάνυσμα βαρών υπολογίζεται με ελαχιστοποίηση της συνάρτησης:

$$J\left(\vec{w}, b\right) = \frac{1}{2}\left\|\vec{w}\right\|^2, \text{ δεδομένου ότι ισχύει } \min_i \left|\left\langle \vec{w}, \vec{p_i} \right\rangle + b\right| = 1 \qquad (3.4.6)$$

Οι περαιτέρω υπολογισμοί πραγματοποιούνται με την μέθοδο του τετραγωνικού προγραμματισμού, η θεωρία του οποίου κρίνεται πως ξεφεύγει από τα πλαίσια της συγκεκριμένης εργασίας και γι' αυτό δεν θα αναφερθεί. Τελικά, η εξίσωση του ταξινομητή είναι:

$$y\left(\vec{x}\right) = \sum_{i=1}^{m} y_i a_i \left\langle \vec{p}_i, \vec{x} \right\rangle + b \qquad (3.4.7)$$

όπου a_i είναι υπολογιζόμενες σταθερές, $\vec{p}_i$ το i πρότυπο διάνυσμα υποστήριξης, y_i η κλάση του $\vec{p}_i$ και $\vec{x}$ το άγνωστο διάνυσμα που επρόκειτο να ταξινομηθεί. Αν το αποτέλεσμα της εξίσωσης 3.4.7 είναι θετικό τότε το διάνυσμα $\vec{x}$ ταξινομείται στην κλάση +1, ενώ αν είναι αρνητικό στην κλάση -1.

Στην περίπτωση μη γραμμικώς διαχωρίσιμης πληροφορίας εκπαίδευσης (ενότητα 3.1.3.3), η εξίσωση του ταξινομητή γίνεται:

$$y\left(\vec{x}\right) = \sum_{i=1}^{m} y_i a_i k\left(\vec{p}_i, \vec{x}\right) + b \qquad (3.4.8)$$

3.4.3. Ελαστικοποίηση του Αλγόριθμου

Ο αλγόριθμος των μηχανών διανυσμάτων υποστήριξης σπάνια χρησιμοποιεί μόνο το βέλτιστο αυστηρό περιθώριο για την ταξινόμηση πολύπλοκων πληροφοριών εκπαίδευσης. Η ύπαρξη ενός αυστηρού περιθωρίου διαχωρισμού των δυο κλάσεων είναι πολύ πιθανό να οδηγήσει σε λάθη κατά την ταξινόμηση των πρότυπων παραδειγμάτων ή υπερπροσαρμογή (overfitting) [52]. Για το λόγο αυτό, χρησιμοποιούνται οι μηχανές διανυσμάτων υποστήριξης ***ελαστικού περιθωρίου (soft margin)***, οι οποίες περιέχουν έναν επιπλέον όρο (παράμετρος κανονικοποίησης) ώστε να ελέγχεται η πολυπλοκότητα του ταξινομητή κατά την διαδικασία της εκπαίδευσης.

Για να γίνει πιο ελαστικός ο αλγόριθμος, εισάγεται μια μεταβλητή η οποία μετρά τον βαθμό παραβίασης του βέλτιστου αυστηρού υπερεπιπέδου για κάθε πρότυπο παράδειγμα. Η μεταβλητή αυτή ονομάζεται ***μεταβλητή χαλάρωσης περιθωρίου (margin slack variable)*** και συμβολίζεται με ξ_i, $i=1,2,...,n$. Αν το πρότυπο ταξινομείται σωστά σύμφωνα με την εξίσωση 3.4.7 ή 3.4.8 τότε $\xi_i=0$, ενώ αν η ταξινόμηση είναι σωστή αλλά βρίσκεται πιο κοντά στο υπερεπίπεδο απ' ότι το άκρο του περιθωρίου τότε $\xi_i \in (0,1)$. Σε περίπτωση λανθασμένης ταξινόμησης το $\xi_i \geq 1$.

Με χρήση της μεταβλητής χαλάρωσης περιθωρίου, το πρόβλημα εύρεσης του κατάλληλου διανύσματος βαρών τροποποιείται. Η λύση προκύπτει από την ελαχιστοποίηση της συνάρτησης:

$$J\left(\xi,\vec{w},b\right)=\frac{1}{2}\left\|\vec{w}\right\|^2+C\sum_{i=1}^{l}\xi_i \text{, δεδομένου ότι ισχύει: } \min_i\left|\left\langle\vec{w},\vec{x_i}\right\rangle+b\right|= \quad (3.4.9)$$
$$=1-\xi_i \text{ με } i=1,2,...,l$$

Η σταθερά C είναι μια ***παράμετρος κανονικοποίησης (regularization parameter)*** και είναι $C\geq 0$. Η σταθερά C είναι ένα μέτρο ελέγχου της αυστηρότητας του αλγόριθμου απέναντι σε πρότυπα που βρίσκονται πολύ κοντά στο βέλτιστο περιθώριο ή σε λάθη ταξινόμησης. Όσο αυξάνεται η τιμή της C τόσο ο αλγόριθμος γίνεται πιο αυστηρός και για $C\to\infty$ ο αλγόριθμος είναι οι μηχανές διανυσμάτων υποστήριξης αυστηρού περιθωρίου. [52]

3.5. Αλγόριθμος Perceptron

3.5.1. Εισαγωγή

Ο αλγόριθμος ***Perceptron*** αναπαριστά την λειτουργία ενός και μόνο νευρώνα. Αποτελεί μια από τις παλαιότερες προσεγγίσεις διαχωρισμού δυο κλάσεων στη μηχανική μάθηση, επινοήθηκε από τον Frank Rosenblatt στα τέλη της δεκαετίας του ’60 . Ο αλγόριθμος κατά την διαδικασία της μάθησης, όπως και στις μηχανές διανυσμάτων υποστήριξης, προσπαθεί να υπολογίσει το βέλτιστο υπερεπίπεδο διαχωρισμού δυο κλάσεων. Η διαφορά των δυο αλγορίθμων βρίσκεται στον τρόπο υπολογισμού του υπερεπιπέδου. Ο αλγόριθμος Perceptron αναπτύχθηκε για γραμμικώς διαχωρίσιμη πληροφορία εκπαίδευσης δυο κλάσεων. Για τον διαχωρισμό περισσότερων κλάσεων χρησιμοποιούνται τα νευρωνικά δίκτυα (neural networks).

3.5.2. Ο Βιολογικός Νευρώνας

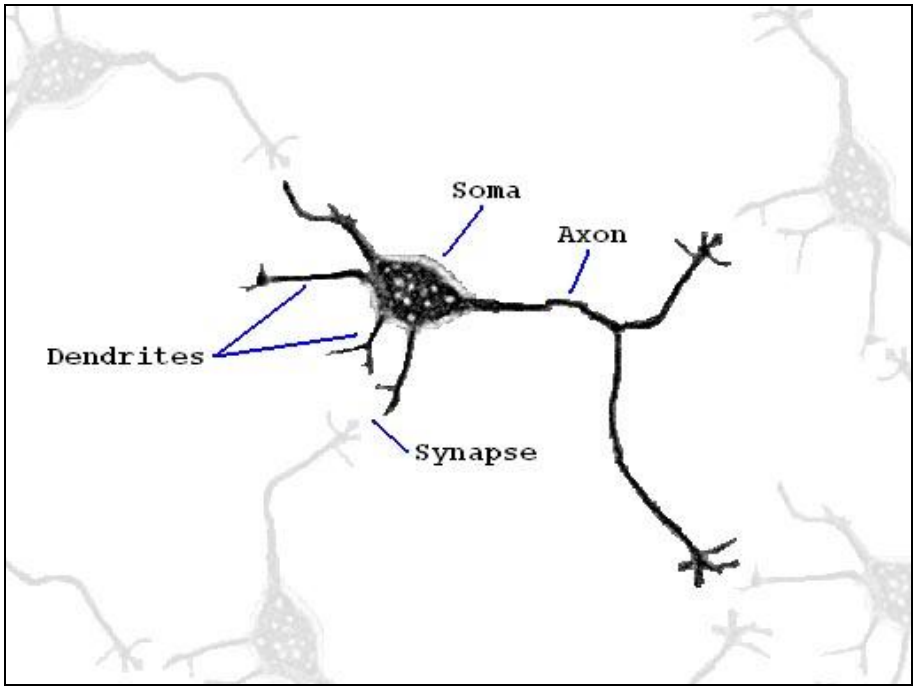
Το νευρικό σύστημα αποτελείται κυρίως από εξειδικευμένα κύτταρα, τα νευρικά κύτταρα ή ***νευρώνες (neurons)***. Ο νευρώνας είναι ένα μεγάλο σε μέγεθος κύτταρο, το οποίο αποτελείται από το κυτταρικό σώμα (περικάρυο), τις αποφύσεις του, οι οποίες είναι ο άξονας και οι δενδρίτες και τις συνάψεις. Η λειτουργία των νευρώνων είναι να αποδέχονται ερεθίσματα και να τα μεταφέρουν σε άλλα όργανα του σώματος.

Παρακάτω παρουσιάζεται μια γενική εικόνα της λειτουργίας του κάθε τμήματος του νευρώνα:

- Το κυτταρικό σώμα: είναι το μεγαλύτερο τμήματα του κυττάρου και περιέχει το γενετικό υλικό.
- Οι δενδρίτες: η πύλη εισόδου του νευρώνα, δέχονται ηλεκτρικούς παλμούς από άλλους νευρώνες.

• Ο άξονας: η πύλη εξόδου του νευρώνα, μοιάζει με μακρόστενη κλωστή και στέλνει σήματα προς τους άλλους νευρώνες υπό μορφή ηλεκτρικών παλμών.

• Οι συνάψεις: είναι τα σημεία ένωσης μεταξύ των διακλαδώσεων του άξονα ενός νευρώνα και των δενδριτών από άλλους νευρώνες. Είναι κύστες με ηλεκτροχημικό υλικό, κυρίως ιόντα νατρίου, καλίου και χλωρίου (Na^+,K^+,Cl^-). Το υλικό αυτό μεταδίδει την ηλεκτρική δραστηριότητα του άξονα στους δενδρίτες. Η ευκολία με την οποία γίνεται αυτή η μετάδοση επηρεάζεται από το πλάτος της σύναψης, την απόστασή της από τον δενδρίτη και την πυκνότητα του υλικού. Το ποσοστό της ηλεκτρικής δραστηριότητας που μεταδίδεται τελικά στο δενδρίτη λέγεται συναπτικό βάρος. [28]



Εικόνα 3.5.1: Δομή νευρώνα. **[54]**

Οι φορείς της πληροφορίας στο νευρικό σύστημα είναι οι ηλεκτρικοί παλμοί. Κάθε ηλεκτρικός παλμός ταξιδεύει από τον άξονα ενός νευρώνα στους δενδρίτες των παραληπτών νευρώνων μέσω των συνάψεων. Κάθε νευρώνας συλλέγει όλο το ηλεκτρικό φορτίο που δέχεται από κάθε σύναψη στους δενδρίτες του ζυγίζοντας το εισερχόμενο φορτίο με το αντίστοιχο συναπτικό βάρος. Έτσι, όσο πιο ισχυρή είναι η συναπτική ζεύξη τόσο πιο έντονα συμμετέχει το συγκεκριμένο φορτίο εισόδου στο συνολικό άθροισμα.

Αν το άθροισμα του φορτίου ξεπερνά μια τιμή κατώφλι, μέσα σε μια χρονική περίοδο, τότε ο νευρώνας πυροβολεί (fires) μέσω του άξονά του, δηλαδή παράγει ηλεκτρικούς παλμούς με μεγάλη συχνότητα (υπάρχουν νευρώνες που πυροβολούν με συχνότητα 100Hz) [54]. Αντίθετα, όταν το φορτίο δεν ξεπερνά το συγκεκριμένο κατώφλι, τότε οι παλμοί που παράγονται είναι αραιοί και ο νευρώνας μένει αδρανής.

3.5.3. Μάθηση-Ταξινόμηση

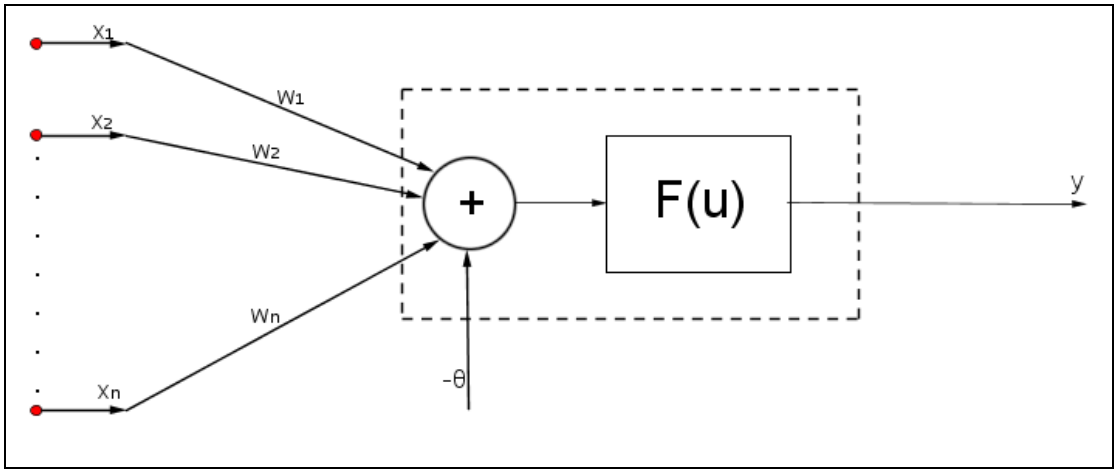
Όπως ο βιολογικός νευρώνας, ο Perceptron έχει πολλές εισόδους, ένα μηχανισμό μετατροπής της πληροφορίας εισόδου και μια πύλη εξόδου. Έστω η πληροφορία εκπαίδευσης (training data) $D = \left\{ (\vec{p}_1, y_1), ..., (\vec{p}_n, y_n) \right\}$. Το κάθε πρότυπο παράδειγμα της πληροφορίας είναι n διαστάσεων, $\vec{p} = [x_1, x_2, ..., x_n]$ και y_i είναι η κλάση του. Το κάθε στοιχείο του προτύπου αποτελεί την πληροφορία που εισέρχεται από μια είσοδο (αντίστοιχη του δενδρίτη) στο κεντρικό τμήμα του αλγορίθμου (αντίστοιχο του κυτταρικού σώματος). Επομένως, για το πρότυπο $\vec{p}$, υπάρχει ένα διάνυσμα $\vec{w} = [w_1, w_2, ..., w_n]^T$, τα στοιχεία του οποίου είναι τα συναπτικά βάρη των χαρακτηριστικών $x_1, x_2, ..., x_n$. Ο αλγόριθμος Perceptron χαρακτηρίζεται από μια τιμή θ, η οποία ονομάζεται κατώφλι ενεργοποίησης (αντίστοιχο με το κατώφλι πάνω από το οποίο ενεργοποιείται ο νευρώνας). Η διέγερση, u, του νευρώνα περιγράφεται από την συνάρτηση [28]:

$$\boxed{u = \sum_{i=1}^{n} w_i x_i - \theta} \qquad (3.5.1)$$

Η συνάρτηση ενεργοποίησης εξαρτάται από την τιμή της συνάρτησης διέγερσης, είναι μη γραμμική και έχει διωνυμική μορφή:

$$F(u)=\begin{cases}+1 & \alpha\nu \quad u>0\\ -1 & \alpha\nu \quad u\le 0\end{cases} \tag{3.5.2}$$

Η έξοδος δηλαδή του αλγορίθμου (αντίστοιχο με το σήμα που μεταδίδεται στον άξονα του νευρώνα) είναι η τιμή +1 ή -1 ανάλογα με την τιμή της διέγερσης.



Εικόνα 3.5.2: Αλγόριθμος Perceptron.

Η εξίσωση 3.5.1 μπορεί να απλοποιηθεί, θεωρώντας το κατώφλι θ ως ένα επιπλέον στοιχείο κάθε προτύπου με τιμή -1, δηλαδή $x_o=-1$. Στην περίπτωση αυτή τα διανύσματα των πρότυπων παραδειγμάτων και των συναπτικών βαρών θα είναι: $\vec{p}'=[x_{0,}x_1,x_2,...,x_n]$ και $\vec{w}'=[w_0,w_1,w_2,...,w_n]^T$ ενώ η συνάρτηση διέγερσης γίνεται:

$$u=\sum_{i=0}^{n} w_i x_i \tag{3.5.3}$$

Η εξίσωση 3.5.3 είναι το εσωτερικό γινόμενο των διανυσμάτων $\vec{p}'$ και $\vec{w}'$. Επομένως, όπως και στις μηχανές διανυσμάτων υποστήριξης (support

vectors machines), η εξίσωση $\left\langle \vec{w}', \vec{p}' \right\rangle = 0$ αντιστοιχεί σε ένα υπερεπίπεδο στο χώρο n διαστάσεων, το οποίο διέρχεται από το σημείο μηδέν. Τα πρότυπα, για τα οποία $u > 0$, βρίσκονται από τη μια πλευρά του υπερεπιπέδου και αντιστοιχούν στην κλάση $y = +1$ ενώ τα πρότυπα, για τα οποία $u < 0$, βρίσκονται στην αντίθετη πλευρά με κλάση $y = -1$. Σκοπός λοιπόν της εκπαίδευσης είναι να βρεθούν τα διανύσματα των συναπτικών βαρών για τα οποία διαχωρίζονται σωστά οι κλάσεις.

Η εκπαίδευση του αλγόριθμου Perceptron γίνεται με έναν επαναληπτικό κανόνα. Τα πρότυπα παραδείγματα εισέρχονται με κυκλική σειρά στο δίκτυο και υπολογίζονται τα συναπτικά τους βάρη. Όταν ολοκληρωθεί ένας πλήρης ***κύκλος (round)*** χρήσης όλων των προτύπων, γίνεται επανάληψη της διαδικασίας. Σκοπός του επαναληπτικού κανόνα είναι, σε κάθε κύκλο, να διορθώνονται τα συναπτικά βάρη του κάθε προτύπου έτσι ώστε το σήμα εξόδου, y, του αλγορίθμου να είναι το ίδιο με την κλάση του προτύπου ($y_i = +1$ ή $y_i = -1$).

Η διόρθωση ενός διανύσματος συναπτικών βαρών κατά τον κύκλο k γίνεται με πρόσθεση ενός ποσοστού b του προτύπου στο διάνυσμα συναπτικών βαρών του κύκλου k-1:

$$\boxed{\vec{w}'^{\,k} = \vec{w}'^{\,k-1} + b(y_i - y)\vec{p}'} \qquad (3.5.4)$$

όπου b μια σταθερά που ονομάζεται ***ρυθμός εκπαίδευσης (learning rate)***. Με τον τρόπο αυτό, το σήμα εξόδου στο νέο κύκλο θα πλησιάσει την κλάση του προτύπου και οι επαναλήψεις θα σταματήσουν όταν $y = y_i$.

3.6. κ-Κοντινότερος-Γείτονας Ταξινομητής

3.6.1. Εισαγωγή

Οι ταξινομητές διακρίνονται σε δυο κατηγορίες, ανάλογα με τον τρόπο εκτέλεσης της ταξινόμησης. Οι ***γρήγοροι ταξινομητές (eager classifiers),*** κατά την διαδικασία της μάθησης με πρότυπα παραδείγματα, αναπτύσσουν ένα μοντέλο, το οποίο στο μέλλον θα ταξινομήσει άγνωστα παραδείγματα. Οι ***αργοί ταξινομητές (lazy classifiers)*** δεν επιμένουν στην διαδικασία της μάθησης. Όταν δέχονται μια ομάδα πρότυπων παραδειγμάτων με σκοπό να εκπαιδευτούν, αποθηκεύουν τα πρότυπα, χωρίς περαιτέρω επεξεργασία. Μόνο όταν τους ζητηθεί να ταξινομήσουν άγνωστα παραδείγματα, χρησιμοποιούν τα πρότυπα. Επειδή τα πρότυπα παραδείγματα δεν χρησιμοποιούνται για την ανάπτυξη κάποιου μοντέλου, αλλά η χρήση τους γίνεται την ώρα της ταξινόμησης, οι αργοί ταξινομητές μπορεί να είναι δαπανηροί υπολογιστικά. Ένα παράδειγμα αργού ταξινομητή είναι ο ***Κοντινότερος-Γείτονας (Nearest-Neighbor),*** ο οποίος περιγράφηκε για πρώτη φορά στις αρχές της δεκαετίας το ΄50. Ο κοντινότερος γείτονας κατατάσσει τα άγνωστα παραδείγματα με βάση την ομοιότητα-αναλογία τους με τα πρότυπα της πληροφορίας εκπαίδευσης.

3.6.2. Ταξινόμηση

Η βασική ιδέα του κοντινότερου-γείτονα ταξινομητή είναι πως ένα άγνωστο παράδειγμα $\vec{x}$ ταξινομείται στην κλάση y, στην οποία ανήκει ο κοντινότερος γείτονάς του. Όταν λαμβάνονται υπόψη περισσότεροι από ένας γείτονες, έστω k, ο ταξινομητής ονομάζεται ***κ-Κοντινότερος-Γείτονας (k-Nearest-Neighbor)***.

Κάθε πρότυπο διάνυσμα απεικονίζεται με ένα σημείο στον χώρο, δημιουργώντας ένα διάγραμμα σημείων. Όταν ο ταξινομητής δέχεται μια

ομάδα άγνωστων παραδειγμάτων, «ψάχνει» στο διάγραμμα των πρότυπων παραδειγμάτων τα κ-κοντινότερα πρότυπα για το κάθε παράδειγμα. Το μέγεθος που ορίζει «πόσο κοντά» είναι ένα πρότυπο σε ένα άγνωστο παράδειγμα είναι μια σημαντική παράμετρος του αλγορίθμου.

Όταν τα χαρακτηριστικά των παραδειγμάτων είναι διακριτές μεταβλητές, ένας απλός τρόπος εντοπισμού κοντινών γειτόνων είναι ο εξής [44]: Έστω ότι η τιμή ενός χαρακτηριστικού A_k για ένα άγνωστο παράδειγμα $\vec{x} = [x_1, x_2, ..., x_n]$ είναι a_x. Τότε η συνάρτηση, η οποία υπολογίζει την γειτνίαση του παραδείγματος $\vec{x}$ με κάθε πρότυπο, $\vec{p} = [p_1, p_2, ..., p_n]$, για το συγκεκριμένο χαρακτηριστικό είναι: $d\left(\vec{x}, \vec{p}\right) = \begin{cases} 1 & \alpha\nu \quad a_x = a_p \\ 0 & \alpha\nu \quad a_x \neq a_p \end{cases}$. Το άθροισμα των συναρτήσεων $d\left(\vec{x}, \vec{p}\right)$ όλων των χαρακτηριστικών, για ένα ζεύγος αγνώστου παραδείγματος-προτύπου δίνει το μέτρο της γειτνίασής τους.

Αν τα χαρακτηριστικά είναι συνεχείς μεταβλητές, η γειτνίαση του $\vec{x}$ με κάθε πρότυπο $\vec{p}$, μπορεί να υπολογιστεί με την ευκλείδεια απόστασή τους:

$$d\left(\vec{x}, \vec{p}\right) = \sqrt{\sum_{i=1}^{n} \left(x_i - p_i\right)^2} \qquad (3.6.1)$$

Ένας γενικός τύπος της απόστασης δυο παραδειγμάτων είναι η εξίσωση Minkowski:

$$d\left(\vec{x}, \vec{p}\right) = \left(\sum_{i=1}^{n} \left|x_i - p_i\right|^m\right)^{\frac{1}{m}} \qquad (3.6.2)$$

όπου για $m = 1$ είναι η απόσταση Manhattan και για $m = 2$ η ευκλείδεια απόσταση (εξίσωση 3.6.1).

Όταν βρεθούν οι κ-κοντινότεροι-γείτονες για ένα παράδειγμα, ο ταξινομητής είναι έτοιμος να το κατατάξει σε μια από τις κλάσεις των

γειτόνων. Ο πιο απλός τρόπος είναι η ταξινόμηση στην κλάση, στην οποία ανήκουν οι περισσότεροι γείτονες. Ένας πιο ευαίσθητος τρόπος ταξινόμησης του παραδείγματος είναι να ληφθεί υπόψη η επίδραση του κάθε γείτονα στην τελική απόφαση [55]. Ως μέτρο της επίδρασης ορίζεται το αντίστροφο της απόστασης, δηλαδή όσο μικρότερη η απόσταση των δυο παραδειγμάτων, τόσο μεγαλύτερη η επίδραση του προτύπου στην ταξινόμηση του αγνώστου παραδείγματος. Για κάθε κλάση υπολογίζεται η παρακάτω σχέση:

$$\mathrm{Vote}(y_i) = \sum_{k=1}^{n} \left[\frac{1}{d\left(\vec{x}, \vec{p}_k\right)^m} \times 1\left(y_x, y_k\right) \right] \tag{3.6.3}$$

όπου η σχέση $1\left(y_x, y_k\right)$ λαμβάνει την τιμή 1 για $y_x = y_k$ και την τιμή 0 για $y_x \neq y_k$. Ο εκθέτης m είναι συνήθως μονάδα, ενώ όσο μεγαλώνει μειώνεται η επίδραση των πιο μακρινών γειτόνων. Τελικά, το παράδειγμα ταξινομείται στην κλάση με την μεγαλύτερη τιμή $\mathrm{Vote}\left(y_i\right)$.

Επανερχόμενοι στην ευκλείδεια απόσταση δυο παραδειγμάτων, το μέτρο αυτό μπορεί να γραφτεί ως εξής:

$$d\left(\vec{x}, \vec{p}\right) = \left\| \vec{x} - \vec{p} \right\| = \sqrt{\left\langle \vec{x}, \vec{x} \right\rangle + \left\langle \vec{p}, \vec{p} \right\rangle - 2\left\langle \vec{x}, \vec{p} \right\rangle} \tag{3.6.4}$$

Τα εσωτερικά γινόμενα στην εξίσωση 3.6.4 μπορούν να αντικατασταθούν με συναρτήσεις πυρήνα. Όταν η πληροφορία εκπαίδευσης είναι πολύπλοκη, η αλλαγή αυτή τροποποιεί την κατανομή των πρότυπων παραδειγμάτων και μπορεί να αυξήσει την απόδοση του αλγόριθμου. [56]

4

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΛΟΓΙΣΜΙΚΑ

«Ελπίζουμε πως η γενική ιδέα της Γονιδιακής Οντολογίας (Gene Ontology), ειδικά οι διαχωρισμοί μεταξύ των βιολογικών, μοριακών και κυτταρικών διαδικασιών, θα βρει υποστηρικτές ανάμεσα στους βιολόγους, έτσι ώστε όλοι μαζί να διευκολύνουμε την μεγάλη ενοποίηση της βιολογίας, την οποία προμηνύουν οι αποκρυπτογραφήσεις των γονιδιώματων των οργανισμών.»

The Gene Ontology Consortium, "*Gene Ontology: tool for the unification of biology*"

4.1. ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

4.1.1. Βιβλιοθήκη UniProt

Η **UniProt** (http://www.uniprot.org/) αποτελεί μια κοινοπραξία του European Bioinformatics Institute (EBI), του Swiss Institute of Bioinformatics (SIB) και του Protein Information Resource (PIR) με σκοπό την παροχή περιεκτικών πληροφοριών και αλληλουχιών για γνωστές πρωτεΐνες. Η UniProt αποτελείται από τέσσερις συνιστώσες: την κεντρική βάση δεδομένων: **UniProt Knowledgebase (UniProtKB)**, την βάση δεδομένων ομαδοποιημένων αλληλουχιών: **UniProt Reference Clusters (UniRef)**, την βάση δεδομένων όλων των διαθέσιμων πληροφοριών που υπάρχουν δημοσιευμένες για κάθε πρωτεΐνη: **UniProt Archive (UniParc)** και την βάση δεδομένων μεταγενομικών και περιβαλλοντικών δειγμάτων: **UniProt Metagenomic and Enviromental Sequences (UniMES)**. [27]

Η UniProtKB είναι μια εξειδικευμένη βάση πληροφοριών για γνωστές πρωτεΐνες. Κάθε εγγραφή στην UniProtKB χαρακτηρίζεται από μια πληθώρα δεδομένων, όπως την αλληλουχία αμινοξέων, το όνομα και την περιγραφή της πρωτεΐνης, ταξινομία οργανισμών, βιολογικές οντολογίες, ταξινομήσεις, παραπομπές καθώς και ενδείξεις της ποιότητας των παρεχόμενων πληροφοριών. Χωρίζεται σε δυο συνιστώσες την **UniProt/Swiss-Prot** και την **UniProt/TrEMBL**. Η UniProt/Swiss-Prot περιέχει υψηλής ποιότητας, χειροκίνητα χαρακτηρισμένες και χωρίς περιττές πληροφορίες αλληλουχίας πρωτεϊνών. Η αλληλουχία κάθε εγγραφής της UniProt/Swiss-Prot έχει προκύψει από συνένωση όλων των διαθέσιμων αλληλουχιών για τη συγκεκριμένη εγγραφή, ενώ τα χαρακτηριστικά της προέρχονται από την βιβλιογραφία και πολλαπλές υπολογιστικές αναλύσεις. Αντίθετα η UniProt/TrEMBL περιέχει εγγραφές οι οποίες έχουν αναλυθεί με υπολογιστικά εργαλεία και ο χαρακτηρισμός και η ταξινόμησή τους έχει γίνει αυτόματα.

Η βάση δεδομένων UniRef περιέχει εγγραφές οι οποίες έχουν προκύψει από την αυτόματη συνένωση των εγγραφών της UniProtKB και

επιλεγμένων εγγραφών της UniParc. Η UniRef αποτελείται από τρεις συνιστώσες, την **UniRef100**, **UniRef90** και **UniRef50**. Σκοπός της UniRef είναι η μείωση του πλήθους των αλληλουχιών μέσω της συνένωσής τους με ορισμένο ποσοστό ομοιότητας. Η συνένωση των αλληλουχιών πραγματοποιείται με τον λογισμικό CD-HIT. Η UniRef100 κατασκευάζεται με την συνένωση αλληλουχιών που έχουν ποσοστό ομοιότητας 100%. Η UniRef90 κατασκευάζονται από την UniRef100 με συνένωση αλληλουχιών οι οποίες έχουν ποσοστό ομοιότητας >90% και παρόμοια η UniRef50 κατασκευάζεται από την UniRef90 με ελάχιστη ομοιότητα 50%. Με τον τρόπο αυτό το μέγεθος ενός συνόλου αλληλουχιών μειώνεται περίπου κατά 40% στην UniRef90 και 65% στην UniRef50. Οι συνιστώσες της UniRef αποτελούν βάσεις δεδομένων χωρίς περιττές εγγραφές και είναι πιο εύχρηστες υπολογιστικά. [58]

4.1.2. Βάση Δεδομένων Pfam

Η **Pfam** (http://pfam.sanger.ac.uk/) είναι μια βάση δεδομένων πρωτεϊνικών οικογενειών, για την κατασκευή των οποίων χρησιμοποιείται ο αλγόριθμος πολλαπλής στοίχισης με Κρυφά Μαρκοβιανά Μοντέλα (HMMs). Η δημιουργία των οικογενειών γίνεται με την χρήση του λογισμικού HMMER. Κύριο χαρακτηριστικό κάθε πρωτεϊνικής οικογένειας είναι οι πρωτεϊνικές περιοχές (domains) που περιέχονται στα μέλη της. Επιπλέον κάθε οικογένεια χαρακτηρίζεται από τους διαφορετικούς τρόπους συνύπαρξης πρωτεϊνικών περιοχών στις αλληλουχίες της (architectures), από τις επιμέρους αλληλουχίες, τα ταξινομικά τους χαρακτηριστικά, τις φυλογενετικές τους σχέσεις, τις βιοχημικές αντιδράσεις που συμμετέχουν και τις τριτοταγείς δομές τους.

Η Pfam διακρίνεται σε δυο τμήματα την **Pfam-A** και την **Pfam-B**. Η βάση Pfam-A περιέχει χειροκίνητα χαρακτηρισμένες πρωτεϊνικές οικογένειες, ενώ η Pfam-B αυτόματα χαρακτηρισμένες. Όπως προκύπτει η Pfam-B είναι κατώτερη ποιοτικά από την Pfam-A, όμως περιέχει πρωτεϊνικές οικογένειες οι οποίες δεν έχουν ακόμα τοποθετηθεί στην Pfam-A και μπορεί να χρησιμοποιηθεί όταν δεν επαρκούν τα δεδομένα της Pfam-A. Η τελευταία

έκδοση της Pfam-A (11/2011) περιέχει 13672 οικογένειες, πολλές από τις οποίες περιέχουν πάνω από 100.000 μέλη, ενώ πάνω από το 75% των γνωστών πρωτεϊνών περιέχονται στη συγκεκριμένη βάση. Επιπλέον η Pfam-B βάση περιέχει 460125 οικογένειες. Επιπλέον οι εγγραφές της Pfam ταξινομούνται σε φυλές (clans). Οι φυλές αποτελούνται από σχετικές-ομόλογες πρωτεϊνικές οικογένειες. [59]

Κατά της χρήση της Pfam για ανάλυση άγνωστων αλληλουχιών, τα δεδομένα που χρησιμοποιούνται από την βάση είναι οι αλληλουχίες των πρωτεϊνικών περιοχών και οι πίνακες τιμών συγκεκριμένης θέσης (PSSM), με τη χρήση των οποίων αξιολογούνται ποιοτικά οι στοιχίσεις.

4.1.3. Οντολογία Γονιδίων

Η ***Οντολογία Γονιδίων (Gene Ontology)*** (http://www.geneontology.org/) είναι ένα λεξιλόγιο ενοποίησης και χαρακτηρισμού των ρόλων των γονιδίων και των παραγώγων τους σε μια πληθώρα οργανισμών. Η αποκρυπτογράφηση γονιδιωμάτων ευκαριωτικών οργανισμών (Saccharomyces cerevisiae, Caenorhabditis elegans, Drosophila Melanogaster) στα τέλη της δεκαετίας του ‘90, αποκάλυψε πως ένα μεγάλο ποσοστό των γονιδίων ενός μικροοργανισμού έχει αυξημένα ποσοστά ομοιότητας με αντίστοιχα γονίδια σε άλλους μικροοργανισμούς. Τα γονίδια αυτά κωδικοποιούσαν πρωτεΐνες, οι οποίες εκτελούσαν σημαντικές κυτταρικές λειτουργίες, όπως η αντιγραφή του DNA, η μετάφραση και ο μεταβολισμός. Οι ομοιότητες των γονιδίων σε διαφορετικούς μικροοργανισμούς και το γεγονός ότι πολλές κυτταρικές λειτουργίες πραγματοποιούνται με τον ίδιο τρόπο στους μικροοργανισμούς και τα θηλαστικά, γνώση η οποία προήλθε από πειράματα την δεκαετία του ‘80, έδειξαν την ανάγκη ενοποίησης των κοινών γονιδίων, γονιδιακών παραγώγων και λειτουργιών σε ένα ενιαίο λεξιλόγιο.

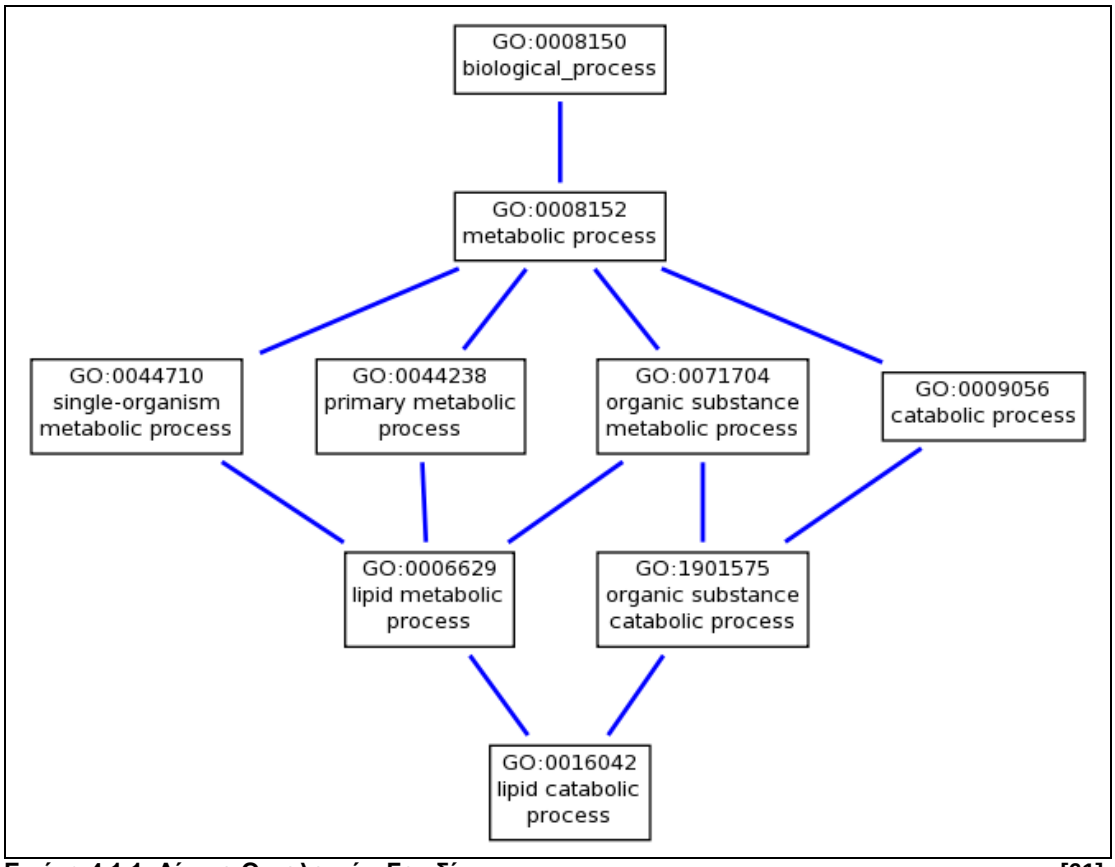
Η Οντολογία Γονιδίων ξεκίνησε ως μια κοινοπραξία των τριών βάσεων δεδομένων των γονιδιωμάτων που αποκρυπτογραφήθηκαν στα τέλη της δεκαετίας του ’90. Σκοπός ήταν η παροχή πληροφοριών για γονίδια και παράγωγα γονιδίων σε ένα ενιαίο λεξιλόγιο για όλους τους οργανισμούς.

Σήμερα το πρόγραμμα της Οντολογίας Γονιδίων στηρίζεται από πολλά ινστιτούτα. Το υπάρχον λεξιλόγιο δίνει την δυνατότητα εύρεσης και τεκμηρίωσης της λειτουργίας ενός γονιδίου ή παραγώγου του σε έναν οργανισμό, του οποίου το γονιδίωμα δεν είναι γνωστό. Επομένως η Οντολογία Γονιδίων μέσω της ομοιότητας μιας άγνωστης αλληλουχίας με γνωστές, παρέχει πληροφορίες για την πιθανή λειτουργία της άγνωστης. Κάθε όρος στο λεξιλόγιό της (GO term) παρέχει ορισμένες πληροφορίες. Οι όροι χωρίζονται σε τρείς κατηγορίες, ανάλογα με το είδος της πληροφορίας: κυτταρικό συστατικό (cellular component), μοριακή λειτουργία (molecular function) και βιολογική διεργασία (biological process).

Οι όροι που ανήκουν στην κατηγορία του κυτταρικού συστατικού αναφέρουν το μέρος του κυττάρου στο οποίο ενεργεί ένα γονιδιακό προϊόν. Οι όροι αυτοί βοηθούν στην κατανόηση της δομής των ευκαρυωτικών κυττάρων. Ορισμένα παραδείγματα είναι το ριβόσωμα (ribosome), η πυρηνική μεμβράνη (nuclear membrane) και το σύστημα Golgi (Golgi apparatus). Οι όροι της μοριακής λειτουργίας ορίζουν τη βιοχημική δραστηριότητα ενός γονιδιακού προϊόντος χωρίς να προσδιορίζουν τον χώρο και τον χρόνο της δραστηριότητας. Ορισμένα παραδείγματα είναι οι όροι ενζυμο (enzyme), μεταφορέας (transporter) και συνδέτης (ligand). Τέλος οι όροι της βιολογικής διεργασίας δηλώνουν την διεργασία στην οποία συνεισφέρει ένα γονίδιο ή παράγωγό του, όπως κυτταρική ανάπτυξη (cell growth), μεταβολισμός πυριμιδίνης (pyrimidine metabolism) και μετάφραση (translation).

Ένα προϊόν γονιδίου μπορεί να χαρακτηρίζεται και από τις τρεις κατηγορίες οντολογίας και μάλιστα με περισσότερους από έναν όρο στην καθεμία. Ο συνδυασμός των τριών κατηγοριών δίνει μια πλήρη εικόνα των λειτουργιών, των χώρων και των διεργασιών όπου συμμετέχει μια πρωτεΐνη. Επιπλέον οι όροι Οντολογίας Γονιδίων έχουν σχέση προγόνου-απογόνου. Συνδέονται μεταξύ τους σε ένα δίκτυο κόμβων, όπου οι κόμβοι πρόγονοι εξειδικεύονται σε κόμβους απογόνους, οι οποίοι παρέχουν πιο λεπτομερείς πληροφορίες. Στην παρακάτω εικόνα ο όρος της βιολογικής διεργασίας έχει ως απόγονο τον όρο της μεταβολικής διεργασίας, ο οποίος έχει τέσσερις απογόνους. Το δίκτυο καταλήγει από τον γενικό όρο της βιολογικής

διεργασίας στον όρο διεργασίας καταβολισμού λιπιδίου με πέντε διαφορετικά μονοπάτια. [60]



Εικόνα 4.1.1: Δίκτυο Οντολογιών Γονιδίων. **[61]**

Η κοινοπραξία της Οντολογίας Γονιδίων έχει πλέον αποκτήσει πολλά μέλη, τα οποία δημιούργησαν μια νέα μορφή βάσεων δεδομένων, τα ***Χαρακτηριστικά Οντολογίας Γονιδίων (Gene Ontology Annotation, GOA)*** (http://www.geneontology.org/GO.downloads.annotations.shtml). Οι βάσεις δεδομένων GOA περιέχουν πρωτεΐνες από βιβλιοθήκες όπως η UniProt, η PAMGO, η EcoCyc & EcoliHub, η FlyBase, η SGD, η dictyBase, η WormBase, η Sanger GeneBD κ.α., χαρακτηρισμένες με όρους Οντολογίας Γονιδίων. Αυτές οι βάσεις δεδομένων αφορούν συνήθως έναν συγκεκριμένο οργανισμό. Η UniProt συμμετέχει στην κοινοπραξία από το 2001. Το 2003 δημιούργησε την UniProt-GOA, η οποία περιέχει εγγραφές της UniProt, χωρίς τις αλληλουχίες, χαρακτηρισμένες με όρους Οντολογίας Γονιδίων [62]. Η GOA περιέχει πρωτεΐνες διαφόρων οργανισμών.

4.1.4. Ενζυμική Ονοματολογία

Η ***Ενζυμική Ονοματολογία (Enzyme Nomenclature)*** (http://www.chem.qmul.ac.uk/iubmb/enzyme/) είναι ένα σύστημα ταξινόμησης των ενζύμων με βάση την χημική αντίδραση που καταλύουν. Από τα τέλη της δεκαετίας του ’50 ο ρυθμός ανακάλυψης νέων ενζύμων αυξήθηκε σημαντικά. Οι ερευνητές ονομάτιζαν τα ένζυμα που ανακάλυπταν χωρίς την χρήση κάποιου ενιαίου συστήματος ονοματολογίας και το αποτέλεσμα ήταν διαφορετικά ένζυμα να έχουν το ίδιο όνομα και πανομοιότυπα ένζυμα να έχουν διαφορετικά ονόματα. Επιπλέον ορισμένα ονόματα δεν παρείχαν καμία πληροφορία για την χημική δραστηριότητα των ενζύμων, ενώ άλλα δεν ήταν αρκετά λεπτομερή.

Για τους παραπάνω λόγους, η Διεθνής Ένωση Βιοχημείας και Μοριακής Βιολογίας (IUBMB) πρότεινε το 1955 την εγκαθίδρυση ενός ενιαίου συστήματος ταξινόμησης των ενζύμων με βάση την λειτουργίας τους. Η πρώτη λίστα Ενζυμικής Ονοματολογίας εκδόθηκε το 1961, ενώ μετά από πολυετή επεξεργασία, η τελική λίστα εκδόθηκε το 1992. Νέες καταχωρίσεις και διαγραφές από την λίστα των ενζυμικών κατηγοριών πραγματοποιούνται μέχρι και σήμερα. Η έκδοση 28/11/2012 περιέχει 5007 εγγραφές. [63]

Κάθε εγγραφή της Ενζυμικής Ονοματολογίας περιγράφεται με έναν τετραψήφιο αριθμό ενζυμικής ταξινόμησης και αντιστοιχεί σε μια χημική αντίδραση που καταλύεται από ένζυμο. Συνήθως το όνομα του τετραψήφιου αριθμού είναι το όνομα του ενζύμου. Οι αριθμοί των έξι μεγάλων ενζυμικών κατηγοριών είναι:

Πίνακας 4.1.1: Βασικές κατηγορίες Ενζυμικής Ονοματολογίας

Κατηγορία ενζύμων	Κωδικός EC
Οξειδοαναγωγάσες (Oxidoreductases)	1. -. -.-
Τρανσφεράσες (Transferases)	2. -. -.-
Υδρολάσες (Hydrolases)	3. -. -.-
Λυάσες (Lyases)	4. -. -.-
Ισομεράσες (Isomerases)	5. -. -.-
Λιγάσες (Ligases)	6. -. -.-

Το πρώτο ψηφίο του αριθμού δηλώνει την γενική κατηγορία χημικής αντίδρασης, την οποία καταλύει το ένζυμο. Τα δυο επόμενα ψηφία ορίζουν την υποκατηγορία της χημικής αντίδρασης, ενώ το τέταρτο ψηφίο δηλώνει το υπόστρωμα. Για παράδειγμα η κατηγορία 3.-.-.- αναφέρεται στις υδρολάσες και η υποκατηγορία 3.1.-.- αναφέρεται στις υδρολάσες εστερικών δεσμών, ενώ η υποκατηγορία 3.1.1.- αναφέρεται στις υδρολάσες καρβοξυλικών εστερικών δεσμών. Συνολικά για τις υδρολάσες υπάρχουν 13 υποκατηγορίες.

Η Ενζυμική Ονοματολογία χρησιμοποιείται ευρέως από πολλές βάσεις δεδομένων, καθώς αποτελεί σημαντικό χαρακτηριστικό για τον πλήρη χαρακτηρισμό ενός ενζύμου, μιας χημικής αντίδρασης ή ενός μεταβολικού μονοπατιού. Επιπλέον Η Οντολογία Γονιδίων έχει δημιουργήσει μια βάση αντιστοίχισης των όρων GO με τους EC αριθμούς, την ec2go (http://www.geneontology.org/external2go/ec2go). Για παράδειγμα:

EC:3 > GO:hydrolase activity ; GO:0016787
EC:3.1 > GO:hydrolase activity, acting on ester bonds ; GO:0016788
EC:3.2 > GO:hydrolase activity, acting on glycosyl bonds ; GO:0016798
EC:3.3 > GO:hydrolase activity, acting on ether bonds ; GO:0016801
EC:3.4 > GO:peptidase activity ; GO:0008233
EC:3.5 > GO:hydrolase activity, acting on carbon-nitrogen (but not peptide) bonds ; GO:0016810
EC:3.6 > GO:hydrolase activity, acting on acid anhydrides ; GO:0016817
EC:3.7 > GO:hydrolase activity, acting on acid carbon-carbon bonds ; GO:0016822
EC:3.8 > GO:hydrolase activity, acting on acid halide bonds ; GO:0016824
EC:3.9 > GO:hydrolase activity, acting on acid phosphorus-nitrogen bonds ; GO:0016825
EC:3.10 > GO:hydrolase activity, acting on acid sulfur-nitrogen bonds ; GO:0016826
EC:3.11 > GO:hydrolase activity, acting on acid carbon-phosphorus bonds ; GO:0016827
EC:3.12 > GO:hydrolase activity, acting on acid sulfur-sulfur bonds ; GO:0016828
EC:3.13 > GO:hydrolase activity, acting on carbon-sulfur bonds ; GO:0046508

4.2. ΛΟΓΙΣΜΙΚΑ

4.2.1 Πακέτο Λογισμικών BLAST

Το **BLAST (Basic Local Alignment Search Tool)** είναι ένα πακέτο βιοπληροφορικών προγραμμάτων, τα οποία χρησιμοποιούνται για εύρεση ομολογιών μεταξύ άγνωστων-εξεταζόμενων και γνωστών αλληλουχιών. Σχεδιάστηκε στις Η.Π.Α., το 1990, από τους Stephen Altschul, Warren Gish, Webb Miller, Eugene Myers, και David J. Lipman και αποτελεί το πιο ευρέως χρησιμοποιούμενο εργαλείο βιοπληροφορικής. Όπως προκύπτει από το όνομά του, τα προγράμματα του BLAST εκτελούν τοπική στοίχιση κατά την εξέταση αλληλουχιών. Το BLAST είναι διαθέσιμο στο διαδίκτυο από το Εθνικό Κέντρο Πληροφορίας Βιοτεχνολογίας των ΗΠΑ (National Center for Biotechnology Information, NCBI), (http://blast.ncbi.nlm.nih.gov/Blast.cgi), ή τοπικά με την αυτόνομη έκδοση (standalone blast+), ως ελεύθερο λογισμικό.

Η βασική αρχή λειτουργίας των προγραμμάτων του BLAST βασίζεται στην μέθοδο k-γραμμάτων, την τοπική στοίχιση και τον δυναμικό προγραμματισμό. Αρχικά η εξεταζόμενη αλληλουχία διαχωρίζεται σε μικρότερες υποομάδες k γραμμάτων. Η παράμετρος k επιλέγεται από τον χρήστη και όσο μικρότερη είναι, τόσο πιο ευαίσθητη γίνεται η αναζήτηση ομοιοτήτων. Στην βάση γνωστών αλληλουχιών που χρησιμοποιείται, αναζητούνται τμήματα των αλληλουχιών όμοια με μια υποομάδα. Αν βρεθεί ομοιότητα, υπολογίζεται το σκορ στοίχισης με χρήση πινάκων αντικατάστασης και αν το σκορ είναι μεγαλύτερο από ένα κατώφλι, τότε η στοίχιση των δυο αλληλουχιών συνεχίζεται και προς τις δυο κατευθύνσεις τους με σκοπό την εύρεση περιοχών υψηλής ομοιότητας. Με τον τρόπο αυτό το πρόγραμμα BLAST δεν εξετάζει ολόκληρη την βάση γνωστών αλληλουχιών και η αναζήτηση ομοιοτήτων πραγματοποιείται πιο γρήγορα. Το παρακάτω παράδειγμα δείχνει τον αλγόριθμο λειτουργίας των προγραμμάτων BLAST:

Έστω k=3, ο πίνακας αντικατάστασης είναι ο BLOSUM62 και το κατώφλι του σκορ στοίχισης είναι 10:

- Βήμα 1^{ο}: εύρεση ομοιότητας με την μέθοδο k-γραμμάτων

```
Query  33   VFRTYLTASSTFPVLSMISITFLCTSL  59
               YLT
Sbjct  166  AWREYLTASAAAPVVQAVEVSYIDTAV  192
```

σκορ στοίχισης = 4+4+5=11 > 10

- ΒΗΜΑ 2^{ο}: Συνέχιση της στοίχισης προς τις δυο κατευθύνσεις:

```
Query  33   VFRTYLTASSTFPVLSMISITFLCTSL  59
            +R YLTAS+  PV+  + ++++ T++
Sbjct  166  AWREYLTASAAAPVVQAVEVSYIDTAV  192
```

συνολικό σκορ στοίχισης = 30

Το συνολικό σκορ στοίχισης είναι ένα σημαντικό μέτρο αξιολόγησης της στοίχισης. Το BLAST ονομάζει το σκορ αυτό **bit score** και όσο μεγαλύτερο είναι τόσο πιο ποιοτική είναι η στοίχιση. Το bit score υπολογίζεται από τους πίνακες αντικατάστασης και τις ποινές των κενών. Ένα επιπλέον μέτρο που αξιολογεί την στοίχιση είναι το **e-value**, το οποίο δείχνει την πιθανότητα η συγκεκριμένη στοίχιση να συμβαίνει κατά τύχη. Για παράδειγμα η τιμή e-value=0.05 δηλώνει πως υπάρχει πιθανότητα 5% η στοίχιση να συνέβη κατά τύχη και οι δυο αλληλουχίες να μην έχουν κάποια ομολογία μεταξύ τους. Όσο η τιμή του e-value προσεγγίζει το μηδέν μειώνεται η πιθανότητα τυχαίας στοίχισης. [65]

Όπως προαναφέρθηκε το BLAST αποτελείται από πολλά διαφορετικά προγράμματα, καθένα από τα οποία χρησιμοποιείται για συγκεκριμένο σκοπό. Τα βασικά προγράμματα BLAST διαχωρίζονται σε δυο μεγάλες κατηγορίες ανάλογα με την φύση των εξεταζόμενων αλληλουχιών: 1) προγράμματα για αλληλουχίες νουκλεοτιδίων και 2) προγράμματα για αλληλουχίες αμινοξέων.

Το βασικό πρόγραμμα ανάλυσης νουκλεοτιδικών αλληλουχιών είναι το **blastn**. Το πρόγραμμα **megablast** έχει σχεδιαστεί ώστε να εντοπίζει

αποτελεσματικά μεγάλες σε μήκος στοιχίσεις μεταξύ παρόμοιων αλληλουχιών και για τον λόγο αυτό θεωρείται το πιο ιδανικό στην εύρεση ταυτόσημων αλληλουχιών. Το πρόγραμμα **discontiguous megablast** θεωρείται ιδανικό στην εύρεση παρόμοιων αλληλουχιών και όχι ταυτόσημων. Τέλος το πρόγραμμα **tblastx** μεταφράζει την εξεταζόμενη αλληλουχία νουκλεοτιδίων και αναζητά ομοιότητες σε βάσεις δεδομένων οι οποίες περιέχουν αλληλουχίες αμινοξέων.

Αντίστοιχα με τις νουκλεοτιδικές αλυσίδες, το βασικό πρόγραμμα ανάλυσης αλληλουχιών αμινοξέων είναι το **blastp**. Το πρόγραμμα **psi-blast** είναι το πιο ευαίσθητο πρόγραμμα σε αυτή την κατηγορία γιατί έχει σχεδιαστεί για εύρεση αλληλουχιών μακρινής συγγένειας και χρησιμοποιείται όταν το blastp αδυνατεί να βρει σημαντικές ομοιότητες για μια αλληλουχία. Το **phi-blast** χρησιμοποιείται για αναζήτηση μοτίβων αμινοξέων στις αλληλουχίες της βάσης δεδομένων, ενώ το **rps-blast** εξειδικεύεται στην εύρεση πρωτεϊνικών περιοχών (domains) στην εξεταζόμενη αλληλουχία. Το πρόγραμμα **cdart** βρίσκει πρωτεϊνικές περιοχές που υπάρχουν σε μια εξεταζόμενη αλληλουχία, εκτελώντας το rps-blast, και στη συνέχεια αναζητά σε βάσεις δεδομένων γνωστές πρωτεΐνες που περιέχουν αυτές τις πρωτεϊνικές περιοχές. Τέλος το πρόγραμμα **tblastn** αναζητά ομοιότητες της εξεταζόμενης αλληλουχίας αμινοξέων σε βάσεις δεδομένων που περιέχουν αλληλουχίες μεταφρασμένων νουκλεοτιδίων.[64]

4.2.2. Πακέτο Λογισμικών HMMER

Το **HMMER** (http://hmmer.janelia.org/search/phmmer) είναι ένα ελεύθερο πακέτο προγραμμάτων που χρησιμοποιούνται για αναζήτηση ομολογιών μεταξύ άγνωστων αλληλουχιών και πρωτεϊνικών οικογενειών ή κατασκευή πολλαπλών στοιχίσεων, εφαρμόζοντας την θεωρία των Κρυφών Μαρκοβιανών Μοντέλων (HMMs). Από τον 03/2010 είναι διαθέσιμη η έκδοση HMMER3, η οποία είναι περίπου εκατό φορές πιο γρήγορή από την προηγούμενη. Κατά την ανάλυση άγνωστων αλληλουχιών, τα αριθμητικά αποτελέσματα του HMMER είναι ίδιας λογικής με του BLAST, για κάθε στοίχιση υπολογίζεται ένα **e-value** και ένα **bit score**.

Τα βασικά εργαλεία ανάλυσης άγνωστων πρωτεϊνικών αλληλουχιών είναι το **phmmer**, το οποίο χρησιμοποιεί βάσεις δεδομένων ολόκληρων πρωτεϊνικών αλληλουχιών, το **hmmscan**, το οποίο χρησιμοποιεί βάσεις δεδομένων πρωτεϊνικών περιοχών ή μοτίβων, το **hmmsearch** που αναζητά πρωτεϊνικές περιοχές σε βάσεις δεδομένων ολόκληρων αλληλουχιών και το **jackhammer** το οποίο λειτουργεί όπως το **psi-blast**, αναζητώντας μακρινές ομοιότητες μεταξύ της εξεταζόμενης αλληλουχίας και των γνωστών αλληλουχιών της βάσης. Επιπλέον, το HMMER παρέχει εργαλεία κατασκευής βάσης δεδομένων πρωτεϊνικών περιοχών και πολλαπλής στοίχισης. Το **hmmalign** χρησιμοποιείται για την πολλαπλή στοίχιση αλληλουχιών με ένα συγκεκριμένο HMM. Το **hmmbuild** πραγματοποιεί πολλαπλή στοίχιση στις εξεταζόμενες αλληλουχίες με σκοπό την κατασκευή ενός νέου HMM, δηλαδή την εύρεση περιοχών υψηλής ομοιότητας και κατασκευής του αντίστοιχου PSSM πίνακα. Άλλα δυο εργαλεία είναι το **hmmfetch** που προσκομίζει όνομα-κωδικό ενός προφίλ HMM από μια βάση δεδομένων HMM και το **hmmstat**, το οποίο εμφανίζει τα στατιστικά δεδομένα κάθε προφίλ HMM μιας βάσης.

Το HMMER χρησιμοποιεί τιμές στοίχισης συγκεκριμένης θέσης, οι οποίες βασίζονται στο κάθε προφίλ HMM και όχι στην γενική λογική των πινάκων αντικατάστασης. Η ιδιότητα αυτή το καθιστά πιο ικανό στην εύρεση μακρινών ομολογιών από τα εργαλεία του BLAST. [66]

4.2.3 Λογισμικό PROFEAT

Το **PROFEAT (PROtein FEATures)** είναι ένα πρόγραμμα υπολογισμού φυσικοχημικών ιδιοτήτων των πρωτεϊνών και των πεπτιδίων με βάση την πρωτοταγή τους δομή. Το πρόγραμμα έχει αναπτυχθεί στο Εθνικό Πανεπιστήμιο Σιγκαπούρης (National University of Singapore) από το τμήμα Βιοπληροφορικής και Σχεδιασμού Φαρμάκων (Bioinformatics & Drug Design group, BIDD) και είναι διαθέσιμο μόνο στο διαδίκτυο (http://bidd.cz3.nus.edu.sg/cgi-bin/prof/protein/profnew.cgi). [67]

Τα φυσικοχημικά χαρακτηριστικά μιας αλληλουχίας αμινοξέα που υπολογίζει το πρόγραμμα PROFEAT διακρίνονται σε εννιά κατηγορίες:

1) Σύσταση αμινοξέων (amino acid composition): Περιλαμβάνει τα κλάσματα των 20 αμινοξέων στην εξεταζόμενη αλληλουχία.

2) Σύσταση διπεπτιδίων (dipeptide composition): Περιλαμβάνει τα κλάσματα των 400 διπεπτιδίων στην εξεταζόμενη αλληλουχία.

3) Περιγραφείς Αυτοσυσχέτισης (autocorrelation descriptors): Περιλαμβάνει τις κατανομές συγκεκριμένων ιδιοτήτων των αμινοξέων. Υπάρχουν τρεις διαφορετικοί τύποι περιγραφέων. Οι ιδιότητες που υπολογίζονται παρέχονται από την βάση δεδομένων AAindex.[68]

4) Σύσταση, μετάβαση και κατανομή (composition, transition and distribution): Τα αμινοξέα διαχωρίζονται σε τρεις κλάσεις (1,2 και 3) για καθεμία από της εξής ιδιότητες: υδροφοβικότητα (hydrophobicity), κανονικοποιημένος όγκος δεσμού van der Waals (normalized van der Waals volume) , πολικότητα (polarity), ικανότητα πόλωσης (polarizability), φορτίο (charge), δευτεροταγή δομή (secondary structure) και προσβασιμότητα διαλύτη (solvent accessibility). Με τον τρόπο αυτό, μια αλληλουχία αμινοξέων μετατρέπεται σε μια αλληλουχία των ψηφίων 1, 2 και 3. Ο όρος «σύσταση» αφορά κάθε ιδιότητα ξεχωριστά και περιλαμβάνει το ποσοστό του κάθε ψηφίου στην αλληλουχία. Ο όρος «μετάβαση» αφορά και αυτός κάθε ιδιότητα ξεχωριστά και δηλώνει την συχνότητα η κλάση A να ακολουθείται από την κλάση B ή αντίστροφα στην αλληλουχία. Τέλος ο όρος «κατανομή» αναφέρεται στην κατανομή των κλάσεων κάθε ιδιότητας στην εξεταζόμενη αλληλουχία.

5) Περιγραφείς οιονεί αλληλουχίας (quasi-sequence-order descriptors): Οι συγκεκριμένοι περιγραφείς προέρχονται από τους πίνακες απόστασης των αμινοξέων. [69][[70][71][72]

6) Ψευδο-σύσταση αμινοξέων (pseudo-amino acid composition): Περιλαμβάνει όρους που σχετίζονται με τις τιμές υδροφοβικότητας, υδροφιλικότητας και μοριακού βάρους των πλευρικών αλυσίδων των αμινοξέων της εξεταζόμενης αλληλουχίας. [73]

7) Αμφιφιλική ψευδο-σύσταση αμινοξέων (amphiphilic pseudo-amino acid composition): Περιλαμβάνει όρους που σχετίζονται με την αμφιφιλική συμπεριφορά της εξεταζόμενης αλληλουχίας. [74]

8) Τοπολογικοί περιγραφείς σε ατομικό επίπεδο (topological descriptors at atomic level): Οι όροι αυτοί αφορούν την δευτεροταγή δομή

της εξεταζόμενης αλληλουχίας, η οποία υπολογίζεται αυτόματα από το ίδιο το πρόγραμμα.

9) Συνολικές ιδιότητες αμινοξέων (total amino acid properties): Οι όροι αυτοί αναφέρονται στο άθροισμα των ιδιοτήτων κάθε αμινοξέος της εξεταζόμενης αλληλουχίας. [75]

4.2.4. Λογισμικό RAPIDMINER

Το **RAPIDMINER** (http://rapid-i.com/content/view/181/190/) είναι ελεύθερο λογισμικό κατασκευής αλγόριθμων μηχανικής μάθησης (machine learning), εξόρυξης πληροφορίας (data mining), ανάλυσης και επεξεργασίας πληροφορίας (data analysis). Η ανάπτυξη του λογισμικού ξεκίνησε με το όνομα YALE (Yet Another Learning Environment) το 2001 υπό την καθοδήγηση της καθηγήτριας Katharina Morik στο Τεχνολογικό Πανεπιστήμιο του Ντόρτμουντ (TU Dortmund University). Σήμερα είναι ένα από τα πιο χρησιμοποιούμενα ελεύθερα λογισμικά εξόρυξης πληροφορίας.

Είναι γραμμένο σε γλώσσα java. Ένα σημαντικό πλεονέκτηματά του είναι η μεγάλη ευελιξία που προσφέρει στον χρήστη, κατά την διάρκεια κατασκευής μιας διαδικασίας ανάλυσης. Ο χρήστης μπορεί να επιλέξει ανάμεσα σε περισσότερες από 500 διαφορετικές λειτουργίες. Η δομή της διαδικασίας γράφεται και αποθηκεύεται σε αρχείο XML. Όμως, ολόκληρος ο αλγόριθμος παριστάνεται στο γραφικό περιβάλλον του προγράμματος σαν ένα δέντρο λειτουργιών, στο οποίο ο χρήστης μπορεί να επέμβει και να διαχειριστεί εύκολα την διαδικασία ανάλυσης.[40]

4.2.5. Λογισμικό CD-HIT

Το **CD-HIT** είναι πρόγραμμα ομαδοποίησης (clustering) των πρωτεϊνικών αλληλουχιών. Αναπτύχθηκε από την ερευνητική ομάδα του Weizhong Li στο Κέντρο Έρευνας Βιολογικών Συστημάτων του πανεπιστημίου της Καλιφόρνια στο Σαν Ντιέγκο (Center for Research in

Biological Systems (CRBS) at University of California San Diego (UCSD)). Η πρώτη έκδοση του CD-HIT δημοσιοποιήθηκε το 2001 και από το 2004 χρησιμοποιείται ως ελεύθερο λογισμικό. Η ελεύθερη χρήση του προγράμματος, σε συνδυασμό με την εξαιρετικά γρήγορή ταχύτητά του, το έχουν καταστήσει αρκετά δημοφιλές και χρησιμοποιείται σε βιβλιοθήκες όπως η UniProt και η PBD. Πλέον το CD-HIT αποτελεί ένα πακέτο πολλών εξειδικευμένων προγραμμάτων.

Η υψηλή ταχύτητα του CD-HIT κατά την ομαδοποίηση των πρωτεϊνών οφείλεται στο ότι δεν πραγματοποιεί συνολική στοίχιση των αλληλουχιών, αλλά χρησιμοποιώντας την μέθοδο k-γραμμάτων αποφεύγει περιττές στοιχίσεις. Αρχικά οι εξεταζόμενες αλληλουχίες διατάσσονται σε σειρά αυξανόμενου μήκους και η μακρύτερη ορίζεται ως η αντιπροσωπευτική αλληλουχία της πρώτης ομάδας. Οι επόμενες αλληλουχίες στοιχίζονται μαζί της σε ζεύγη και αν η ομοιότητα της στοίχισης ξεπερνά ένα ποσοστό κατώφλι π.χ. 90% τότε ομαδοποιούνται με την αλληλουχία-αντιπρόσωπο της πρώτης ομάδας. Αντίθετα, αν το ποσοστό ομοιότητας δεν ξεπερνά το κατώφλι τότε η εξεταζόμενη αλληλουχία ορίζει μια νέα ομάδα. Το CD-HIT χρησιμοποιώντας την μέθοδο των k-γραμμάτων χωρίζει την εξεταζόμενη αλληλουχία σε λέξεις μήκους 2-5. Αποδεικνύεται πως για να ομαδοποιηθούν δυο αλληλουχίες πρέπει να έχουν έναν ελάχιστο αριθμό όμοιων διπεπτιδίων, τριπεπτιδίων, τετραπεπτιδίων ή πεμπτιδίων. Για παράδειγμα, αν το κατώφλι ομοιότητας είναι 85%, δυο αλληλουχίες μήκους λίγο μεγαλύτερο από 100 αμινοξέα πρέπει να έχουν 70 όμοια διπεπτίδια για να ομαδοποιηθούν. Υπολογίζοντας τον αριθμό όμοιων πεπτιδίων, το CD-HIT αποφεύγει την συνολική σύγκριση των αλληλουχιών και αυξάνει την ταχύτητά του. Ο χρήστης έχει τη δυνατότητα επιλογής του k και του κατωφλιού ομοιότητας, ενώ για τον περιορισμό σφαλμάτων οι τιμές του k ,2,3,4 και 5, πρέπει να επιλέγονται για κατώφλια μεγαλύτερα από 40%, 50%, 60% και 70% αντίστοιχα. [77]

ΙΙ. ΤΕΧΝΙΚΟ ΜΕΡΟΣ

1. ΕΙΣΑΓΩΓΗ

Το τεχνικό μέρος της εργασίας πραγματοποιήθηκε σε περιβάλλον Linux (Ubuntu 11.10) καθώς όλα τα λογισμικά και οι βάσεις δεδομένων διατίθενται ελεύθερα για χρήση. Ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η Perl. Η γλώσσα Perl (Practical Report and Extraction Language) εμφανίστηκε για πρώτη φορά το 1987. Δεν απαιτεί μεταγλωττιστή, είναι διερμηνευτική γλώσσα. Δυο πακέτα modules της Perl που χρησιμοποιήθηκαν είναι η BioPerl (έκδοση 1.6.1.) και η DBI (έκδοση 1.618). Η BioPerl περιέχει modules χρήσιμα για βιοπληροφορικές διαδικασίες (ανάγνωση βάσεων δεδομένων, ανάγνωση αποτελεσμάτων ανάλυσης, απόκτηση αλληλουχιών από βάσης δεδομένων κ.α.). Η DBI περιέχει συναρτήσεις και μεθόδους σύνδεσης μέσω της Perl σε βάσεις δεδομένων της MySQL, MSSQL, Oracle, Informix, Sybase, ODBC κ.α. Επιπλέον χρησιμοποιήθηκε το λογισμικό MySQL Workbench (έκδοση 5.2.38) για την κατασκευή βάσεων δεδομένων. Με τον τρόπο αυτό τα δεδομένα και τα αποτελέσματα οργανόνωνταν σε πίνακες και η μεταχείρισή τους γινόταν εύκολη.

Όπως έχει αναφερθεί στην εισαγωγή της εργασίας, το τεχνικό μέρος διακρίνεται σε δυο ενότητες. Αρχικά περιγράφεται ο αλγόριθμος που αναπτύχθηκε για την πρόβλεψη πρωτεϊνικής λειτουργίας με χρήση Οντολογιών Γονιδίων. Στη δεύτερη ενότητα μελετώνται οι συμπεριφορές των αλγόριθμων μηχανικής μάθησης (Perceptron, Μηχανές Διανυσμάτων Υποστήριξης, Μπεϋζιανός Ταξινομητής Πυρήνα, Δέντρα Απόφασης, κ-Κοντινότερος-Γείτονας) για δυο διαφορετικά είδη πληροφοριών εκπαίδευσης (PFAM και PROFEAT χαρακτηριστικών) και οι καλύτεροι αλγόριθμοι χρησιμοποιούνται για το διαχωρισμό θερμοανθετικών ενζύμων από μη θερμοανθεκτικά, ίδιας ενζυμικής κλάσης. Για τα δυο μέρη της δεύτερης ενότητας περιγράφονται παρακάτω οι αντίστοιχοι αλγόριθμοι.

Όλος ο κώδικας Perl βρίσκεται στο παράρτημα (παράρτημα $1^{ης}$ ενότητας, παράρτημα (Α) $2^{ης}$ ενότητας και παράρτημα (Β) $2^{ης}$ ενότητας).

2. ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΟΥ ΠΡΟΒΛΕΨΗΣ ΠΡΩΤΕΪΝΙΚΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΕ ΧΡΗΣΗ ΟΝΤΟΛΟΓΙΩΝ

2.1 Εισαγωγή

Σκοπός της συγκεκριμένης ενότητας του τεχνικού μέρος ήταν η ανάπτυξη ενός αλγορίθμου, ο οποίος θα προβλέπει την πρωτεϊνική λειτουργία άγνωστων πρωτεϊνικών αλληλουχιών με την χρήση όρων Οντολογίας Γονιδίων και θα συσχετίζει την κάθε οντολογία με συγκεκριμένη Ενζυμική κλάση. Η είσοδος του αλγορίθμου θα ήταν ένα αρχείο πρωτεϊνικών αλληλουχιών σε μορφή fasta, ενώ η έξοδος ένα αρχείο σε μορφή csv, το οποίο θα περιείχε τα αποτελέσματα, δηλαδή το όνομα κάθε αλληλουχίας με τους κωδικούς Οντολογίας Γονιδίων και Ενζυμικής Ονοματολογίας που θα είχαν προβλεφτεί. Για την ανάπτυξη του συγκεκριμένου αλγορίθμου δυο ήταν οι βασικές συνιστώσες: 1) Η βάση δεδομένων που θα χρησιμοποιούνταν και 2) το πρόγραμμα ανάλυσης των άγνωστων αλληλουχιών και οι παράμετροί του.

Η βάση δεδομένων έπρεπε να περιέχει γνωστές πρωτεΐνες χαρακτηρισμένες από όρους Οντολογίας Γονιδίων. Για τον σκοπό αυτό χρησιμοποιήθηκε η Gene Ontology Annotation Database και συγκεκριμένα το αρχείο κειμένου gene_association.goa_uniprot_noiea (έκδοση 4/12/2012). Το συγκεκριμένο αρχείο περιέχει 205632 εγγραφές (entries) από 61098 ξεχωριστές πρωτεΐνες διαφόρων οργανισμών, οι οποίες δεν έχουν υποστεί ηλεκτρονική επεξεργασία (no inferred from electronic annotation). Όπως έχει αναφερθεί στο κεφάλαιο 4 του θεωρητικού μέρους, η Gene Ontology Annotation Database περιέχει γνωστές αλληλουχίες από την βιβλιοθήκη της UniProt. Η συγκεκριμένη έκδοση αποτελούνταν από 16308 αλληλουχίες της UniProt/SwissProt και 44790 αλληλουχίες της UniProt/TrEMBL. Για πρόγραμμα ανάλυσης των άγνωστων αλληλουχιών επιλέχθηκε το blastp (αυτόνομη έκδοση ncbi-blast-2.2.26+).

2.2. Προετοιμασία - Κατασκευή Κατάλληλης Βάσης Δεδομένων

Πριν την κατασκευή του αλγόριθμου, δημιουργήθηκε μια βάση δεδομένων στο MySQL Workbench με το όνομα “GO_db”, η οποία περιείχε δυο πίνακες: τον πίνακα “annotations”, στον οποίο φορτώθηκαν τα δεδομένα του αρχείου gene_association.goa_uniprot_noiea και ο πίνακας “ec2go”, ο οποίος περιείχε όλες τις συσχετίσεις όρων Οντολογίας Γονιδίων με κωδικούς Ενζυμικής Ονοματολογίας. Επιπλέον, δημιουργήθηκε μια βάση δεδομένων των αλληλουχιών του αρχείου gene_association.goa_ uniprot_noiea, σε μορφή fasta, ώστε να χρησιμοποιηθεί από το blastp. Η παραπάνω διαδικασία περιγράφεται αναλυτικά σε έξι στάδια:

ΣΤΑΔΙΟ 1^{ο}: Δημιουργία βάσης δεδομένων με το όνομα “GO_db” στο MySQL Workbench με χρήση του αρχείου Perl **create_MySQL_schema.pl**.

ΣΤΑΔΙΟ 2^{ο}: Απόκτηση του αρχείου gene_association.goa_ uniprot_noiea και του αρχείου κειμένου ec2go (έκδοση 1.487), το οποίο περιέχει 4158 εγγραφές (3999 κωδικούς Ενζυμικής Ονοματολογίας και 4126 όρους Οντολογίας Γονιδίων.

ΣΤΑΔΙΟ 3^{ο}: Δημιουργία πίνακα με όνομα “annotations” στην βάση “GO_db”, με χρήση του αρχείου Perl **annotations_parser.pl**. Το αρχείο δημιουργεί έναν πίνακα με 17 στήλες με το όνομα “annotations” στην βάση δεδομένων που επιλέγει ο χρήστης (στην ενότητα αυτή το όνομα της βάσης ήταν “GO_db”). Καθεμία από τις στήλες αντιστοιχεί σε ένα χαρακτηριστικό των εγγραφών του gene_association.goa_uniprot_noiea: db_name, object_acc, object_symbol, qualifier, GO_id, db_reference, evidence, additional_id, GO_aspect, object_name, synonym, object_type, taxon_id, date, assigned_by, annotation_extension, gene_product_form_id. Στη συνέχεια λαμβάνει από το αρχείο gene_association.goa_uniprot_noiea τα χαρακτηριστικά κάθε εγγραφής και τα φορτώνει στον πίνακα στην στήλη όπου αντιστοιχεί το καθένα (εικόνα 2.1-2.2). Με τον τρόπο αυτό οργανώνονται τα χαρακτηριστικά κάθε εγγραφής σε έναν πίνακα βάσης δεδομένων και καθίσταται εύκολη η ανάγνωση και η μετέπειτα χρήση τους.

#	ind	db_name	object_acc	object_symbo	qualifier	GO_id	db_reference	evidence	additional_id	GO_aspect
1	1	UniProtKB	A0A098	GSM1		GO:0005515	PMID:18510927	IPI	UniProtKB:Q9XEK5	F
2	2	UniProtKB	A0A8M2	lsm14a-a		GO:0003723	PMID:17074753	IDA		F
3	3	UniProtKB	A0A8M2	lsm14a-a		GO:0005737	PMID:17074753	IDA		C
4	4	UniProtKB	A0A8M2	lsm14a-a		GO:0017148	PMID:17074753	IMP		P
5	5	UniProtKB	A0A8M2	lsm14a-a		GO:0017151	PMID:17074753	IPI	UniProtKB:P54824	F
6	6	UniProtKB	A0A8M2	lsm14a-a		GO:0030529	PMID:17074753	IPI	UniProtKB:P21574	C
7	7	UniProtKB	A0A8M2	lsm14a-a		GO:0043234	PMID:17074753	IPI	UniProtKB:Q8AV13	C
8	8	UniProtKB	A0AAR7	CCAMK		GO:0004683	PMID:16810257	IDA		F
9	9	UniProtKB	A0AAR7	CCAMK		GO:0009608	PMID:16810257	IDA		P
10	10	UniProtKB	A0AAR7	CCAMK		GO:0009931	PMID:16810257	IDA		F
11	11	UniProtKB	A0AAR7	CCAMK		GO:0046777	PMID:16810257	IDA		P
12	12	UniProtKB	A0AQ71	bai		GO:0009953	GO_REF:000002	ISS	UniProtKB:Q8SXY6	P
13	13	UniProtKB	A0ASJ9	A0ASJ9		GO:0001664	GO_REF:000003	ISS	PANTHER:PTN0000	F
14	14	UniProtKB	A0ASJ9	A0ASJ9		GO:0003924	GO_REF:000003	ISS	PANTHER:PTN0000	F
15	15	UniProtKB	A0ASJ9	A0ASJ9		GO:0004871	GO_REF:000003	ISS	PANTHER:PTN0000	F
16	16	UniProtKB	A0ASJ9	A0ASJ9		GO:0005834	GO_REF:000003	ISS	PANTHER:PTN0000	C

Εικόνα 2.1: Πίνακας "annotations".

object_name	synonym	object_type	taxon_id	date	assigned_by	annotation_extension	gene_product_form_id
Gamete-specific protein minus 1	A0A098_CHLRE|GSM1	protein	3055	2012-11-26	IntAct	occurs_in(GO:0005634)	
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-15	UniProtKB		
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-15	UniProtKB		
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-15	UniProtKB		
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-15	UniProtKB		
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-15	UniProtKB		
Protein LSM14 homolog A-A	L14AA_XENLA|lsm14a-a|lsm14a|rap55a-a	protein	8355	2009-12-14	UniProtKB		
Calcium and calcium/calmodulin-depen	CCAMK_LOTJA|CCAMK	protein	34305	2010-09-23	UniProtKB		
Calcium and calcium/calmodulin-depen	CCAMK_LOTJA|CCAMK	protein	34305	2010-09-23	UniProtKB		
Calcium and calcium/calmodulin-depen	CCAMK_LOTJA|CCAMK	protein	34305	2010-09-23	UniProtKB		
Calcium and calcium/calmodulin-depen	CCAMK_LOTJA|CCAMK	protein	34305	2010-09-23	UniProtKB		
Transmembrane emp24 domain-contair	TMEDA_DROSI|GD21160|bai	protein	7240	2010-03-29	UniProtKB		
G protein alpha subunit AgGq6	A0ASJ9_ANOGA	protein	7165	2011-01-25	RefGenome		
G protein alpha subunit AgGq6	A0ASJ9_ANOGA	protein	7165	2011-01-25	RefGenome		
G protein alpha subunit AgGq6	A0ASJ9_ANOGA	protein	7165	2011-01-25	RefGenome		
G protein alpha subunit AgGq6	A0ASJ9_ANOGA	protein	7165	2011-01-25	RefGenome		
G protein alpha subunit AgGq6	A0ASJ9_ANOGA	protein	7165	2012-11-24	GOC		

Εικόνα 2.2: Πίνακας "annotations", συνέχεια.

ΣΤΑΔΙΟ 4ο: Χρήση του αρχείου Perl **ec2go.pl** με σκοπό την δημιουργία ενός πίνακα "ec2go" στην βάση δεδομένων "GO_db", ο οποίος θα περιέχει τις αντιστοιχίες των Οντολογιών Γονιδίων με κωδικούς Ενζυμικής Ονοματολογίας. Το συγκεκριμένο αρχείο δημιουργεί τον πίνακα "ec2go", ο οποίος αποτελείται από δυο στήλες ("EC_number", "GO_id"). Από το αρχείο κειμένου ec2go λαμβάνει κάθε κωδικό Ενζυμικής Ονοματολογίας (EC_number) και τον αντίστοιχο κωδικό Οντολογίας Γονιδίων (GO_id) και τους φορτώνει στον πίνακα ec2go.

ΣΤΑΔΙΟ 5ο: Χρήση του αρχείου Perl **distinct_acc_parser.pl**. με σκοπό την καταγραφή σε αρχείο κειμένου, με το όνομα "acc_numbers", των κωδικών της UniProt που περιέχονται στην στήλη "object_acc" του πίνακα "annotations". Οι κωδικοί αυτοί είναι τα ονόματα των αλληλουχιών. Η επιλογή τους γίνεται χωρίς επαναλήψεις, δηλαδή από τις 205632 εγγραφές (entries) επιλέγονται 61098 κωδικοί "object_acc".

ΣΤΑΔΙΟ 6ο: Απόκτηση των αλληλουχιών που αντιστοιχούν στους κωδικούς του αρχείου “acc_numbers” από την UniProt. Το αρχείο fasta που απέστειλε η UniProt ονομάστηκε “GO_db_sequences” και μετατράπηκε σε βάση δεδομένων για στοίχιση αλληλουχιών, εκτελώντας στην γραμμή εντολών (command line) την εντολή:

```
>formatdb –i /path/GO_db_sequences –p T
```

2.3. Αλγόριθμος Πρόβλεψης Λειτουργίας Άγνωστων Αλληλουχιών

Ο αλγόριθμος πρόβλεψης πρωτεϊνικής λειτουργίας που αναπτύχθηκε, περιέχεται στο αρχείο Perl **blastp_for_GO.pl**. Οι παράμετροι του blastp εξαρτώνται από το μήκος της αλληλουχίας που εξετάζεται. Για τον λόγο αυτό κρίθηκε σκόπιμος ο διαχωρισμός των εξεταζόμενων αλληλουχιών σε τρεις κατηγορίες ανάλογα με το μήκος τους: 1) μήκος <85 αμινοξέα, 2) μήκος >85 αμινοξέα και <300 και 3) μήκος >300 αμινοξέα. Ο διαχωρισμός αυτός πραγματοποιείται με χρήση του αρχείου Perl **filter_length.pl**. Ο αλγόριθμος του filter_length.pl δέχεται ως είσοδο το αρχείο fasta των εξεταζόμενων αλληλουχιών, μετρά το μήκος τους και δημιουργεί τρία νέα αρχεία fasta, όπου τυπώνει την κάθε αλληλουχία ανάλογα με το μήκος της.

Τα τρία αρχεία αλληλουχιών που προκύπτουν χρησιμοποιούνται ως μεταβλητές εισόδου στον αλγόριθμο του blastp_for_GO.pl. Συνολικά οι μεταβλητές εισόδου του blastp_for_GO.pl είναι οκτώ και ο αλγόριθμος χωρίζεται σε τέσσερα στάδια.

Μεταβλητές εισόδου:

$file_in : Το όνομα του αρχείου που περιέχει τις προς ανάλυση πρωτεϊνικές αλληλουχίες (σε μορφή fasta)

$file_out : Το όνομα του αρχείου, όπου θα τυπωθούν τα αποτελέσματα του blastp και το όνομα του πίνακα στο MySQL Workbench, όπου θα φορτωθούν τα αποτελέσματα

$blastp_db : η βάση δεδομένων των γνωστών πρωτεϊνικών αλληλουχιών που θα χρησιμοποιήσει το blastp

$evalue : παράμετρος του blastp: η ελάχιστη τιμή e-value που πρέπει να έχει μια στοίχιση για να τυπωθεί στα αποτελέσματα

$matrix : παράμετρος του blastp: ο πίνακας αντικατάστασης που θα χρησιμοποιηθεί για την αξιολόγηση των στοιχίσεων

$db : το όνομα της βάσης δεδομένων όπου θα δημιουργηθεί ο πίνακας "$file_out" και που περιέχεται ο πίνακας "ec2go"

$user_name : όνομα χρήστη του MySQL Workbench

$password : κωδικός χρήστη του MySQL Workbench

Η εντολή στην γραμμή εντολών (command line) για την ανάλυση ενός αρχείου "seqsuences.fasta" έχει την εξής μορφή:

```
> perl blastp_for_GO.pl /path/sequences.fasta blastp_out
/path/GO_db_sequences.fasta e-value sub_matrix GO_db
MySQL_username MySQL_password
```

ΣΤΑΔΙΟ 1^{ο}: Αρχικά δημιουργείται ένα νέο αρχείο κειμένου με το όνομα "process_for_$file_in", στο οποίο θα καταγραφούν πληροφορίες σχετικές με την εκτέλεση του αλγορίθμου. Στη συνέχεια εκτελείται το πρόγραμμα blastp με είσοδο το αρχείο $file_in, παραμέτρους τις μεταβλητές $blastp_db, $evalue, $matrix και έξοδο το αρχείο $file_out. Μόλις ολοκληρωθεί η στοίχιση των αλληλουχιών, τυπώνεται φράση, που ενημερώνει για τον τερματισμό, στο αρχείο "process_for_$file_in".

ΣΤΑΔΙΟ 2^{ο}: Το αρχείο εξόδου $file_out περιέχει τα αποτελέσματα των στοιχίσεων σε μορφή "blast". Δημιουργείται ένας πίνακας με όνομα "$file_out" στην βάση "$db". Ο πίνακας "$file_out" αποτελείται από πέντε στήλες (query_name, hit_name, hsp_legth, hsp_value, hsp_percent_identity), στις οποίες θα φορτωθούν τα αντίστοιχα αποτελέσματα του blastp. Με χρήση του module Bio::SearchIO της BioPerl η ανάγνωση των αποτελεσμάτων γίνεται εύκολη. Για κάθε εγγραφή, ορίζονται πέντε μεταβλητές ($query_name, $hit_name, $hsp_length, $hsp_evalue, $hsp_percent_identity) στις οποίες φορτώνονται το όνομα της εξεταζόμενης πρωτεΐνης (query_name), το όνομα της γνωστής πρωτεΐνης (hit_name), το μήκος (length), το e-value και το ποσοστό ομοιότητας της περιοχής στοίχισης (percent_identity). Στη συνέχεια οι τιμές των πέντε μεταβλητών φορτώνονται στον πίνακα "$file_out" στις αντίστοιχες στήλες. Τέλος, τυπώνεται μια πρόταση στο αρχείο

“process_for_$file_in”, η οποία ενημερώνει για τον τερματισμό της φόρτωσης των αποτελεσμάτων.

#	ind	query_name	hit_name	hsp_length	hsp_evalue	hsp_percent_identity
1	1	makis_c1_5034_length=82	tr|Q0C0P0|Q0C0P0_HYPNA	51	5e-06	47.0588
2	2	makis_c2_2548_length=76	tr|G4N3U5|G4N3U5_MAGO7	70	5e-08	40
3	3	makis_c2_2548_length=76	tr|G4MXF4|G4MXF4_MAGO7	63	3e-07	36.5079
4	4	makis_c2_2548_length=76	tr|G4NLE3|G4NLE3_MAGO7	68	4e-07	41.1765
5	5	makis_c2_2548_length=76	sp|A4RQ29|LSM6_MAGO7	64	5e-07	35.9375
6	6	makis_c2_2548_length=76	tr|G4MRX5|G4MRX5_MAGO7	66	8e-06	31.8182
7	7	makis_c3_5308_length=72	tr|G4N6C5|G4N6C5_MAGO7	63	9e-06	39.6825
8	8	makis_c3_18656_length=85	tr|G4N1Q3|G4N1Q3_MAGO7	51	2e-11	47.0588
9	9	makis_c3_18656_length=85	tr|Q8ILS0|Q8ILS0_PLAF7	51	9e-10	49.0196
10	10	makis_c8_4443_length=50	tr|Q5A299|Q5A299_CANAL	50	5e-13	72
11	11	makis_c8_4443_length=50	tr|Q5AGG1|Q5AGG1_CANAL	42	6e-06	57.1429
12	12	makis_c10_3023_length=68	sp|Q6QN04|RPAB5_CHILA	63	4e-19	53.9683
13	13	makis_c10_3023_length=68	tr|G4MXI6|G4MXI6_MAGO7	63	2e-18	53.9683
14	14	makis_c10_3023_length=68	tr|Q8IC08|Q8IC08_PLAF7	63	2e-12	41.2698
15	15	makis_c23_2239_length=66	tr|Q8IEN2|Q8IEN2_PLAF7	56	4e-07	48.2143
16	16	makis_c28_2234_length=52	sp|Q9KU47|SYI_VIBCH	34	2e-07	52.9412
17	17	makis_c28_2246_length=51	sp|Q9KU47|SYI_VIBCH	42	5e-09	57.1429
18	18	makis_c28_2246_length=51	tr|G4MWJ2|G4MWJ2_MAGO7	29	2e-06	55.1724
19	19	makis_c32_677_length=84	tr|Q81UT1|Q81UT1_BACAN	74	1e-11	37.8378
20	20	makis_c32_677_length=84	tr|Q81UT3|Q81UT3_BACAN	74	2e-10	33.7838

Εικόνα 2.3: Πίνακας αποτελεσμάτων blastp.

ΣΤΑΔΙΟ 3^{ο}: Πλέον τα αποτελέσματα του blastp υπάρχουν καταγεγραμμένα στο MySQL Workbench και μπορούν να χρησιμοποιηθούν με ευκολία. Το επόμενο βήμα είναι η συσχέτιση των αποτελεσμάτων με όρους Οντολογίας Γονιδίων. Πριν την εύρεση των όρων δημιουργείται ένας νέος πίνακας με όνομα “GO_$file_out”, ο οποίος έχει την δομή του πίνακα “$file_out” με μια επιπλέον στήλη, την “GO_id”. Υπενθυμίζεται πως οι όροι Οντολογίας Γονιδίων έχουν καταγραφεί στον πίνακα “annotations”. Επομένως πρέπει να βρεθεί μια κοινή μεταβλητή των πινάκων “annotations” και “$file_out“ ώστε να συσχετιστούν οι δυο πίνακες. Η μεταβλητή αυτή είναι το όνομα της γνωστής πρωτεΐνης για κάθε στοίχιση, το οποίο στον πίνακα “annotations” βρίσκεται στην στήλη “object_acc” και στον “$file_out“ στην στήλη “hit_name”. Για κάθε εγγραφή του πίνακα “$file_out“ επιλέγονται οι καταγραφές όλων των στηλών, ενώ για τον πίνακα “annotations”, επιλέγεται η καταγραφή της στήλης “GO_id”, όπου το περιεχόμενο της “object_acc” υπάρχει στην “hit_name”. Τα επιλεγμένα στοιχεία φορτώνονται στον πίνακα “GO_$file_out”, στις αντίστοιχες στήλες.

ΣΤΑΔΙΟ 4ο: Ο πίνακας “GO_$file_out” περιέχει τα ζητούμενα αποτελέσματα εκτός από τους κωδικούς Ενζυμικής Ονοματολογίας. Για την δημιουργία του τελικού αρχείο αποτελεσμάτων "final_results_$file_out.csv" χρησιμοποιείται ένα πρόχειρο αρχείο και ο πίνακας ec2go. Το τελικό αρχείο περιέχει οκτώ στήλες για την κάθε εγγραφή: "query_name, hit_name, hsp_length, hsp_evalue, hsp_percent_identity, GO_id, EC_number" (εικόνα 2.4). Στο τέλος της διαδικασίας διαγράφονται ορισμένα αρχεία κειμένου και πίνακες που δεν χρειάζονται ενώ τυπώνεται και αντίστοιχη φράση στο αρχείο “process_for_$file_in”.

Λόγω του μεγάλου όγκου πληροφορίας που περιέχεται στα αρχεία "final_results_$file_out.csv", δημιουργήθηκε ένα επιπλέον αρχείο Perl, το **calculate_percentage.pl**. Το πρόγραμμα υπολογίζει τον αριθμό και το ποσοστό των αποτελεσμάτων συγκεκριμένου GO_id ή EC_Number.

	A	B	C	D	E	F	G
1	query_name	hit_name	hsp_length	hsp_evalue	hsp_percent_identity	GO_id	EC_number
2							
3	makis_c1_4913_length=263	sp|A0QPF9|RMLA_MYCS2	243	2e-14	27.9835	GO:0000287	no_EC_number
4	makis_c1_4913_length=263	sp|A0QPF9|RMLA_MYCS2	243	2e-14	27.9835	GO:0008879	2.7.7.24
5	makis_c1_4913_length=263	sp|A0QPF9|RMLA_MYCS2	243	2e-14	27.9835	GO:0019300	no_EC_number
6	makis_c1_4913_length=263	sp|A0QPF9|RMLA_MYCS2	243	2e-14	27.9835	GO:0045226	no_EC_number
7	makis_c1_9565_length=294	sp|A2RI01|ECFA1_LACLM	187	1e-14	27.2727	GO:0005215	no_EC_number
8	makis_c1_9565_length=294	sp|A2RI01|ECFA1_LACLM	187	1e-14	27.2727	GO:0005886	no_EC_number
9	makis_c1_9565_length=294	sp|A2RI02|ECFA2_LACLM	209	5e-15	28.7081	GO:0005215	no_EC_number
10	makis_c1_9565_length=294	sp|A2RI02|ECFA2_LACLM	209	5e-15	28.7081	GO:0005886	no_EC_number
11	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	217	6e-09	24.424	GO:0005886	no_EC_number
12	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	200	1e-06	24.5	GO:0005886	no_EC_number
13	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	217	6e-09	24.424	GO:0016021	no_EC_number
14	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	200	1e-06	24.5	GO:0016021	no_EC_number
15	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	217	6e-09	24.424	GO:0042626	no_EC_number
16	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	200	1e-06	24.5	GO:0042626	no_EC_number
17	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	217	6e-09	24.424	GO:0055085	no_EC_number
18	makis_c1_9565_length=294	tr|A4VD54|A4VD54_TETTS	200	1e-06	24.5	GO:0055085	no_EC_number
19	makis_c1_9565_length=294	tr|A4VE58|A4VE58_TETTS	196	5e-08	25.5102	GO:0005886	no_EC_number
20	makis_c1_9565_length=294	tr|A4VE58|A4VE58_TETTS	196	5e-08	25.5102	GO:0016021	no_EC_number
21	makis_c1_9565_length=294	tr|A4VE58|A4VE58_TETTS	196	5e-08	25.5102	GO:0042626	no_EC_number
22	makis_c1_9565_length=294	tr|A4VE58|A4VE58_TETTS	196	5e-08	25.5102	GO:0055085	no_EC_number
23	makis_c1_13264_length=90	sp|A4WVF5|MCAL_RHOS5	40	3e-06	52.5	GO:0050083	4.1.3.24
24	makis_c1_9565_length=294	tr|A6VDB6|A6VDB6_PSEA7	211	5e-08	25.5924	GO:0016021	no_EC_number
25	makis_c1_9565_length=294	tr|A6VDB6|A6VDB6_PSEA7	211	5e-08	25.5924	GO:0034040	no_EC_number
26	makis_c1_9565_length=294	tr|A6VDB6|A6VDB6_PSEA7	211	5e-08	25.5924	GO:0055085	no_EC_number
27	makis_c1_9565_length=294	tr|A8II54|A8II54_CHLRE	225	2e-07	27.1111	GO:0005774	no_EC_number
28	makis_c1_9565_length=294	tr|A8II54|A8II54_CHLRE	225	2e-07	27.1111	GO:0016021	no_EC_number

Εικόνα 2.4: Αρχείο "final_results_$file_out.csv".

3. ΜΕΛΕΤΗ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

3.1 Εισαγωγή

Ο σκοπός της δεύτερης ενότητας του τεχνικού μέρος ήταν διπλός (μελέτη και αξιοποίηση). Οι αλγόριθμοι της μηχανικής μάθησης, όπως έχει αναφερθεί, χρησιμοποιούνται όλο και περισσότερο για την ανάλυση βιολογικών αλληλουχιών. Για τον λόγο αυτό, θεωρήθηκε σκόπιμη η μελέτη της απόδοσής τους στο διαχωρισμό συγκεκριμένων ενζυμικών κλάσεων, σε διωνιμικές (binominal) πληροφορίες εκπαίδευσης. Στη συνέχεια, με βάση τα αποτελέσματα της μελέτης, χρησιμοποιήθηκαν συγκεκριμένοι αλγόριθμοι για την εξαγωγή χρήσιμων πληροφοριών για ένζυμα βιοτεχνολογικού ενδιαφέροντος.

Οι ενζυμικές κλάσεις που επιλέχτηκαν ώστε να μελετηθεί η συμπεριφορά των αλγορίθμων μηχανικής μάθησης είναι οι 3.1.-.-, 3.2.-.-, 3.4.-.-, 3.5.-.-, 3.6.-.-.. Οι κλάσεις αυτές αποτελούν υποκατηγορίες των υδρολασών και με βάση τους κανόνες Ενζυμικής Ονοματολογίας χαρακτηρίζονται ως μέτρια εξειδικευμένες. Ο διωνιμικός χαρακτήρας των πληροφοριών εκπαίδευσης σημαίνει πως κάθε εξεταζόμενο σύνολο αλληλουχιών αποτελείται από δυο κλάσεις, μια υποκατηγορία υδρολασών (π.χ. 3.1.-.-) και αντίθετα παραδείγματα (όχι 3.1.-.-), τα οποία δεν ανήκουν γενικά στις υδρολάσες. Επομένως οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιήθηκαν για τον διαχωρισμό μιας υποκατηγορίας υδρολασών (θετικά πρότυπα πληροφορίας εκπαίδευσης) από μη υδρολάσες (αρνητικά παραδείγματα πληροφορίας εκπαίδευσης).

Ο διαχωρισμός των πρότυπων παραδειγμάτων, όπως έχει ειπωθεί στο Κεφάλαιο 3 του Θεωρητικού μέρους, γίνεται με βάση τις τιμές τους σε συγκεκριμένα χαρακτηριστικά. Το πρώτο λοιπόν ερώτημα στη συγκεκριμένη ενότητα ήταν τί χαρακτηριστικά θα χρησιμοποιηθούν. Τα χαρακτηριστικά έπρεπε να καταδεικνύουν φυσικοχημικές, δομικές ή λειτουργικές ιδιότητες των αλληλουχιών. Για τον σκοπό αυτό χρησιμοποιήθηκαν δυο είδη

χαρακτηριστικών. Η πρώτη κατηγορία ήταν οι κωδικοί PFAM που προέκυπταν για κάθε αλληλουχία, μετά από ανάλυσή τους με το πρόγραμμα HMMER3 και τη βάση δεδομένων πρωτεϊνικών οικογενειών Pfam-A. Τα χαρακτηριστικά αφορούν την ομοιότητα των αλληλουχιών με πρωτεϊνικές οικογένειες. Η δεύτερη κατηγορία χαρακτηριστικών ήταν οι κωδικοί του προγράμματος PROFEAT. Το συγκεκριμένο πρόγραμμα αναλύει τις φυσικοχημικές ιδιότητες των αμινοξέων και κατ' επέκταση της αλληλουχίας. Επομένως, οι αλγόριθμοι μηχανικής μάθησης μελετήθηκαν για δυο διαφορετικά είδη χαρακτηριστικών (δομικών και φυσικοχημικών ιδιοτήτων).

Το δεύτερο μέρος της συγκεκριμένης ενότητας αφορά στην αξιοποίηση των αποτελεσμάτων από τη μελέτη των αλγόριθμων και την χρήση τους για το διαχωρισμό ενζύμων μεσόφιλων, θερμόφιλων και υπερθερμόφιλων μικροοργανισμών. Τα θερμοανθεκτικά και υπερθερμοανθεκτικά ένζυμα, σύμφωνα και με το πρόγραμμα HotZyme, έχουν ιδιαίτερο βιοτεχνολογικό ενδιαφέρον λόγω της αντοχής τους σε υψηλές και ακραίες θερμοκρασίες $\left(50-100+\ {}^{o}\mathrm{C}\right)$. Η αντοχή των ενζύμων σε υψηλές θερμοκρασίες σχετίζεται με τις φυσικοχημικές τους ιδιότητες αλλά και με δομικά χαρακτηριστικά. Για το λόγο αυτό χρησιμοποιήθηκαν και οι δυο κατηγορίες χαρακτηριστικών (PFAM, PROFEAT). Οι πληροφορίες εκπαίδευσης είχαν πάλι διωνυμική μορφή, τα θετικά παραδείγματα ήταν ένζυμα μιας συγκεκριμένης υποκατηγορίας υδρολασών (π.χ. 3.4.-.-) που ανήκουν σε θερμόφιλους ή υπερθερμόφιλους μικροοργανισμούς και τα αρνητικά παραδείγματα ήταν ένζυμα αντίστοιχης κατηγορίας που ανήκουν σε μεσόφιλους μικροοργανισμούς. Η κοινή ενζυμική κατηγορία επιλέχτηκε γιατί τα δυο είδη παραδειγμάτων έπρεπε να διαφέρουν μόνο ως προς τα χαρακτηριστικά που καθορίζουν την αντοχή τους ή μη σε υψηλές θερμοκρασίες.

Για τα δυο μέρη αυτής της ενότητας αναπτύχθηκαν οι ανάλογοι αλγόριθμοι, οι οποίοι παρουσιάζονται στη συνέχεια.

3.2. Μελέτη Αλγόριθμων Μηχανικής Μάθησης

3.2.1. Προετοιμασία - Απόκτηση Πρωτεϊνικών Αλληλουχιών

Το πρώτο στάδιο της μελέτης ήταν η απόκτηση την κατάλληλης πληροφορίας, δηλαδή των πρωτεϊνικών αλληλουχιών. Χρησιμοποιήθηκε η βιβλιοθήκη UniProt (έκδοση 3/10/2012). Πιο συγκεκριμένα, για κάθε κλάση των υδρολασών (3.1.-.-, 3.2.-.-, 3.4.-.-, 3.5.-.-, 3.6.-.-.) επιλέχθηκαν όλες οι αλληλουχίες που ανήκαν σε αρχαία (archaea) (συγκεκριμένη κατηγορία μικροοργανισμών), ήταν χειροκίνητα χαρακτηρισμένες, δηλαδή ανήκαν στην UniProt/SwissProt, και το μήκος τους ήταν μεγαλύτερο από 50 αμινοξέα, καθώς αλληλουχίες μικρότερου μήκους δημιουργούσαν πρόβλημα στο πρόγραμμα PROFEAT και δεν είχαν ιδιαίτερο ενδιαφέρον (χαρακτηρίζονται ως τμήματα (fragments) πρωτεϊνών και όχι ολόκληρες πολυπεπτιδικές αλυσίδες). Επιπλέον στις επιλεγμένες αλληλουχίες πραγματοποιούνταν ομαδοποίηση με την βάση UniRef50, ώστε να μειωθεί η περιττή πληροφορία. Οι ομάδες αλληλουχιών αποκτιόνταν σε αρχείο fasta. Τέλος, με χρήση του προγράμματος cd-hit οι αλληλουχίες ομαδοποιούνταν ξανά με κατώφλι ομοιότητας 70%, ώστε να μειωθεί ακόμα περισσότερο η περιττή πληροφορία. Η διπλή ομαδοποίηση είχε σαν σκοπό την αποφυγή ομάδων με πολύ κοινές αλληλουχίες, γεγονός που θα αύξανε την περιττή πληροφορία κατά την μηχανική μάθηση. Η ίδια διαδικασία πραγματοποιήθηκε και για όλες τις αλληλουχίες αρχαίων, οι οποίες είχαν κωδικό Ενζυμικής Ονοματολογίας αλλά δεν ανήκαν στις υδρολάσες.

Στο σημείο αυτό δημιουργήθηκε μια βάση δεδομένων στο MySQL Workbench με όνομα “ML_archaea_db”, με χρήση του αρχείου Perl **create_MySQL_schema.pl**. Επιπλέον, με το αρχείο Perl **parser_non.pl** φορτώθηκαν όλες οι αλληλουχίες των μη υδρολασών σε πίνακα με όνομα “non_seqs” στην “ML_archaea_db”. Από τον συγκεκριμένο πίνακα επιλέγονταν στη συνέχεια με τυχαίο τρόπο τα αρνητικά παραδείγματα κάθε πληροφορίας εκπαίδευσης.

Πριν το επόμενο στάδιο, όπως και στην πρώτη ενότητα του πειραματικού μέρους, δημιουργήθηκαν βάσεις δεδομένων στο MySQL

Workbench. Στις βάσεις αυτές φορτώνονταν και επεξεργάζονταν δεδομένα, ώστε να προκύψουν τα τελικά αρχεία πληροφοριών εκπαίδευσης. Εδώ πρέπει να σημειωθεί πως το μέγεθος της πληροφορίας εκπαίδευσης (δηλαδή ο αριθμός των θετικών και αρνητικών πρότυπων παραδειγμάτων) είναι πολύ σημαντική παράμετρος για τους αλγόριθμους μηχανικής μάθησης. Για κάθε κατηγορία υδρολασών δημιουργήθηκαν πέντε διαφορετικές πληροφορίες εκπαίδευσης, οι οποίες διέφεραν ως προς των αριθμών των μη υδρολασών. Σκοπός ήταν να διαπιστωθεί πώς επηρεάζεται η απόδοση των αλγόριθμων ανάλογα με το μέγεθος της πληροφορίας. Πιο αναλυτικά, αν ο αριθμός των υδρολασών ήταν N, οι πληροφορίες εκπαίδευσης είχαν αριθμό μη υδρολασών ίσο με $N\times0.5$, $N\times1$, $N\times2$, $N\times3$, $N\times4$, αντίστοιχα. Οι αλληλουχίες των μη υδρολασών, όπως αναφέρθηκε παραπάνω, περιέχονταν στον πίνακα “non_seqs”. Με το αρχείο Perl **select_random.pl** επιλέγονταν κάθε φορά ένας συγκεκριμένος αριθμός μη υδρολασών με τυχαίο τρόπο, οι οποίες αποτελούσαν τα αρνητικά παραδείγματα της πληροφορίας. Τέλος, για κάθε συντελεστή (factor: 0.5, 1, 2, 3, 4), δημιουργήθηκε μια ξεχωριστή βάση δεδομένων στο MySQL Workbench με όνομα “ML_archaea_xfactor”.

3.2.2. Κατασκευή Πληροφορίας Εκπαίδευσης PFAM Χαρακτηριστικών

Το επόμενο στάδιο ήταν η ανάλυση των αλληλουχιών με τα προγράμματα HMMER3 και PROFEAT. Αρχικά κατασκευάστηκε ο αλγόριθμος δημιουργίας πληροφορίας εκπαίδευσης με χαρακτηριστικά PFAM κωδικούς. Ο αλγόριθμος περιγράφεται στο αρχείο Perl **hmmer_process.pl**, το οποίο χρησιμοποιεί υπορουτίνες (subroutines) του πακέτου **Hmmer_Process.pm**.

Περιγραφή του αρχείου Perl hmmer_process.pl: Πριν την εκτέλεση του αλγορίθμου τα αρχεία των αλληλουχιών που επρόκειτο να αναλυθούν συγκεντρώνονται στον ίδιο κατάλογο, όπου θα εκτελεστεί το πρόγραμμα hmmer_process.pl. Οι μεταβλητές εισόδου είναι οι εξής:

$db : το όνομα της βάσης δεδομένων, “ML_archaea_xfactor ”

$user_name : όνομα χρήστη του MySQL Workbench

$password : κωδικός χρήστη του MySQL Workbench

$table : το όνομα του πίνακα, όπου θα φορτωθούν τα αποτελέσματα του HMMER (π.χ. hyd_3_1, hyd_3_2)

$hmmer_db : η βάση δεδομένων που χρησιμοποιεί ο HMMER, Pfam-A database

$N : ο αριθμός των αρχείων αλληλουχιών που θα αναλυθούν. Ο χρήστης έχει την δυνατότητα να αναλύσει κάθε φορά διαφορετικό αριθμό αρχείων, ανάλογα με την δομή που θέλει να έχει η πληροφορία εκπαίδευσης (διωνυμική, πολυωνυμική). Οι αλληλουχίες που θα αναλυθούν, θα αποτελέσουν τα πρότυπα της πληροφορίας εκπαίδευσης. Τα αρχεία εισέρχονται στο αλγόριθμο με την εξής ρουτίνα:

```
for ($i=6; $i <2*$N+6; $i =$i +2) {
        my $file_in = $ARGV[$i]; # όνομα αρχείου αλληλουχιών
        my $class = $ARGV[$i+1]; # κλάση των αλληλουχιών (π.χ. 3.1.-.-, non_3.1.-.-)
        $hash{$file_in} = $class;
}
```

Ο χρήστης δηλώνει πρώτα το αρχείο των αλληλουχιών, που επρόκειτο να αναλυθούν, και μετά την κλάση τους. Οι μεταβλητές «αρχείο» και «κλάση» εισέρχονται σε έναν σύνθετο πίνακα (hash table), όπου η κλάση είναι μια μεταβλητή και το αντίστοιχο αρχείο είναι το κλειδί της μεταβλητής. Οι μεταβλητές εισόδου είναι συνολικά $5+2\times$$N.

Η εντολή στην γραμμή εντολών (command line) για δυο αρχεία αλληλουχιών και θετική κλάση την 3.6.-.-, έχει την εξής μορφή:

```
>hmmer_process.pl ML_archaea_xfactor MySQL_username
MySQL_password hyd_3_6 /home/thodoris/PROGRAMS-DBs/Pfam-A.hmm
2 /path/seqs_3_6 3.6.-.- /path/seqs_non_3_6 non_3.6.-.-
```

Ο αλγόριθμος ξεκινά με την δημιουργία ενός φακέλου “process_archives” στον οποίο θα συγκεντρωθούν διάφορα αρχεία κειμένου. Στην συνέχεια εκτελούνται δυο υπορουτίνες (subroutines) του Hmmer_Process.pm, η πρώτη (**process_file**) δημιουργεί το αρχείο “process_file”, στο οποίο θα καταγραφούν σημαντικές πληροφορίες κατά την εξέλιξη του αλγόριθμου και η δεύτερη (**mysql_table**) δημιουργεί έναν πίνακα με το όνομα “$table” στην βάση δεδομένων “$db”. Το επόμενο βήμα του αλγόριθμου είναι η ανάλυση κάθε αρχείου αλληλουχιών. Εκτελούνται τρεις

υπορουτίνες του Hmmer_Process.pm. Η υπορουτίνα **del_names** διαβάζει το αρχείο των αλληλουχιών και επανατυπώνει τις αλληλουχίες στο αρχείο “$hmmer_input”, χωρίς περιττές πληροφορίες στον τίτλο τους. Η ολοκλήρωση της **del_names** σημειώνεται στο αρχείο “process_file”. Η υπορουτίνα **run_hmmer** εκτελεί το πρόγραμμα HMMER3, τυπώνοντας στο “process_file” την επιτυχή ολοκλήρωση της ανάλυσης. Η **hmmer_parser** φορτώνει τα αποτελέσματα στον πίνακα “$table” και τυπώνει στο “process_file” τον αριθμό των ξεχωριστών PFAM κωδικών που βρέθηκαν για όλες τις αλληλουχίες. Στο τελικό στάδιο του αλγόριθμου εκτελούνται τρεις υπορουτίνες του Hmmer_Process.pm με σκοπό τη δημιουργία της πληροφορίας εκπαίδευσης, η οποία θα είναι ένα αρχείο σε μορφή csv. Η υπορουτίνα **pfam_tables** δημιουργεί ένα πίνακα στην βάση “$db”, όπου καταγράφονται οι ξεχωριστοί PFAM κωδικοί που βρέθηκαν στην ανάλυση όλων των αρχείων “$file_in”. Η υπορουτίνα **create_csv** δημιουργεί το τελικό αρχείο csv. Στο αρχείο αυτό κάθε γραμμή αντιστοιχεί σε μια αλληλουχία, και κάθε στήλη σε έναν κωδικό PFAM (εκτός από τις δυο πρώτες, όπου αναγράφονται τα ονόματα και οι κλάσεις των αλληλουχιών). Αν ο HMMER3 έχει βρει ομοιότητα μεταξύ μιας αλληλουχίας και ενός PFAM κωδικού, τότε στο κελί καταγράφεται το σκορ της στοίχισης (bit score), αλλιώς η τιμή μηδέν. Το συγκεκριμένο αρχείο είναι η πληροφορίας εκπαίδευσης και παρουσιάζεται στην εικόνα 3.1.

Feature	[class]	[PF09334.6]	[PF01588.15]	[PF00133.17]	[PF08264.8]	[PF01406.14]	[PF01000.21]	[PF01193.19]	[PF00056.18]	[PF02866.13]
UniRef50_A1RSE3	non_3.2.-.-	0	0	0	0	0	92.2	42.3	0	0
UniRef50_A1RW47	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_A3CTY6	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_A3MU00	non_3.2.-.-	106.1	0	205.8	60.4	41.3	0	0	0	0
UniRef50_A4YD92	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_B1L5U5	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_B2J5F8	non_3.2.-.-	0	0	0	0	0	0	0	115.8	109.3
UniRef50_B3EWG9	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_C3N549	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_C3N5H5	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_C5A4H7	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_D0KQM8	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_D3SQW0	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O08358	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O26377	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O26911	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O27083	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O27397	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O27911	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O28088	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O28550	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O28786	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O28987	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O29021	non_3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O29876	3.2.-.-	0	0	0	0	0	0	0	0	0
UniRef50_O32462	3.2.-.-	0	0	0	0	0	0	0	0	0

Εικόνα 3.1: Πληροφορία εκπαίδευσης PFAM χαρακτηριστικών.

Επιπλέον κατασκευάζεται ένα αρχείο csv, που περιέχει την παραπάνω πληροφορία εκπαίδευσης, αλλά με διαφορετική δομή. Το αρχείο αυτό

χρησιμοποιείται από την υπορουτίνα **pfam_for_prof** και δημιουργείται ένας πίνακας με όνομα “$table”_pfam””, ο οποίος περιέχει τρεις στήλες, τα ονόματα των πρότυπων παραδειγμάτων, την κλάση τους και όλα τα σκορ των PFAM κωδικών (εικόνα 3.2). Ο πίνακας θα χρησιμοποιηθεί στη συνέχεια για την κατασκευή της πληροφορίας εκπαίδευσης με PROFEAT χαρακτηριστικά.

#	ind	query_name	vector_1	vector_2
1	2	UniRef50_A1RSE3	non_3.2.-.-	0,0,0,0,0,92.2,42.3,0,0,0,0

PROFEAT στον αντίστοιχο server. Τα αποτελέσματα της ανάλυσης συγκεντρώνονται σε κοινό κατάλογο (./profeat_files). Οι μεταβλητές εισόδου είναι οι εξής:

$class : το όνομα του πίνακα που περιέχει τα PFAM αποτελέσματα, συνήθως είναι η κλάση των θετικών παραδειγμάτων, π.χ. hyd_3_1, hyd_3_2 κτλ.

$db : το όνομα της βάσης δεδομένων, “ML_archaea_x_factor ”.

$user_name : όνομα χρήστη του MySQL Workbench

$password : κωδικός χρήστη του MySQL Workbench

Η εντολή στην γραμμή εντολών (command line) για θετική κλάση την 3.6.-.-, έχει την εξής μορφή:

```
> profeat_process.pl hyd_3_6 ML_archaea_xfactor MySQL_username
MySQL_password
```

Ο συγκεκριμένος αλγόριθμος εκτελεί τρεις βασικές διαδικασίες. Αρχικά ενώνει όλα τα αρχεία με τα αποτελέσματα του PROFEAT σε ένα ενιαίο αρχείο. Στη συνέχεια φορτώνει τα δεδομένα του ενιαίου αρχείου σε έναν πίνακα της βάσης “ML_archaea_db_xfactor ”, με όνομα “$table”_profeat””. Τέλος, συνδυάζοντας τον πίνακα “$table”_pfam””, που έχει προκύψει από το πρόγραμμα hmmer_process.pl και τον “$table”_profeat”” τυπώνει την ζητούμενη πληροφορία εκπαίδευσης.

Πιο αναλυτικά, όλα τα δεδομένα του καταλόγου ./profeat_files τυπώνονται κατάλληλα σε ένα πρόχειρο αρχείο “$draft_file_1”. Στη συνέχεια εκτελούνται τρεις υπορουτίνες του Profeat_Process.pm. Με την υπορουτίνα **prof_shaping** τυπώνεται ένα δεύτερο πρόχειρο αρχείο κατάλληλης διαμόρφωσης, έτσι ώστε τα δεδομένα να χωρίζονται σε δυο στήλες: τα ονόματα των αλληλουχιών και τις τιμές των PROFEAT χαρακτηριστικών. Το αρχείο αυτό χρησιμοποιείται στην υπορουτίνα **mysql_table** για να φορτωθούν τα αποτελέσματα του PROFEAT σε έναν πίνακα δυο στηλών, ίδιας δομής με τον πίνακα που δημιούργησε η υπορουτίνα **pfam_for_prof** στο hmmer_process.pl.

Η τελευταία υπορουτίνα, **final_csv,** τυπώνει την πληροφορία εκπαίδευσης (τα χαρακτηριστικά PROFEAT είναι συνεχείς μεταβλητές). Το πρόβλημα σε αυτό το σημείο είναι η έλλειψη γνώσης της κλάσης κάθε αλληλουχίας. Για τον λόγο αυτό χρησιμοποιείται ο πίνακας “$table”_pfam””.

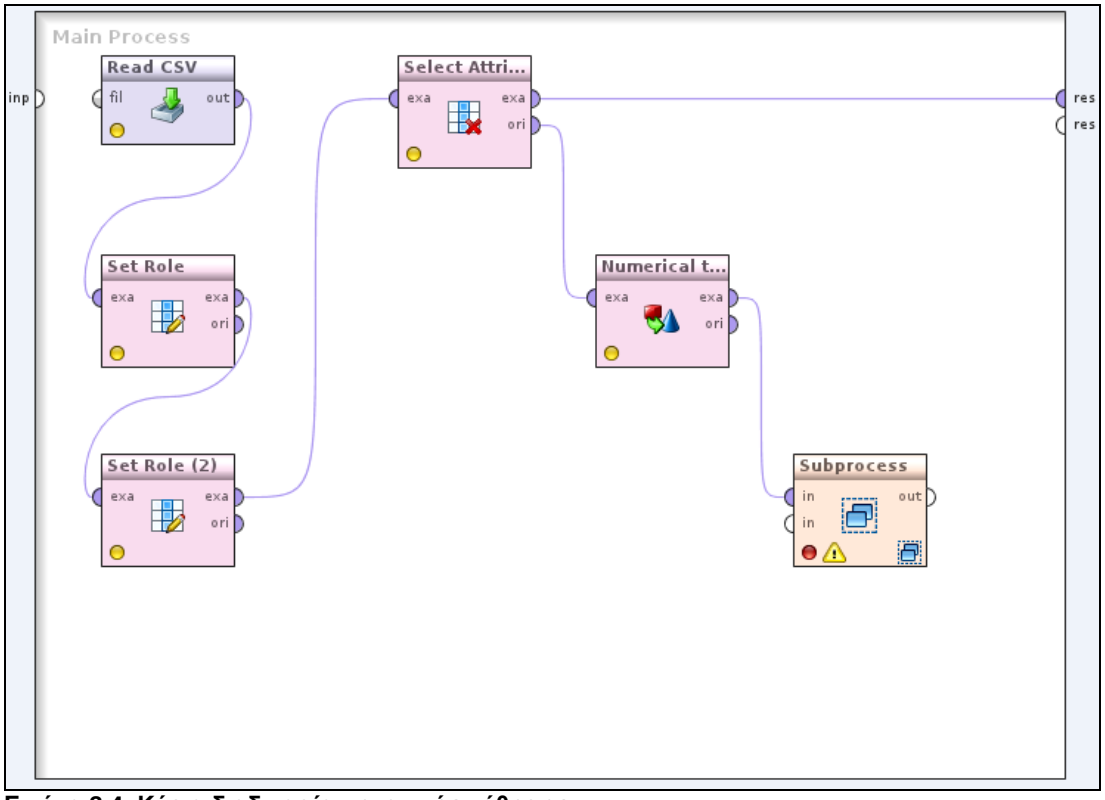
Για κάθε εγγραφή του πίνακα “$table”_profeat”” εντοπίζεται η αντίστοιχη εγγραφή και η κλάση της στον πίνακα “$table”_pfam””. Με τον τρόπο αυτό τυπώνεται το τελικό αρχείο csv.

Feature	[class]	[G1.1.1.1]	[G1.1.1.2]	[G1.1.1.3]	[G1.1.1.4]	[G1.1.1.5]	[G1.1.1.6]	[G1.1.1.7]	[G1.1.1.8]	[G1.1.1.9]	[G1.1.1.10]	[G1.1.1.11]
UniRef50_A1RSE3	non_3.2.-.-	7.5	0	3.75	6.25	1.25	2.5	1.25	10	3.75	12.5	5
UniRef50_A1RW47	3.2.-.-	9.79021	0	6.993007	9.79021	2.797203	5.594406	0.699301	4.195804	4.895105	13.286713	4.195804
UniRef50_A3CTY6	3.2.-.-	9.375	1.041667	4.6875	5.208333	1.5625	9.895833	2.083333	2.604167	2.604167	11.458333	2.083333
UniRef50_A3MU00	non_3.2.-.-	11.25	0	1.25	6.25	10	6.25	3.75	5	5	5	3.75
UniRef50_A4YD92	non_3.2.-.-	8.75	0	3.75	10	3.75	7.5	2.5	8.75	12.5	7.5	3.75
UniRef50_B1L5U5	non_3.2.-.-	5	1.25	5	10	5	6.25	1.25	11.25	1.25	8.75	6.25
UniRef50_B2J5F8	non_3.2.-.-	11.25	1.25	5	3.75	0	8.75	1.25	10	1.25	10	3.75
UniRef50_B3EWG9	non_3.2.-.-	11.25	0	2.5	1.25	3.75	12.5	0	5	5	10	0
UniRef50_C3N549	non_3.2.-.-	3.75	0	3.75	6.25	8.75	5	3.75	2.5	1.25	16.25	2.5
UniRef50_C3N5H5	3.2.-.-	6.280193	1.449275	3.864734	7.246377	3.864734	3.864734	1.449275	9.661836	7.729469	13.52657	1.932367
UniRef50_C5A4H7	non_3.2.-.-	11.25	0	5	6.25	6.25	11.25	1.25	3.75	7.5	10	2.5
UniRef50_D0KQM8	3.2.-.-	3.428571	0.285714	6.142857	8.142857	5.428571	6.571429	1.571429	8.142857	7.857143	8.285714	2
UniRef50_D3SQW0	non_3.2.-.-	8.75	0	7.5	5	1.25	10	1.25	2.5	0	10	1.25
UniRef50_O08358	non_3.2.-.-	5	2.5	2.5	6.25	2.5	6.25	1.25	11.25	5	6.25	2.5
UniRef50_O26377	non_3.2.-.-	11.25	1.25	5	6.25	0	13.75	1.25	10	7.5	8.75	3.75
UniRef50_O26911	non_3.2.-.-	10	3.75	6.25	7.5	3.75	6.25	0	10	7.5	6.25	6.25
UniRef50_O27083	non_3.2.-.-	6.25	0	5	10	0	8.75	1.25	5	8.75	3.75	6.25

Εικόνα 3.3: Πληροφορία εκπαίδευσης PROFEAT χαρακτηριστικών.

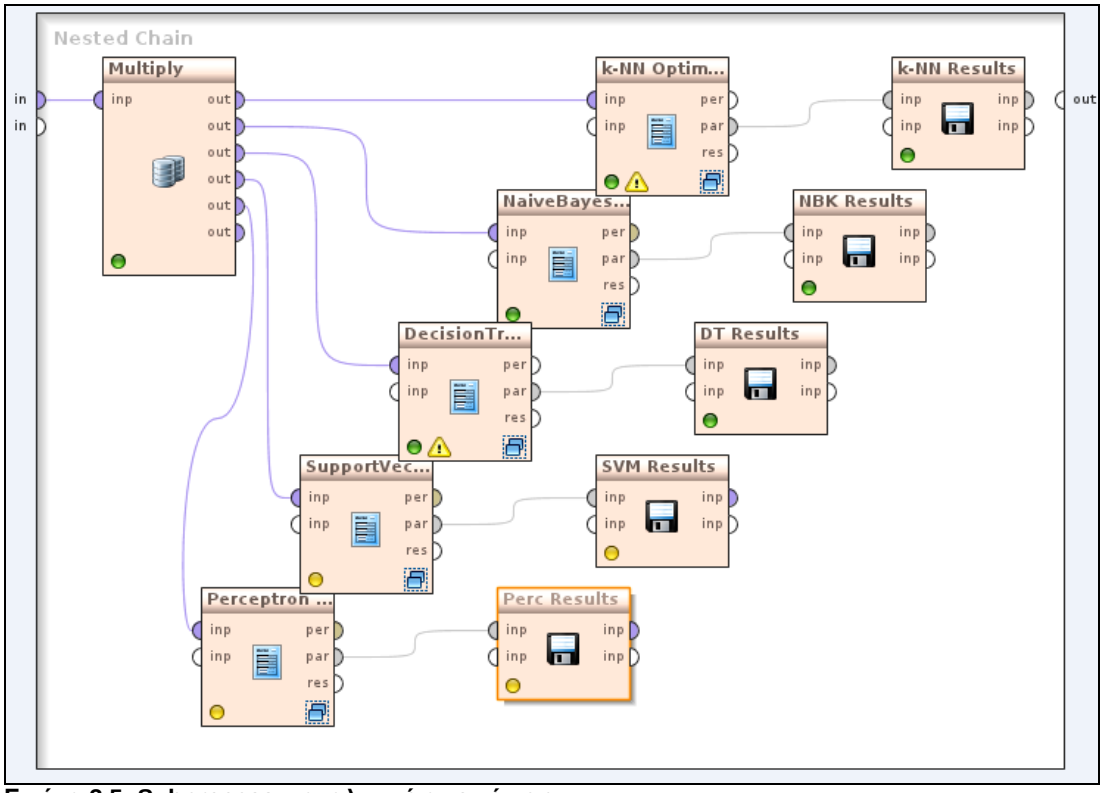
3.2.4. Διαδικασία Μηχανικής Μάθησης

Για την μηχανική μάθηση χρησιμοποιήθηκε το λογισμικό RAPIDMINER (έκδοση 5.2.008). Οι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιήθηκαν είναι ο κ-Κοντινότερος-Γείτονας (k-Nearest-Neighbor), ο Μπεϋζιανός Ταξινομητής Πυρήνα (Naive Bayes Kernel), τα Δέντρα Απόφασης (Decision Trees), οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) και ο αλγόριθμος Perceptron. Λόγω της ύπαρξης παραμέτρων που επηρεάζουν την απόδοση του κάθε αλγόριθμου, επιλέχτηκε η διαδικασία της βελτιστοποίησης παραμέτρων (Parameters Optimization). Για κάθε πληροφορία εκπαίδευσης κάθε αλγόριθμος εκτελέσθηκε για διαφορετικές τιμές παραμέτρων. Η διαδικασία της βελτιστοποίησης είχε σαν σκοπό τη μελέτη της μεταβολής της απόδοσης κάθε αλγόριθμου ανάλογα με τις τιμές των παραμέτρων. Ο αλγόριθμος ανάλυσης μιας πληροφορίας εκπαίδευσης, περιγράφεται ως εξής:



Εικόνα 3.4: Κύρια διαδικασία μηχανικής μάθησης.

Η λειτουργία **Read CSV** διαβάζει την πληροφορία εκπαίδευσης. Οι δυο λειτουργίες **Set Role** ορίζουν τον ρόλο που μπορεί να έχει κάποια στήλη στην πληροφορία εκπαίδευσης. Στον αλγόριθμο η στήλη ονομάτων των αλληλουχιών ορίστηκε ως “id” και η στήλη των κλάσεων ως “label”. Η λειτουργία **Select Attributes** επιλέγει ποια χαρακτηριστικά θα χρησιμοποιηθούν κατά την μάθηση και ταξινόμηση. Στην συγκεκριμένη ενότητα επιλέχθηκαν όλα όσα έχουν πολυωνυμικό χαρακτήρα, δηλαδή όλα. Τέλος η λειτουργία **Numerical to Real** μετατρέπει τα χαρακτηριστικά που έχουν οριστεί ότι λαμβάνουν ακέραιες τιμές, σε χαρακτηριστικά πραγματικών τιμών. Η διαδικασία **Subprocess** παρουσιάζεται στην εικόνα 3.5.



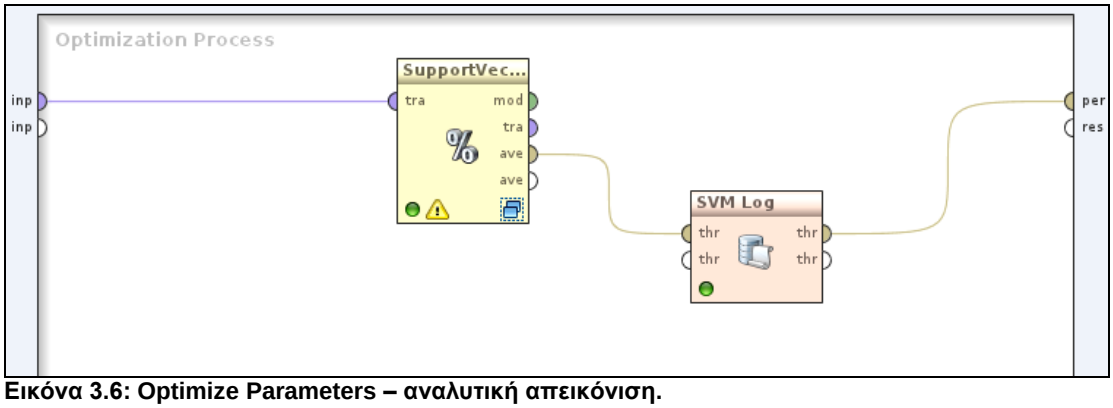
Εικόνα 3.5: Subprocess – αναλυτική απεικόνιση.

Η λειτουργία **Multiply** δημιουργεί αντίγραφα της πληροφορίας εκπαίδευσης. Κάθε αντίγραφο μεταφέρεται στην διαδικασία βελτιστοποίησης κάθε αλγόριθμου (**Grid Optimization**, εδώ έχει ονομαστεί Algorithm Optimization). Σε κάθε διαδικασία βελτιστοποίησης ο χρήστης επιλέγει τις παραμέτρους που θέλει να εξετάσει και ορίζει τις διαφορετικές τιμές τους. Στην εργασία οι παράμετροι που εξετάστηκαν είναι οι εξής:

Αλγόριθμος	***Παράμετρος***	***Τιμές Παραμέτρου***
k-NN	*k*	1 4 8 12
	kernel_type	dot radial polynominal multiquadric
	kernel_degree	1 2 4 10
	kernel_sigma_1	0.01 0.1 1 2 4 6 10
	kernel_gamma	0.01 0.1 1 2 4 6 10
Naive Bayes Kernel	*minimum_bandwidth*	0.1 0.5 1 2 4 8 12 20 30

Decision Trees	*criterion*	gini_index information_gain gain_ratio accuracy
	maximal_depth	10 20 60 120
	minimum_size_for_split	1 3 6 8 10
	confidence	0.01 0.1 0.25 0.5
Support Vector Machines	*kernel_type*	dot radial polynominal multiquadric
	kernel_degree	1 2 4 10
	kernel_sigma_1	0.01 0.1 1 2 4 6 10
	kernel_gamma	0.01 0.1 1 2 4 6 10
	C	0 1 2 5 10 100
Perceptron	*rounds*	1 5 20 50 100 200 300 500 1000
	learning_rate	0.001 0.01 0.05 0.1 0.2 0.3 0.4 0.5 0.8

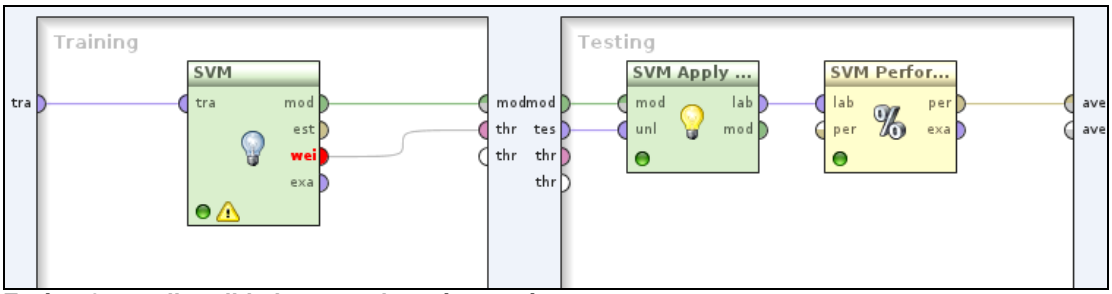
Η καλύτερη απόδοση κάθε βελτιστοποίησης αναγράφεται σε αρχείο κειμένου με την λειτουργία **Write File** (εδώ έχει ονομαστεί Algorithm Results). Το εσωτερικό κάθε διαδικασίας βελτιστοποίησης είναι ίδιο για όλους τους αλγόριθμους. Για τον SVMs περιγράφεται στην εικόνα 3.6.



Εικόνα 3.6: Optimize Parameters – αναλυτική απεικόνιση.

Με την λειτουργία **Split Validation** (εδώ έχει ονομαστεί SupportVectorMachines Validation) εκτελείται η διαδικασία της μάθησης, ταξινόμησης και αξιολόγησης. Στην **Split Validation** η πληροφορία εκπαίδευσης χωρίζεται τυχαία σε δυο μέρη. Εδώ επιλέχθηκε ποσοστό 70% για την πληροφορία που χρησιμοποιήθηκε για μάθηση και ποσοστό 30% για

την πληροφορία που χρησιμοποιήθηκε για ταξινόμηση και αξιολόγηση της μάθησης. Η αναλυτική διαδικασία της **Split Validation** παρουσιάζεται στην εικόνα 3.7. Με την λειτουργία **Log** (εδώ SVM Log) τυπώνονται σε αρχείο κειμένου η απόδοση του αλγόριθμου για κάθε ομάδα τιμών των εξεταζόμενων παραμέτρων του.



Εικόνα 3.7: Split Validation – αναλυτική απεικόνιση.

Κατά την μάθηση εκτελείται ο αλγόριθμος μάθησης (εδώ SVM) ώστε να υπολογιστεί ο ταξινομητής. Στην συνέχεια με την λειτουργία **Apply Model** εφαρμόζεται η ταξινόμηση και με την λειτουργία **Performance** υπολογίζεται η απόδοση του ταξινομητή (εδώ SVM Apply Model και SVM Performance).

Λόγω του μεγάλου αριθμού πληροφοριών εκπαίδευσης, κατασκευάστηκε ένα αρχείο Perl, **rm_multiple_validation.pl**, το οποίο δέχεται ως είσοδο όσες πληροφορίες εκπαίδευσης επιθυμεί ο χρήστης. Με το πρόγραμμα αυτό αναλύθηκαν μαζί οι πέντε πληροφορίες εκπαίδευσης κάθε κατηγορίας υδρολασών (πέντε διαφορετικοί συντελεστές) για ένα είδος χαρακτηριστικού. Η έξοδος του προγράμματος ήταν ένα ενιαίο αρχείο μορφής csv, όπου περιέχονταν όλα τα αποτελέσματα όλων των αλγόριθμων, συναρτήσει των παραμέτρων τους, και ένα ενιαίο αρχείο κειμένου, όπου καταγράφονταν οι μέγιστες αποδόσεις και οι πίνακες σύγχυσης. Η εντολή (command line) για τις πληροφορίες εκπαίδευσης PFAM χαρακτηριστικών της κατηγορίας 3.1.-.- ήταν:

```
> rm_multiple_validation.pl /path/multiple_validation.rmp 5
hyd_3_1 /path/x05/pfam_hyd_3_1.csv x05
/path/x01/pfam_hyd_3_1.csv x1 /path/x2/pfam_hyd_3_1.csv x2
/path/x3/pfam_hyd_3_1.csv x3 /path/x4/pfam_hyd_3_1.csv x4
```

3.3. Αξιοποίηση Αλγόριθμων Μηχανικής Μάθησης

3.3.1. Κατασκευή Βάσης Δεδομένων Θερμοανθεκτικών και μη Θερμοανθεκτικών Μικροοργανισμών

Στην τελευταία ενότητα του τεχνικού μέρους, χρησιμοποιήθηκαν οι αλγόριθμοι μηχανικής μάθησης για τον διαχωρισμό θερμοανθεκτικών και μη θερμοανθεκτικών υδρολασών. Πιο συγκεκριμένα, αφού μελετήθηκαν τα αποτελέσματα του πρώτου μέρους και διαπιστώθηκε η συμπεριφορά των αλγόριθμων στα δυο διαφορετικά είδη πληροφορίας εκπαίδευσης, χρησιμοποιήθηκαν τα συμπεράσματα για το δεύτερο μέρος. Σκοπός ήταν η ανάπτυξη αλγόριθμου μηχανικής μάθησης που για μια συγκεκριμένη κατηγορία υδρολασών (π.χ. 3.1.-.-) θα μπορούσε να διακρίνει αν μια άγνωστη αλληλουχία προερχόταν από μεσόφιλο, θερμόφιλο ή υπερθερμόφιλο μικροοργανισμό.

Αρχικά έπρεπε να βρεθούν οι βέλτιστες θερμοκρασίες ανάπτυξης διαφόρων μικροοργανισμών, στους οποίους θα αναζητούνταν στη συνέχεια οι αλληλουχίες υδρολασών. Για τον σκοπό αυτό μελετήθηκε αντίστοιχη βιβλιογραφία και τα αποτελέσματα συγκεντρώθηκαν σε αρχείο κειμένου “opt_temperature.csv” και παρουσιάζονται στον παρακάτω πίνακα:

Πίνακας 3.1: Μικροοργανισμοί που χρησιμοποιήθηκαν

Μικροοργανισμός	Βέλτιστη Θερμοκρασία Ανάπτυξης$\left(^{o}\mathrm{C}\right)$	Κατηγορία Μικροοργανισμών
hyperthermus_butylicus	100	archaea
pyrobaculum_islandicum	100	archaea
methanopyrus_kandleri	98	archaea
pyrobaculum_aerophilum	98	archaea
pyrococcus_horikoshii	98	archaea
pyrococcus_abyssi	97	archaea
pyrococcus_furiosus	97	archaea
aeropyrum_pernix	95	archaea
staphylothermus_marinus	92	archaea
aquifex_aeolicus	90	bacteria
thermococcus_kodakaraensis	90	archaea

thermofilum_pendens	88	archaea
thermoproteus_neutrophilus	88	archaea
aquifex_pyrophilus	85	bacteria
archaeoglobus_fulgidus	85	archaea
methanococcus_jannaschii	85	archaea
sulfolobus_solfataricus	85	archaea
thermococcus_litoralis	83	archaea
fervidobacterium_nodosum	80	bacteria
methanococcus_vulcanius	80	archaea
sulfolobus_acidocaldarius	80	archaea
thermotoga_maritima	80	bacteria
thermotoga_neapolitana	80	bacteria
thermosipho_africanus	77	bacteria
metallosphaera_sedula	75	archaea
thermoanaerobacter_tengcongensis	75	bacteria
thermus_thermophilus	75	bacteria
bacillus_licheniformis	70	bacteria
bacillus_stearothermophilus	70	bacteria
thermus_aquaticus	70	bacteria
thermoanaerobacter_pseudethanolicus	69	bacteria
methanothermobacter_thermautotrophicus	65	bacteria
acidothermus_cellulolyticus	60	bacteria
clostridium_thermocellum	60	bacteria
picrophilus_torridus	60	archaea
symbiobacterium_thermophilum	60	bacteria
thermoplasma_acidophilum	60	archaea
thermoplasma_volcanium	60	archaea
moorella_thermoacetica	58	bacteria
chloroflexus_aurantiacus	55	bacteria
geobacillus_kaustophilus	55	bacteria
geobacillus_stearothermophilus	55	bacteria
nitratiruptor_sp.	55	bacteria
pelotomaculum_thermopropionicum	55	bacteria
thermosynechococcus_elongatus	55	bacteria
thermobifida_fusca	50	bacteria
chlorobium_tepidum	48	bacteria
methylococcus_capsulatus	45	bacteria
bacillus_polymyxa	40	bacteria
serratia_marcescens	40	bacteria

lactococcus_lactis	40	bacteria
salmonella_typhimurium	37	bacteria
clostridium_tetani	37	bacteria
desulfovibrio_vulgaris	37	bacteria
flavobacterium_meningosepticum	37	bacteria
treponema_pallidum	37	bacteria
vibrio_cholerae	37	bacteria
pseudomonas_stutzeri	35	bacteria
sulfurovum_sp.	33	bacteria
bacillus_cereus	30	bacteria
oceanobacillus_iheyensis	30	bacteria
bacillus_subtilis	30	bacteria
lactobacillus_acidophilus	30	bacteria
yersinia_pestis	30	bacteria
synechocystis_sp.	26	bacteria
leifsonia_xyli	23	bacteria
methanococcoides_burtonii	23	archaea

[10][11][78][79[80][81][82]

Από την βάση δεδομένων UniProt/SwissProt (έκδοση 11/2012) αποκτήθηκαν όλες οι αλληλουχίες υδρολασών των παραπάνω μικροοργανισμών σε αρχείο μορφής fasta, “seqs.fasta”. Επιπλέον αποκτήθηκε ένα αρχείο κειμένου “data.txt”, που περιείχε τον κωδικό κάθε αλληλουχίας, τον μικροοργανισμό στο οποίο έχει βρεθεί και τον κωδικό Ενζυμικής Ονοματολογίας.

Το επόμενο βήμα ήταν η μεταφορά της πληροφορίας στο MySQL Workbench, έτσι ώστε να είναι εύκολη η διαχείρισή της. Αρχικά με το αρχείο Perl **create_MySQL_schema.pl** δημιουργήθηκε βάση δεδομένων με όνομα “ML_thermo_db” και στη συνέχεια χρησιμοποιήθηκε το αρχείο **all_data_parser.pl**, ώστε να φορτωθεί η πληροφορία των αρχείων “seqs.fasta” και “data.txt” σε έναν πίνακα με όνομα “full_data”. Ο αλγόριθμος δέχεται ως είσοδο τα αρχεία “seqs.fasta” και “data.txt”. Για την πληροφορία του κάθε αρχείου κατασκευάζει τον κατάλληλο πίνακα. Το κοινό στοιχείο των δυο πινάκων είναι τα ονόματα των αλληλουχιών. Επομένως με κριτήριο το όνομα κάθε αλληλουχίας ενώνει τα υπόλοιπα στοιχεία των δυο πινάκων και προκύπτει ο τελικός με στήλες: protein_id, ec_number, organism, sequence.

#	ind	protein_id	ec_number	organism	seq
1	1	Q9V2L2	3.5.99.7	pyrococcus_abyssi	MHPKVDALLSRFPRITLIPWETPIQYLPRISRELGVDVYVKRDDLTGLGIGGNKIRKLEFLLGDALSRGCDTVITIGAVHSNHAFV
2	2	Q8U4R3	3.5.99.7	pyrococcus_furiosus	MHPKVQSLLSKFPRVELIPWETPIQYLPNISKLVGADIYVKRDDLTGLGIGGNKIRKLEYLLGDAIIRKADVIITVGAVHSNHAFV
3	3	O57809	3.5.99.7	pyrococcus_horikoshii	MHPKIFALLAKFPRVELIPWETPIQYLPNISREIGADVYIKRDDLTGLGIGGNKIRKLEYLLGDALSKGADVVITVGAVHSNHAFV
4	4	Q9WY68	3.5.99.7	thermotoga_maritima	MRIDLSLKPTPVQFLKRLSEKYGFNIYVKRDDLTELVGSGNKIRKLEYLLWEALKKGATTVFTCGGLQSNHARATAYVSRRYGLK
5	5	P37878	3.2.2.21	bacillus_subtilis	MTWHEVNDVIVITLPEIFDMNANLGYLTREKNECMYEIENNIITKVIAIGEIRSLVQVSVINNKQMIVQFLNDSRPVEQWKREEI
6	6	Q9Y9P1	3.2.2.-	aeropyrum_pernix	MGRLLPKSFYYRQPDVVARELLGKILVSCASGACMRCMVTEAEAYFGECDPASRARRGRGRIWRALYGEPGRALVYGMHRQW
7	7	B7JSE3	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVNGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
8	8	Q73CV5	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
9	9	B7IJ32	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
10	10	C1EZP2	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
11	11	B7HEV9	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
12	12	B7HXM9	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVNGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
13	13	A7GLP0	3.2.2.-	bacillus_cereus	MKAPPPFYEGDTLDVAKKLLGQKLVHIVDGVKRSGYIVEVEAYKGPDDKAAHSYGGRRTERTEIMFGAPGHAYVYLIYGMYHCF
14	14	B9IRB4	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPGDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
15	15	Q81HD0	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
16	16	Q63FD4	3.2.2.-	bacillus_cereus	MQAPPSFYEGDTLEVAKKLLGQKLVHIVDGIKRSGIIVEVEAYKGPDDKAAHSYGGRRTDRTEVMFGAPGHAYVYLIYGMYHCF
17	17	Q65DF9	3.2.2.-	bacillus_licheniformis	MNPAHEPLPLEFYNKPTIELARSLLGCLLVKETEEGPASGYIVETEAYKGPGDRAAHSYGNRRTKRTEIMYREAGVVYTYTMHTH
18	18	P94378	3.2.2.-	bacillus_subtilis	MTREKNPLPITFYQKTALELAPSLLGCLLVKETDEGTASGYIVETEAYMGAGDRAAHSFNNRRTKRTEIMFAEAGRVYTYVMHTH
19	19	Q8KBD5	3.2.2.-	chlorobium_tepidum	MKRLGADFYQMPTILLAERLLGKIFVHHEVSGRVTKGRIVETEAYLGDGDEACHAWRGMTERNHVMFGPPGHLYIYFSYGCHY
20	20	Q896H4	3.2.2.-	clostridium_tetani	MKIQRKFYEKSALQVAKYLLGKILVNEVEGITLKGKIVETEAYIGAIDKASHAYGGKKTERVMPLYGKPGTAYVYLIYGMYHCFNV

Εικόνα 3.8: Πίνακας full_data.

Επιπλέον δημιουργήθηκε ο πίνακας “opt_temperatures” στον οποίο φορτώθηκε όλη η πληροφορία του πίνακα 3.1. Ο λόγος ήταν η ανάγκη ύπαρξης ενός πίνακα, όπου θα περιείχε την σχέση οργανισμού-βέλτιστης θερμοκρασίας. Το βήμα αυτό πραγματοποιήθηκε με το αρχείο Perl **temperature_parser.pl**. Ο πίνακας “opt_temperatures” αποτελείται από τρεις στήλες: organism, temperature, category.

Η βάση δεδομένων “ML_thermo_db” ήταν έτοιμη να χρησιμοποιηθεί ώστε να δημιουργηθούν οι επιθυμητές πληροφορίες εκπαίδευσης. Το κλειδί στην δημιουργία μιας πληροφορίας εκπαίδευσης ήταν τα θερμοκρασιακά όρια των δυο κλάσεων. Για παράδειγμα, η πληροφορία εκπαίδευσης μπορούσε να περιέχει ως αρνητικά πρότυπα (αρνητική κλάση) τις αλληλουχίες οργανισμών που ζουν κάτω από 40 °C και ως θετικά (θετική κλάση) εκείνες των οργανισμών που ζουν πάνω από 80 °C. Για την δημιουργία πληροφοριών εκπαίδευσης κατασκευάστηκε το αρχείο Perl **select_thermo.pl** κι το πακέτο **Select_Thermo.pm**.

Περιγραφή του αρχείου Perl select_thermo.pl: Το πρόγραμμα δημιουργεί αρχεία fasta, τα οποία θα αναλυθούν με το hmmer_process.pl και το profeat_process.pl ώστε να προκύψουν οι πληροφορίες εκπαίδευσης. Ο χρήστης ονομάζει τις κατηγορίες αλληλουχιών όπως επιθυμεί (π.χ. meso, thermo, hyper), ορίζει τα θερμοκρασιακά όρια της καθεμίας και δηλώνει αν θέλει να επιλεχτούν όλες οι αλληλουχίες από την “ML_thermo_db” ή περιορισμένος αριθμός με τυχαίο τρόπο. Οι μεταβλητές εισόδου είναι:

$db : το όνομα της βάσης δεδομένων, “ML_thermo_db ”

$user_name : όνομα χρήστη του MySQL Workbench

$password : κωδικός χρήστη του MySQL Workbench

$N : το πλήθος των κλάσεων, στη συγκεκριμένη εργασία 2

$ec : η κατηγορία Ενζυμικής Ονοματολογίας που θα ανήκουν όλες οι αλληλουχίες

Το πλήθος των κλάσεων επηρεάζει τον αριθμό των υπόλοιπων μεταβλητών εισόδου, όπως και στο πρόγραμμα **hmmer_process.pl**:

```
for ($i=5; $i<5*$N+5; $i=$i+5) {
        my $category = $ARGV[$i]; # το όνομα της κλάσης
        my $temperature_low= $ARGV[$i+1]; # το κάτω θερμοκρασιακό όριο
        my $temperature_high = $ARGV[$i+2]; # το άνω θερμοκρασιακό όριο
        my $type_of_select = $ARGV[$i+3]; # all ή limit
        my $Number_limit = $ARGV[$i+4]; #  all (αν $type_of_select = all) και αριθμός π.χ.
100 (αν $type_of_select = limit)
        push (@{$hash{$category}}, $temperature_low);
        push (@{$hash{$category}}, $temperature_high);
        push (@{$hash{$category}}, $type_of_select);
        push (@{$hash{$category}}, $Number_limit);
}
```

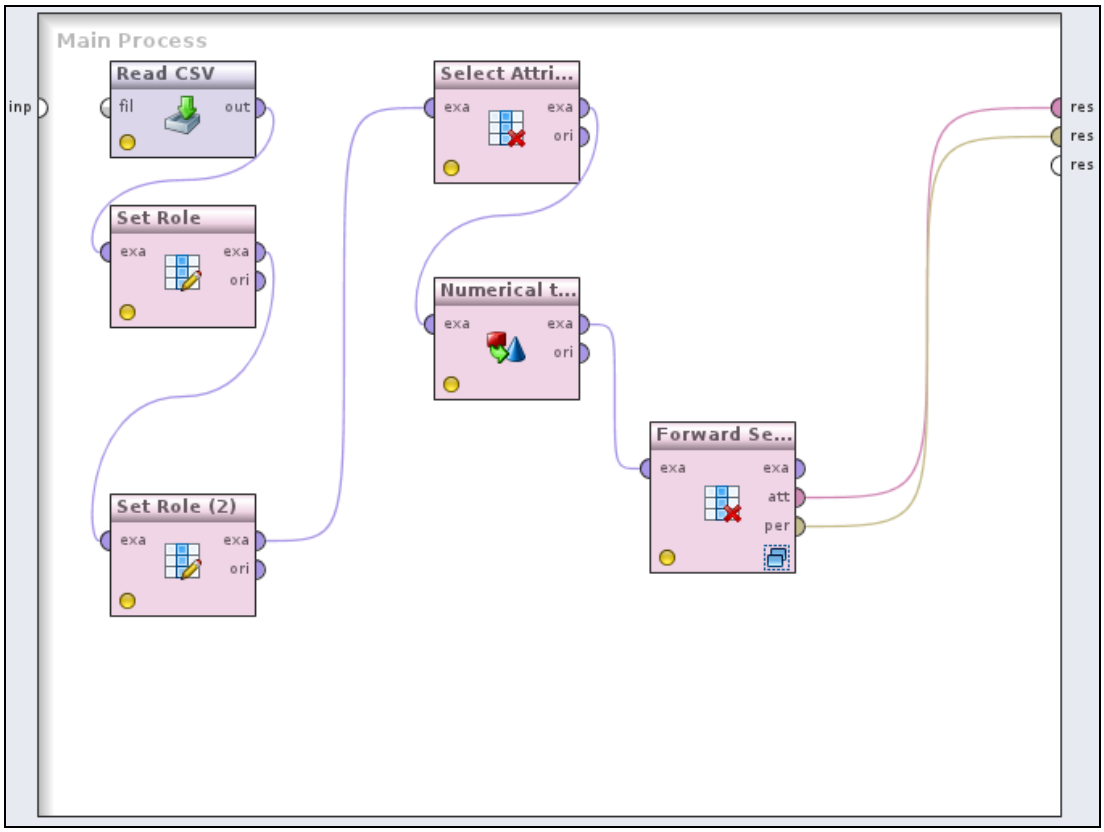
Επομένως για 2 κλάσεις ενζυμικότητας 3.4.-.- (μεσόφιλοι 20-30°C με περιορισμό 200 αλληλουχιών, υπερθερμόφιλοι 90-110°C χωρίς περιορισμό) η εντολή είναι η εξής:

```
> select_thermo.pl ML_Thermo_db MySQL_username MySQL_password
2 3.4. meso_20_30 20 30 limit 200 hyper_90_110 90 110 all all
```

Ο αλγόριθμος του προγράμματος εκτελεί δυο υπορουτίνες του πακέτου Select_thermo.pm. Αν για μια συγκεκριμένη θερμοκρασιακή κατηγορία υδρολασών έχει οριστεί να επιλεγούν όλες οι αντίστοιχες αλληλουχίες από τον πίνακα “all_data”, τότε χρησιμοποιείται η υπορουτίνα **select_seqs**, αντίθετα, για περιορισμένο αριθμό αλληλουχιών χρησιμοποιείται η υπορουτίνα **select_seqs_limit**. Για κάθε θερμοκρασιακή κατηγορία φορτώνονται οι αντίστοιχες αλληλουχίες σε αρχείο fasta και ομαδοποιούνται με το πρόγραμμα CD-HIT (κατώφλι ομοιότητας 70%). Η διαδικασία αυτή πραγματοποιείται με την υπορουτίνα **cd_hit_clustering.** Με τον τρόπο αυτό δημιουργούνται τα δυο αρχεία fasta, που θα να αναλυθούν στη συνέχεια ώστε να προκύψουν οι πληροφορίες εκπαίδευσης.

3.3.2. Διαδικασία Μηχανικής Μάθησης

Οι πληροφορίες εκπαίδευσης αναλύθηκαν με τα προγράμματα **hmmer_process.pl** και **profeat_process.pl**, όπως στην ενότητα 3.2. Το τμήμα αυτό της εργασίας πραγματοποιήθηκε στο περιβάλλον του RAPIDMINER. Χρησιμοποιήθηκε η λειτουργία **Forward Selection** (εικόνα 3.9), η οποία επιλέγει συγκεκριμένα χαρακτηριστικά που μεγιστοποιούν την απόδοση του ταξινομητή, χρησιμοποιώντας την λειτουργία **Cross-Validation** (10 validations).



Εικόνα 3.9: Feature Selection με Cross-Validation.

Οι πληροφορίες εκπαίδευσης που αναλύθηκαν ήταν τρεις. Η πρώτη περιείχε υπερθερμόφιλες ($80^{o}C < T < 110^{o}C$) και μεσόφιλες ($20^{o}C < T < 40^{o}C$) υδρολάσες 3.1.-.-, η δεύτερη υδρολάσες ίδιων θερμοκρασιακών περιοχών με την πρώτη αλλά κατηγορίας 3.6.-.- και η τρίτη μεσόφιλες και θερμρόφιλες ($50^{o}C < T < 65^{o}C$) υδρολάσες της 3.1.-.-..

ΙΙΙ. ΑΠΟΤΕΛΕΣΜΑΤΑ - ΣΥΜΠΕΡΑΣΜΑΤΑ

1. ΠΡΟΒΛΕΨΗ ΠΡΩΤΕΪΝΙΚΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΜΕ ΧΡΗΣΗ ΟΝΤΟΛΟΓΙΩΝ

Το λογισμικό blastp ανακαλύπτει ομοιότητες στην αλληλουχία των αμινοξέων μεταξύ μιας άγνωστης πρωτεΐνης και μιας γνωστής. Το αποτέλεσμα περιέχει το ταίριασμα των δυο αλληλουχιών, χωρίς περαιτέρω πληροφορίες. Επιπλέον σε ένα μεγάλο όγκο αποτελεσμάτων είναι αδύνατο να ελέγχεται το κάθε ταίριασμα αλληλουχιών. Με τους κωδικούς Οντολογίας Γονιδίων και Ενζυμικής Ονοματολογίας ο χρήστης όχι μόνο χαρακτηρίζει το κάθε ταίριασμα με επιπλέον πληροφορίες, κυτταρικά, βιολογικά, μοριακά χαρακτηριστικά και ενζυμική λειτουργία, αλλά μπορεί να εντοπίσει εύκολα τα αποτελέσματα που τον ενδιαφέρουν (π.χ. όλα τα ταιριάσματα που καταδεικνύουν εύρεση υδρολάσης). Έτσι η βάση δεδομένων UniProt-GOA αποτελεί ένα άμεσο εργαλείο στην πρόβλεψη της λειτουργίας άγνωστων αλληλουχιών και στην εύρεση των επιθυμητών αποτελεσμάτων ανάμεσα σε πολλά άλλα.

Παρακάτω παρουσιάζονται τα αποτελέσματα εύρεσης ενζυμικών κλάσεων άγνωστων αλληλουχιών σε δείγμα μεταγονιδιωματικής ανάλυσης, το οποίο χωρίστηκε σε τρία αρχεία, ανάλογα με το μήκος των αλληλουχιών (set_1_85 με μήκος<85, set_86_300 με μήκος $\in$[86,300] και set_300+ με μήκος>300).

	set_1_85		**set_86_300**		**set_300+**	
	ταιριάσματα	ποσοστό	ταιριάσματα	ποσοστό	ταιριάσματα	ποσοστό
1.-.-.-	210	1,76	5251	2,32	4628	2,61
2.-.-.-	207	1,74	3090	1,37	4232	2,39
3.-.-.-	463	3,89	5466	2,42	5930	3,34
4.-.-.-	214	1,80	1400	0,62	1265	0,71
5.-.-.-	37	0,31	1077	0,48	799	0,45
6.-.-.-	280	2,35	2842	1,26	3477	1,96
no_EC_number	10494	88,15	206824	91,54	157098	88,54
σύνολο	11905		225950		177429	

2. ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

2.1. Μελέτη Συμπεριφοράς των Αλγόριθμων

Αρχικά παρουσιάζονται τα αποτελέσματα από το πρόγραμμα hmmer_process.pl. Οι αλληλουχίες αναλύθηκαν με το λογισμικό HMMER και με βάση τα αποτελέσματά του κατασκευάστηκε η κάθε πληροφορία εκπαίδευσης. Για κάθε πληροφορία εκπαίδευσης το hmmer_process.pl υπολόγισε τον αριθμό των θετικών και αρνητικών παραδειγμάτων και τους ξεχωριστούς PFAM κωδικούς που βρέθηκαν για τα θετικά παραδείγματα, για τα αρνητικά και όσους καταγράφηκαν στην πληροφορία εκπαίδευσης.

• **Κατηγορία υδρολασών: 3.1.-.-**

	Αριθμός πρότυπων παραδειγμάτων			**PFAM κωδικοί (χαρακτηριστικά)**			
	θετικά	αρνητικά	σύνολο	θετικών	αρνητικών	σύνολο	πληροφορίας εκπαίδευσης
×05	392	192	584	90	262	352	346
×1	392	392	784	90	414	504	489
×2	392	784	1176	90	546	636	623
×3	392	1176	1568	90	624	714	695
×4	392	1568	1960	90	662	752	734

• **Κατηγορία υδρολασών: 3.2.-.-**

	Αριθμός πρότυπων παραδειγμάτων			**PFAM κωδικοί (χαρακτηριστικά)**			
	θετικά	αρνητικά	σύνολο	θετικών	αρνητικών	σύνολο	πληροφορίας εκπαίδευσης
×05	36	18	54	28	35	63	62
×1	36	36	72	28	78	106	106
×2	36	72	108	28	140	168	168
×3	36	108	144	28	188	216	216
×4	36	144	180	28	230	258	256

• **Κατηγορία υδρολασών: 3.4.-.-**

	Αριθμός πρότυπων παραδειγμάτων			PFAM κωδικοί (χαρακτηριστικά)			
	θετικά	αρνητικά	σύνολο	θετικών	αρνητικών	σύνολο	πληροφορίας εκπαίδευσης
×05	97	49	146	57	111	168	167
×1	97	97	194	57	183	240	239
×2	97	194	291	57	248	305	299
×3	97	291	388	57	335	392	389
×4	97	388	485	57	424	481	479

• **Κατηγορία υδρολασών: 3.5.-.-**

	Αριθμός πρότυπων παραδειγμάτων			PFAM κωδικοί (χαρακτηριστικά)			
	θετικά	αρνητικά	σύνολο	θετικών	αρνητικών	σύνολο	πληροφορίας εκπαίδευσης
×05	168	84	252	38	168	206	205
×1	168	168	336	38	233	271	269
×2	168	336	504	38	404	442	437
×3	168	504	672	38	433	471	465
×4	168	672	840	38	528	566	560

• **Κατηγορία υδρολασών: 3.6.-.-**

	Αριθμός πρότυπων παραδειγμάτων			PFAM κωδικοί (χαρακτηριστικά)			
	θετικά	αρνητικά	σύνολο	θετικών	αρνητικών	σύνολο	πληροφορίας εκπαίδευσης
×05	213	107	320	113	192	305	294
×1	213	213	426	113	302	415	399
×2	213	426	639	113	457	570	553
×3	213	639	852	113	527	640	612
×4	213	852	1065	113	560	673	644

Όπως προκύπτει, το σύνολο των PFAM κωδικών, δεν είναι πάντα ίσο με το σύνολο των χαρακτηριστικών της πληροφορίας εκπαίδευσης. Ο λόγος είναι ότι ορισμένοι PFAM κωδικοί βρέθηκαν για θετικά και αρνητικά παραδείγματα, αλλά στην πληροφορία εκπαίδευσης καταγράφηκαν μια φορά. Στην συνέχεια παρουσιάζονται τα αποτελέσματα των αλγόριθμων μηχανικής μάθησης για τις πληροφορίες εκπαίδευσης PFAM χαρακτηριστικών, κάθε κατηγορίας υδρολασών.

- **<u>Κατηγορία υδρολασών: 3.1.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	96,53	99,13	95,80
	×1	96,10	98,98	92,38
	×2	98,27	97,14	97,14
	×3	97,64	96,00	95,24
	×4	98,45	97,25	94,64

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	90,75	100,00	86,55
	×1	87,88	98,73	74,29
	×2	98,27	99,01	95,24
	×3	97,64	98,32	92,86
	×4	98,11	97,20	92,86

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	95,38	99,12	94,12
	×1	93,94	100,00	86,87
	×2	97,98	99,00	99,24
	×3	95,71	99,07	84,92
	×4	97,93	99,02	90,18

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	99,50	100,00	98,95
	×3	99,39	98,92	100,00
	×4	99,51	98,96	98,96

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	94,80	97,41	94,96
	×1	94,81	96,97	91,43
	×2	98,27	96,26	98,10
	×3	97,42	99,14	91,27
	×4	98,28	97,22	93,75

- **<u>Κατηγορία υδρολασών: 3.2.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	86,67	100,00	81,82
	×1	85,00	100,00	72,73
	×2	96,77	100,00	83,33
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	80,00	78,57	100,00
	×1	80,00	100,00	63,64
	×2	90,32	100,00	50,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	73,33	73,33	100,00
	×1	75,00	100,00	54,55
	×2	83,87	100,00	16,67
	×3	92,86	100,00	66,67
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	73,33	73,33	100,00
	×1	85,00	100,00	72,73
	×2	90,32	100,00	50,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

- **<u>Κατηγορία υδρολασών: 3.4.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	97,73	100,00	96,97
	×1	94,83	100,00	90,32
	×2	96,55	100,00	90,32
	×3	100,00	100,00	100,00
	×4	97,93	100,00	88,89

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	81,82	80,49	100,00
	×1	94,83	100,00	90,32
	×2	96,55	100,00	90,32
	×3	100,00	100,00	100,00
	×4	97,93	100,00	88,89

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	79,55	100,00	72,73
	×1	89,66	100,00	80,65
	×2	93,10	100,00	80,65
	×3	97,41	100,00	86,36
	×4	95,86	100,00	77,78

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	90,91	100,00	87,88
	×1	93,10	100,00	87,10
	×2	95,40	100,00	87,10
	×3	100,00	100,00	100,00
	×4	97,93	100,00	88,89

- **<u>Κατηγορία υδρολασών: 3.5.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	97,37	100,00	96,15
	×1	95,05	97,92	92,16
	×2	97,35	97,87	93,88
	×3	98,51	98,04	96,15
	×4	99,20	95,83	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	97,37	100,00	96,15
	×1	81,19	100,00	62,75
	×2	98,68	100,00	95,92
	×3	98,51	98,04	96,15
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	94,74	100,00	92,31
	×1	93,07	100,00	86,27
	×2	98,01	100,00	93,88
	×3	97,01	100,00	88,46
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	97,87	100,00	95,74
	×3	98,48	98,04	96,15
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	96,05	100,00	94,23
	×1	96,04	100,00	92,16
	×2	98,01	97,92	95,92
	×3	98,51	98,04	96,15
	×4	99,60	97,87	100,00

- **<u>Κατηγορία υδρολασών: 3.6.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	94,79	100,00	91,94
	×1	82,68	76,34	100,00
	×2	99,48	100,00	98,57
	×3	98,82	95,31	100,00
	×4	98,74	100,00	93,22

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	73,96	71,76	98,39
	×1	95,28	98,51	92,96
	×2	99,48	100,00	98,57
	×3	99,61	100,00	98,36
	×4	98,74	100,00	93,22

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	93,75	100,00	90,32
	×1	93,70	100,00	88,73
	×2	98,95	100,00	97,14
	×3	98,43	96,72	96,72
	×4	96,86	98,04	84,75

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	99,43	100,00	97,56

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	94,79	100,00	91,94
	×1	88,98	100,00	80,28
	×2	99,48	100,00	98,57
	×3	99,61	98,39	100,00
	×4	98,74	98,25	94,92

Τα παραπάνω αποτελέσματα δείχνουν, στο σύνολό τους, πως οι δυο κλάσεις στις πληροφορίες εκπαίδευσης, ανεξάρτητα από το είδος των υδρολασών, είναι καλά χωρισμένες από μόνες τους. Το γεγονός αυτό ήταν αναμενόμενο, καθώς οι PFAM κωδικοί είναι χαρακτηριστικά πρωτεϊνικών οικογενειών. Οι αλγόριθμοι εμφανίζουν πολύ υψηλή ευστοχία (100%), δηλαδή έχουν την ικανότητα να ταξινομούν τις μη υδρολάσες σωστά ($FP = 0$). Όμως τα ποσοστά επιστροφής δεν είναι εξίσου υψηλά. Οι αλγόριθμοι δεν έχουν υψηλή ευαισθησία και ταξινομούν ορισμένες υδρολάσες ως μη υδρολάσες (σφάλμα τύπου 1, $FN > 0$). Το γεγονός αυτό, ίσως οφείλεται σε υδρολάσες που χαρακτηρίζονται από πολλούς PFAM κωδικούς και καταλύουν περισσότερες από μια αντιδράσεις.

Εξετάζοντας τα αποτελέσματα για την κατηγορία 3.1.-.-, η οποία αποτελείται από 392 αλληλουχίες, διαπιστώνεται μια μικρή αυξητική τάση και σταθεροποίηση γύρω από συγκεκριμένες τιμές, για την ακρίβειας, την ευστοχία και την επιστροφή, με αύξηση του μεγέθους της πληροφορίας εκπαίδευσης. Η συμπεριφορά αυτή εμφανίζεται σε μικρότερο βαθμό στις κατηγορίες 3.6.-.- (213 αλληλουχίες) και 3.5.-.- (168 αλληλουχίες). Στις κατηγορίες 3.4.-.- (97 αλληλουχίες) και 3.2.-.- (36 αλληλουχίες) παρατηρούνται μεγάλες αποκλίσεις μεταξύ των διαφορετικών μεγεθών των πληροφοριών εκπαίδευσης. Η συμπεριφορά αυτή οφείλεται στο ότι ένα λάθος στην ταξινόμηση έχει μεγάλο αρνητικό αντίκτυπο στις τιμές της ακρίβειας, ευστοχίας και επιστροφής.

Επιπλέον, σε όλες τις κατηγορίες υδρολασών, στους αλγόριθμους Perceptron, Μηχανές Διανυσμάτων Υποστήριξης και Δέντρα Απόφασης διαπιστώνεται ότι το μέγεθος των αρνητικών παραδειγμάτων πρέπει να είναι τουλάχιστον το διπλάσιο από αυτό των θετικών. Οι πληροφορίες εκπαίδευσης x05 και x1 ίσως να μην περιέχουν αρκετή ποσότητα πληροφορίας ώστε να γίνει σωστά η μάθηση. Οι αλγόριθμοι Perceptron, Μηχανές Διανυσμάτων Υποστήριξης, Δέντρα Απόφασης και κ-Κοντινότερος Γείτονας εμφανίζουν υψηλές τιμές για μεγάλες πληροφορίες εκπαίδευσης. Όμως τα ποσοστά ευαισθησίας κοντά στο 90% δείχνουν ότι μια υδρολάση με πολύπλοκα χαρακτηριστικά ίσως ταξινομηθεί λάθος.

Η απόδοση του Perceptron εξαρτάται από την παράμετρο rounds και καθόλου από το learning_rate. Γενικά η ακρίβεια φτάνει στην μέγιστη τιμή της

για $rounds \geq 20$. Η απόδοση των Μηχανών Διανυσμάτων Υποστήριξης, μεγιστοποιείται όταν χρησιμοποιείται ο πυρήνας dot, ο πυρήνας polynominal (kernel_degree=1) και ορισμένες φορές ο πυρήνας radial (kernel_gamma=0.01). Συγχρόνως όσο μεγαλύτερη είναι η πληροφορία εκπαίδευσης τόσο καλύτερα αποτελέσματα δίνουν υψηλές τιμές του C, δηλαδή υπάρχει θόρυβος στην πληροφορία και οι κλάσεις διαχωρίζονται με αυστηρά περιθώρια. Παρόμοια συμπεριφορά έχει και κ-Κοντινότερος Γείτονας σε ό,τι αφορά τις συναρτήσεις πυρήνα. Επιπλέον η ακρίβεια μεγιστοποιείται για k=1 και σε μικρές πληροφορίες εκπαίδευσης για $k = \{1,4\}$. Τέλος τα Δέντρας Απόφασης εμφανίζουν μέγιστη ακρίβεια κυρίως με κριτήριο διακοπής το gini_index ή accuracy και συγχρόνως $maximal_depth \geq 60$ για μεγάλες πληροφορίες εκπαίδευσης (της κατηγορίας 3.1.-.-) και $maximal_depth \geq 20$ για μικρότερες πληροφορίες.

Ο αλγόριθμος που εμφανίζει τα καλύτερα αποτελέσματα είναι ο Μπεϋζιανός Ταξινομητής Πυρήνα. Σε ορισμένες κατηγορίες υδρολασών οι τιμές ακρίβειας, ευστοχίας και επιστροφής είναι 100%, ανεξάρτητα από το μέγεθος της πληροφορίας εκπαίδευσης. Η ακρίβειά του μεγιστοποιείται για $estimation_mode = greedy$ και $minimum_bandwidth = \{0.1,0.5,1\}$. Γενικά όσο αυξάνεται η παράμετρος $minimum_bandwidth$ μειώνεται η ακρίβεια του αλγόριθμου. Όμως η επιλογή greedy παρουσιάζει ένα σημαντικό πρόβλημα. Κατά την ταξινόμηση με $minimum_bandwidth = \{0.1,0.5,1\}$ και σε μικρά μεγέθη πληροφορίας εκπαίδευσης, ένα ποσοστό των παραδειγμάτων δεν ταξινομείται. Ο ταξινομητής αδυνατεί να ταξινομήσει κυρίως τις μη υδρολάσες. Η συμπεριφορά αυτή βελτιώνεται όσο αυξάνεται το μέγεθος της πληροφορίας εκπαίδευσης και στο μέγεθος x4 ταξινομούνται όλες σχεδόν οι υδρολάσες. Αυτό σημαίνει πως η επιλογή greedy απαιτεί μεγάλα μεγέθη πληροφορίας εκπαίδευσης, ώστε να μειωθούν οι πιθανότητες αδυναμίας ταξινόμησης ενός παραδείγματος.

Για την ενότητα της αξιοποίησης των αλγόριθμων μηχανικής μάθησης σε πληροφορία εκπαίδευσης PFAM χαρακτηριστικών επιλέχθηκε ο Μπεϋζιανός Ταξινομητής Πυρήνα με παραμέτρους $estimation_mode = greedy$ και $minimum_bandwidth = 5$.

Στη συνέχεια παρουσιάζονται τα αποτελέσματα για την πληροφορία εκπαίδευσης PROFEAT χαρακτηριστικών. Όλες οι πληροφορίες αποτελούνταν από 1024 χαρακτηριστικά (είναι ο αριθμός των PROFEAT χαρακτηριστικών που υπολογίζει το λογισμικό για κάθε αλληλουχία).

- **<u>Κατηγορία υδρολασών: 3.1.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	98,84	99,12	99,12
	×1	96,97	100,00	93,86
	×2	96,55	100,00	89,74
	×3	26,82	26,82	100,00
	×4	19,76	19,76	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	98,26	98,25	99,12
	×1	50,65	-	0,00
	×2	66,38	-	0,00
	×3	73,18	-	0,00
	×4	80,24	-	0,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	100,00	100,00	100,00
	×1	99,57	100,00	99,12
	×2	99,71	100,00	99,15
	×3	99,79	100,00	99,20
	×4	99,66	99,13	99,13

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	98,25	100,00	96,49
	×2	97,69	100,00	93,10
	×3	97,20	100,00	89,52
	×4	98,97	100,00	94,78

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	98,84	99,12	99,12
	×1	98,70	99,12	98,25
	×2	99,71	100,00	99,15
	×3	98,50	100,00	94,40
	×4	99,48	100,00	97,39

- **<u>Κατηγορία υδρολασών: 3.2.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	96,77	91,67	100,00
	×3	100,00	100,00	100,00
	×4	96,15	80,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

- **<u>Κατηγορία υδρολασών: 3.4.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	97,67	96,30	100,00
	×1	55,17	55,17	100,00
	×2	70,11	-	0,00
	×3	75,00	-	0,00
	×4	78,62	-	0,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	100,00	100,00	100,00
	×4	100,00	100,00	100,00

- **<u>Κατηγορία υδρολασών: 3.5.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	99,34	100,00	97,96
	×3	99,50	98,21	100,00
	×4	99,60	100,00	98,23

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	64,00	64,00	100,00
	×1	99,00	98,08	100,00
	×2	67,55	-	0,00
	×3	72,77	-	0,00
	×4	77,29	-	0,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	99,50	98,21	100,00
	×4	100,00	100,00	100,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	100,00	100,00	100,00
	×1	100,00	100,00	100,00
	×2	100,00	100,00	100,00
	×3	99,01	98,18	98,18
	×4	99,60	100,00	98,25

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	98,67	97,96	100,00
	×1	100,00	100,00	100,00
	×2	99,34	98,00	100,00
	×3	98,51	96,43	98,18
	×4	100,00	100,00	100,00

- **<u>Κατηγορία υδρολασών: 3.6.-.-</u>**

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Perceptron	×05	97,89	98,55	98,55
	×1	98,43	98,51	98,51
	×2	98,43	97,06	98,51
	×3	99,61	100,00	98,33
	×4	99,37	100,00	96,83

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Support Vector Machines	×05	72,63	72,63	100,00
	×1	52,76	52,76	100,00
	×2	98,43	97,06	98,51
	×3	76,47	-	0,00
	×4	80,19	-	0,00

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Decision Trees	×05	98,95	100,00	98,57
	×1	100,00	100,00	100,00
	×2	98,43	98,48	97,01
	×3	100,00	100,00	100,00
	×4	99,37	100,00	96,83

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
Naive Bayes Kernel	×05	95,79	100,00	94,20
	×1	96,83	100,00	94,03
	×2	96,86	98,41	92,54
	×3	99,61	100,00	98,33
	×4	98,11	100,00	90,48

		Accuracy (Ακρίβεια)	Precision (Ευστοχία)	Recall (Επιστροφή)
k-Nearest Neighbor	×05	98,95	100,00	98,55
	×1	99,21	100,00	98,51
	×2	97,38	98,44	94,03
	×3	100,00	100,00	100,00
	×4	99,06	100,00	95,24

Αρχικά διαπιστώνεται πως το μέγεθος της πληροφορίας εκπαίδευσης δεν έχει καθοριστικό ρόλο στην απόδοση των αλγόριθμων. Με εξαίρεση τον αλγόριθμο Perceptron, ο οποίος στην κατηγορία 3.1.-.- και στις πληροφορίες εκπαίδευσης x3, x4 αντιμετωπίζει πρόβλημα και τις Μηχανές Διανυσμάτων Υποστήριξης, οι οποίες δεν λειτούργησαν σωστά, οι υπόλοιποι αλγόριθμοι πετυχαίνουν πολύ υψηλά ποσοστά για κάθε μέγεθος πληροφορίας. Σε σύγκριση των δυο ειδών πληροφοριών εκπαίδευσης (PROFEAT και PFAM) μπορούμε να πούμε πως τα αποτελέσματα είναι καλύτερα με τα PROFEAT χαρακτηριστικά γιατί 1) είναι περισσότερα σε αριθμό από τα PFAM και 2) επειδή κάθε χαρακτηριστικό είναι συνεχής μεταβλητή και λαμβάνει διαφορετικές τιμές σε κάθε πρότυπο, η πληροφορία είναι αρκετά πιο λεπτομερής. Όμως ο μεγάλος αριθμός χαρακτηριστικών (1024) δημιουργεί προβλήματα στις Μηχανές Διανυσμάτων υποστήριξης, στον Perceptron και για τη μάθηση απαιτείται περισσότερος χρόνος απ' αυτόν της μάθησης με PFAM χαρακτηριστικά.

Η απόδοση του αλγόριθμου Perceptron μεγιστοποιείται για rounds>5, αλλά σε μικρές πληροφορίες εκπαίδευσης υπήρξαν αλγόριθμοι που δεν χρειάστηκαν γύροι μάθησης (rounds=1) για να μεγιστοποιηθεί η απόδοσή τους, επαληθεύοντας ότι η πληροφορία των PROFEAT χαρακτηριστικών διαχωρίζει πολύ ικανοποιητικά τις δυο κλάσεις. Ο Μπεϋζιανός Ταξινομητής Πυρήνα, όπως και στα PFAM χαρακτηριστικά, λειτουργεί καλύτερα για $\text{estimation_mode} = \text{greedy}$ και $\text{minimum_bandwidth} = [0.1,2]$. Επιπλέον δεν παρουσιάστηκε το πρόβλημα μη ταξινόμησης παραδειγμάτων. Το γεγονός αυτό ενισχύει την προτίμηση της πληροφορίας εκπαίδευσης των PROFEAT χαρακτηριστικών. Ο κ-Κοντινότερος-Γείτονας μεγιστοποιεί τις αποδόσεις του με τις συναρτήσεις πυρήνα dot και polynominal (με

kernel_degree=1,2,4) και με k=4, αλλά ορισμένες φορές και με k=8 ή 12. Οι αποδόσεις των Δέντρων Απόφασης δείχνουν πως ο συγκεκριμένος αλγόριθμος λειτουργεί πολύ ικανοποιητικά. Μάλιστα, ανεξάρτητα από το κριτήριο διακοπής και τις άλλες παραμέτρους, τα Δέντρα Απόφασης έχουν πάντα απόδοση κοντά στο 100%. Για τον λόγο αυτό επιλέχτηκαν για την επόμενη ενότητα του τεχνικού μέρους.

2.2. Αξιοποίηση των Αλγόριθμων για το Διαχωρισμό Ενζύμων με Βάση τη Θερμοκρασία Δράσης τους

Τα αποτελέσματα του hmmer_process.pl για τις τρεις πληροφορίες εκαπίδευσης παρουσιάζονται στον παρακάτω πίνακα:

- **Κατηγορία υδρολασών: 3.1.-.-**

Αριθμός πρότυπων παραδειγμάτων			**PFAM κωδικοί (χαρακτηριστικά)**			
hyper	meso	σύνολο	hyper	meso	σύνολο	πληροφορίας εκπαίδευσης
99	212	311	55	143	198	166
thermo	meso	σύνολο	thermo	meso	σύνολο	πληροφορίας εκπαίδευσης
99	212	311	74	143	217	165

- **Κατηγορία υδρολασών: 3.6.-.-**

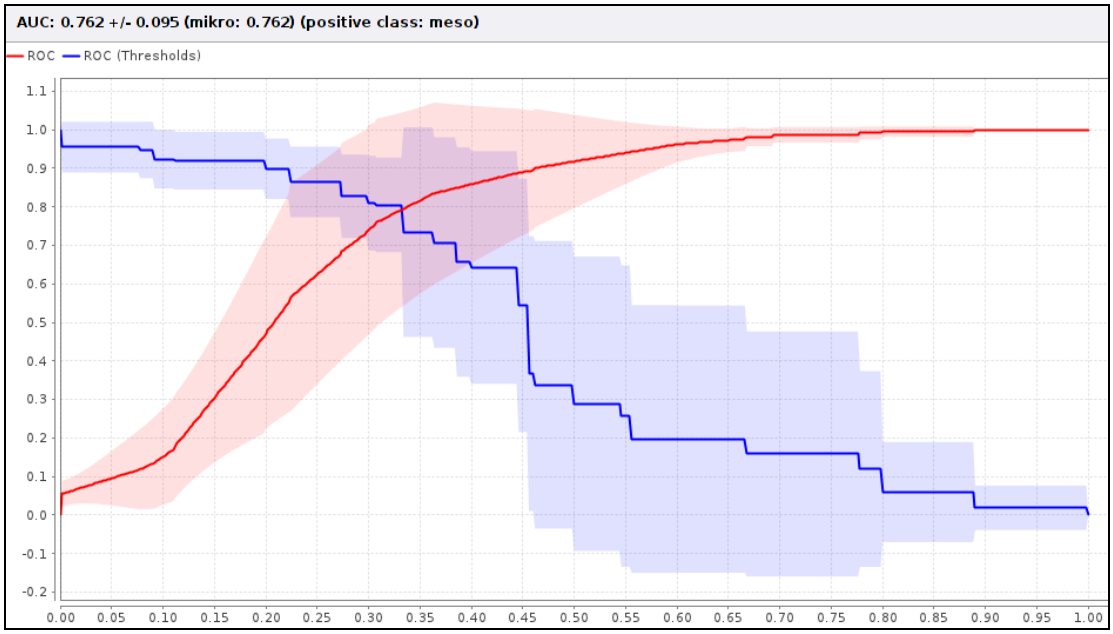
Αριθμός πρότυπων παραδειγμάτων			**PFAM κωδικοί (χαρακτηριστικά)**			
hyper	meso	σύνολο	hyper	meso	σύνολο	πληροφορίας εκπαίδευσης
94	199	293	95	117	212	144

Όπως προκύπτει, οι δυο διαφορετικές κλάσεις, σε κάθε πληροφορία εκπαίδευσης, είχαν πολλά όμοια PFAM χαρακτηριστικά, επομένως ο ταξινομητής είχε να διαχωρίσει πιο σύνθετη πληροφορία απ’ ότι στην πρώτη

ενότητα, αυτή της μελέτης. Υπενθυμίζεται πως για τα PFAM χαρακτηριστικά χρησιμοποιήθηκε ο Μπεϋζιανός Ταξινομητής Πυρήνα και για τα PROFEAT τα Δέντρα Απόφασης. Παρακάτω παρουσιάζονται τα ποσοστά, τα διαγράμματα ROC και οι τιμές AUC των ταξινομητών για τα PFAM χαρακτηριστικά:

- **Κατηγορία υδρολασών: 3.1.-.-, κλάσεις hyper-meso**

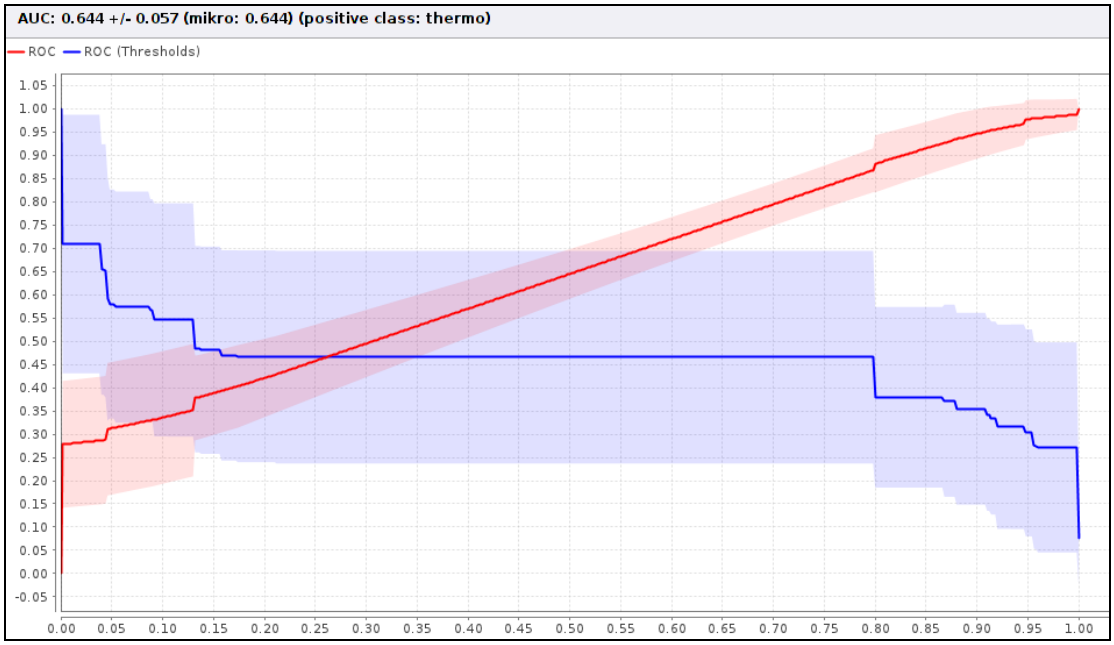
Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
hyper	91,04	62,89	86,27
meso	84,94	97,13	



Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [PF01867.11], [PF09827.4], [PF04414.7], [PF01981.11], [PF01351.13], [PF01876.11], [PF01339.12], [PF04032.11], [PF00867.13], [PF01868.11], [PF01939.11], [PF03463.10], [PF01900.14], [PF01974.12], [PF01975.12], [PF01138.16], [PF00459.20].

- **Κατηγορία υδρολασών: 3.1.-.-, κλάσεις thermo-meso**

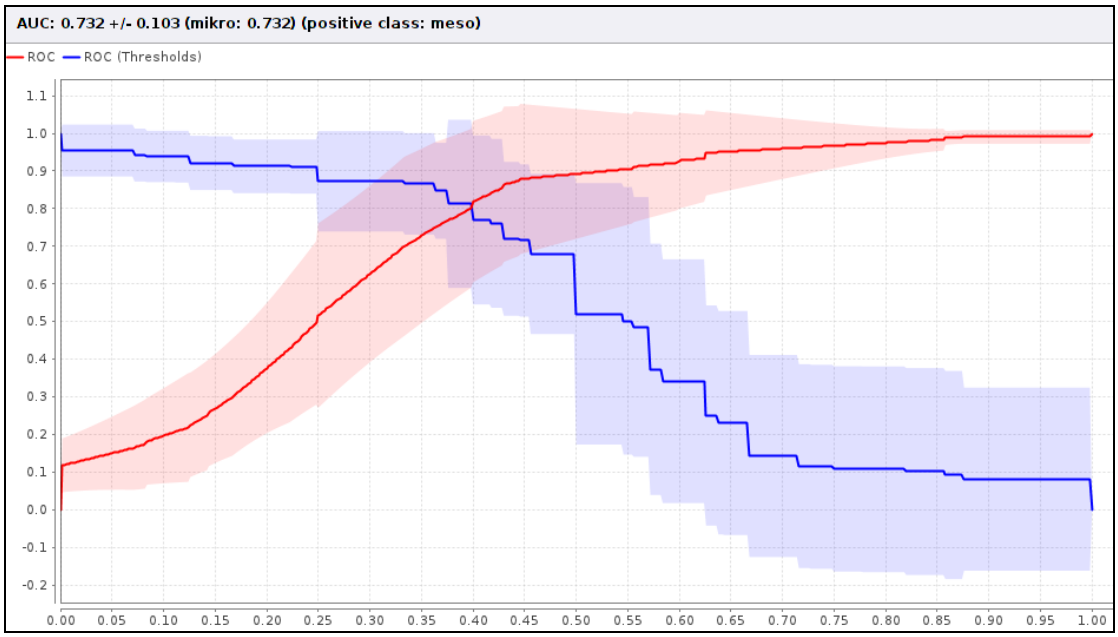
Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
thermo	83,87	26,8	75,23
meso	74,18	97,60	



Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [PF02601.10], [PF01336.20], [PF13361.1], [PF01351.13], [PF02075.12], [PF03463.10], [PF09194.5], [PF01138.16], [PF01868.11], [PF01974.12], [PF04032.11].

- **Κατηγορία υδρολασών: 3.6.-.-, κλάσεις hyper-meso**

Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
hyper	90,74	52,13	82,88
meso	81,09	97,47	

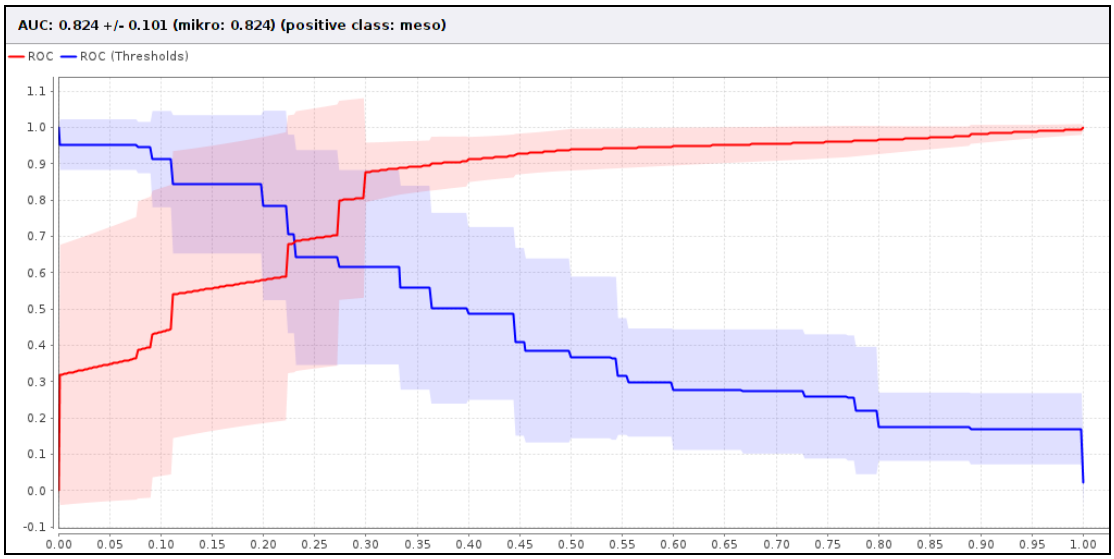


Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [PF03266.10], [PF03308.11], [PF01131.15], [PF01751.17], [PF00270.24], [PF00692.14], [PF00006.20], [PF02492.14], [PF02673.13], [PF00719.14], [PF02374.10], [PF13191.1], [PF01931.13], [PF10662.4], [PF12848.2].

Γενικά παρατηρείται πως οι πληροφορίες εκπαίδευσης δεν διαχωρίζονται εύκολα. Οι ταξινομητές με βάση τα ποσοστά επιστροφής και ευστοχίας και τις τιμές AUC δείχνουν πως δεν πραγματοποιείται σωστά η μάθηση και το πιο συχνό τους σφάλμα είναι η ταξινόμηση μιας υδρολάσης hyper ή thermo στις meso. Τα μεγαλύτερα ποσοστά στις δυο πληροφορίες εκπαίδευσης hyper-meso σε σχέση με την κατηγορία thermo-meso δείχνουν πως όσο αυξάνεται η θερμοκρασιακή διαφορά των υδρολασών, τόσο καλύτερα αποτελέσματα έχει ο ταξινομητής. Δηλαδή υπάρχουν χαρακτηριστικά PFAM που διαφοροποιούνται ανάλογα με τη θερμοκρασία αλλά ίσως δεν παρέχουν αρκετή πληροφορία ή ο ταξινομητής δεν εκπαιδεύεται στο βέλτιστο βαθμό. Στη συνέχεια παρουσιάζονται τα αποτελέσματα για τα PROFEAT χαρακτηριστικά.

- **Κατηγορία υδρολασών: 3.1.-.-, κλάσεις hyper-meso**

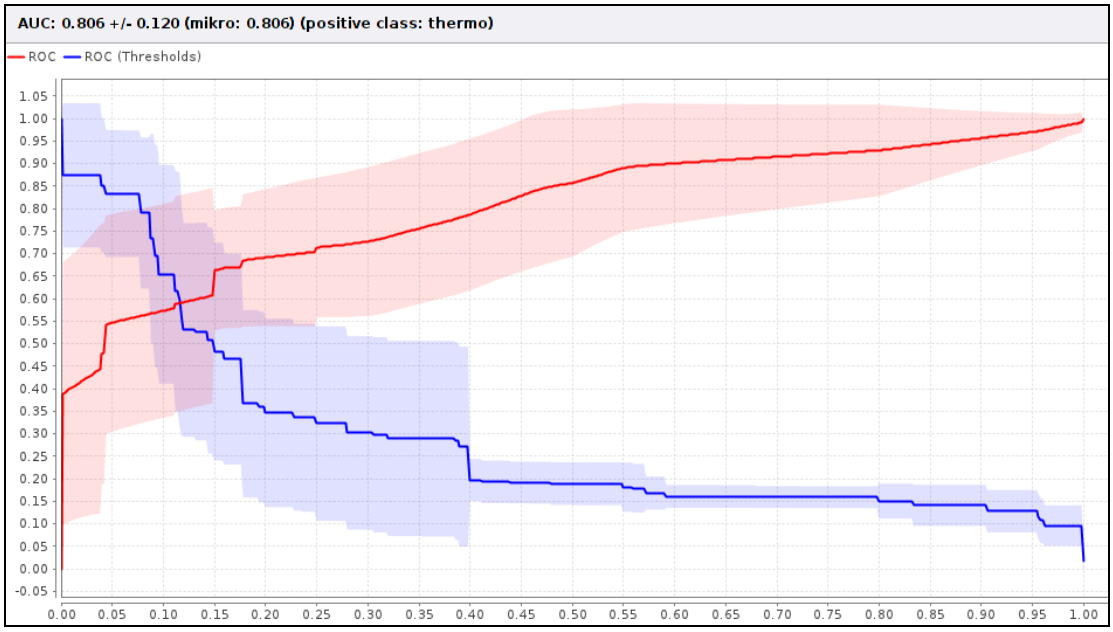
Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
hyper	78,35	78,35	86,27
meso	89,95	89,95	



Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [G2.1.1.76], [G2.1.1.79], [G2.1.1.82], [G2.1.1.97], [G2.1.1.115], [G2.1.1.121], [G2.1.1.131], [G2.1.1.150], [G2.1.1.212], [G2.1.1.245], [G2.1.1.267], [G2.1.1.384], [G7.1.1.6].

- **Κατηγορία υδρολασών: 3.1.-.-, κλάσεις thermo-meso**

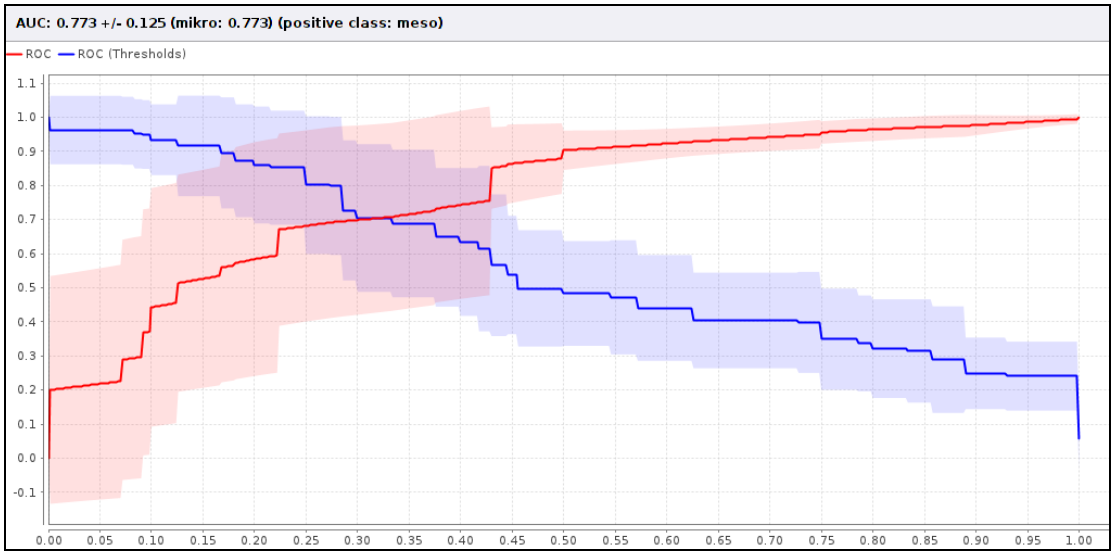
Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
thermo	84,72	62,89	84,64
meso	84,61	94,74	



Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [G1.1.1.15], [G2.1.1.11], [G2.1.1.52], [G2.1.1.139], [G2.1.1.285], [G2.1.1.312], [G2.1.1.315], [G2.1.1.347].

- **Κατηγορία υδρολασών: 3.6.-.-, κλάσεις hyper-meso**

Κλάση	Precision (Ευστοχία)	Recall (Επιστροφή)	Accuracy (Ακρίβεια)
hyper	72,23	71,28	82,19
meso	86,50	87,37	



Τα χαρακτηριστικά που επέλεξε ο ταξινομητής είναι: [G2.1.1.5], [G2.1.1.21], [G2.1.1.179], [G2.1.1.211], [G2.1.1.220], [G2.1.1.228], [G2.1.1.254], [G2.1.1.300], [G5.1.2.14].

Οι αποδόσεις των ταξινομητών με τα PROFEAT χαρακτηριστικά είναι παρόμοιες με αυτές των PFAM, ενώ παρατηρούνται αυξημένες οι τιμές των AUC. Όμως δεν παρατηρείται μεγάλη επιστροφή των μεσόφιλων ενζύμων και δεν υπάρχει διαφορά στα αποτελέσματα hyper-meso και thermo-meso, όπως συμβαίνει με τα PFAM. Η πληροφορία που παρέχουν τα PROFEAT χαρακτηριστικά διαχωρίζει τα θερμοανθεκτικά ένζυμα από τα μεσόφιλα ως ένα ποσοστό, χωρίς αυτό να αυξάνεται όσο αυξάνεται η θερμοκρασία των θεμροανθεκτικών ενζύμων. Τέλος παρατηρείται πως οι διαδικασίες επιλογής χαρακτηριστικών καταλήγουν στην επιλογή χαρακτηριστικών κυρίως της κατηγορίας G2, δηλαδή ποσοστά διπεπτιδίων.

ΠΑΡΑΡΤΗΜΑ 1ης ΕΝΟΤΗΤΑΣ

- **create_MySQL_schema.pl**

```
#!/usr/bin/perl
use strict;
use DBI;

#Arguments-input elements
my $db = $ARGV[0];
my $user_name = $ARGV[1];
my $password = $ARGV[2];

my $dbh = DBI->connect("DBI:mysql:mysql:localhost", $user_name, $password); #connection to
MySQL
my $schema = "CREATE SCHEMA $db" or die "could not created";
my $sth = $dbh->prepare($schema); $sth->execute();
```

- **annotations_parser.pl**

```
#!/usr/bin/perl
use DBI;
use strict;

#Arguments-input elements
my $annotations_file=$ARGV[0];
my $major_name = $ARGV[1];
my $db=$ARGV[2];
my $user_name = $ARGV[3];
my $password = $ARGV[4];

my $dsn = "DBI:mysql:$db:localhost";
my $dbh = DBI->connect($dsn, $user_name, $password); #connection to MySQL

my $draft_name = "draft_$major_name";
table($draft_name); #execution of “table” subroutine for the variable $draft_name
table($major_name); #execution of “table” subroutine for the variable $major_name

open (FILE, "<$annotations_file");
while (my $line = <FILE>) {
        if ($line !~ m/^!/) { #if line does not start with !
                chomp $line; #chomp function removes any newline character from the end line
                my @fields = (split /\t/, $line); #line is separated in fields with tab
                my $db_name = $fields[0];
                my $object_acc = $fields[1];
                my $object_symbol = $fields[2];
                $object_symbol =~ s/\'//g;
                my $qualifier = $fields[3];
                my $GO_id = $fields[4];
                my $db_ref = $fields[5];
                my $evidence = $fields[6];
                my $additional_id = $fields[7];
                my $GO_aspect = $fields[8];
                my $object_name = $fields[9];
                $object_name =~ s/\'//g;
                my $synonym = $fields[10];
                $synonym =~ s/\'//g;
                my $object_type = $fields[11];
                my $taxon_id = $fields[12];
                $taxon_id =~ s/taxon://g;
                my $date = $fields[13];
                $date =~ s[(....)(..)(..)][$1-$2-$3];
                my $assign = $fields[14];
```

```
                my $annotation_extension = $fields[15];
                my $gene_product_form_id = $fields[16];

                my $query="insert into $draft_name (db_name, object_acc, object_symbol, qualifier,
GO_id, db_reference, evidence, additional_id, GO_aspect, object_name, synonym, object_type,
taxon_id, date, assigned_by, annotation_extension, gene_product_form_id) values
(\'$db_name\',\'$object_acc\',\'$object_symbol\',\'$qualifier\',\'$GO_id\',\'$db_ref\',\'$evidence\',\'$addition
al_id\',\'$GO_aspect\',\'$object_name\',\'$synonym\',\'$object_type\',\'$taxon_id\',\'$date\',\'$assign\',\'$an
notation_extension\',\'$gene_product_form_id\');" or die "problem in loading the annotations at
$draft_name"; #loading each variable in the respective column
                my $sth = $dbh->prepare($query);$sth->execute();
        }
}close FILE;

my $query = "SELECT * FROM $draft_name GROUP BY object_acc, GO_id;"; #Select all the entries
from $draft_name with distinct couple object_acc-GO_id
my $sth = $dbh->prepare($query); $sth->execute();

while (my @line = $sth->fetchrow_array()) {
        my $query="insert into $major_name (db_name, object_acc, object_symbol, qualifier, GO_id,
db_reference, evidence, additional_id, GO_aspect, object_name, synonym, object_type, taxon_id, date,
assigned_by, annotation_extension, gene_product_form_id) values
(\'$line[1]\',\'$line[2]\',\'$line[3]\',\'$line[4]\',\'$line[5]\',
\'$line[6]\',\'$line[7]\',\'$line[8]\',\'$line[9]\',\'$line[10]\',\'$line[11]\',\'$line[12]\',\'$line[13]\',\'$line[14]\',\'$line[
15]\',\'$line[16]\',\'$line[17]\');";
        my $sth = $dbh->prepare($query);$sth->execute();
        } #load the above selection in the new table

my $drop = "DROP TABLE IF EXISTS $draft_name;"; #drop the draft table
my $sth = $dbh->prepare($drop); $sth->execute();

sub table { #"table" subroutine – creation of MySQL table "$name"

        (my $name) = @_;

        my $sql_table1 = "DROP TABLE if exists $name;";
        my $sql_table2 = "CREATE TABLE $name(
                ind INTEGER NOT NULL AUTO_INCREMENT,
                db_name VARCHAR(30),
                object_acc VARCHAR(15),
                object_symbol VARCHAR(30),
                qualifier TEXT,
                GO_id VARCHAR(20),
                db_reference VARCHAR(30),
                evidence VARCHAR(5),
                additional_id VARCHAR(30),
                GO_aspect ENUM('P','F','C'),
                object_name VARCHAR(100),
                synonym VARCHAR(100),
                object_type VARCHAR(15),
                taxon_id VARCHAR(15),
                date VARCHAR(12),
                assigned_by VARCHAR(30),
                annotation_extension VARCHAR(100) NOT NULL,
                gene_product_form_id VARCHAR(30) NOT NULL,
                PRIMARY KEY(ind)
        );";
        my $sth = $dbh->prepare($sql_table1); $sth->execute();
        my $sth = $dbh->prepare($sql_table2); $sth->execute();
}
```

- **ec2go.pl**

```
#!/usr/bin/perl/
use strict;
use DBI;
```

```
my $file_in = $ARGV[0];
my $db = $ARGV[1];
my $user_name = $ARGV[2];
my $password = $ARGV[3];

my $dbh = DBI->connect("DBI:mysql:$db:localhost", $user_name, $password); #connection to MySQL
my $table_1 = "DROP TABLE IF EXISTS ec2go;";
my $table_2 = "CREATE TABLE ec2go (
        ind INTEGER NOT NULL AUTO_INCREMENT,
        EC_number VARCHAR(15),
        GO_id VARCHAR(15),
        PRIMARY KEY (ind)
);";
my $sth = $dbh->prepare($table_1);$sth->execute();
my $sth = $dbh->prepare($table_2);$sth->execute();

open (FILE, "<$file_in") or die "could not open file";
while (my $line = <FILE>){
        chomp $line;
        if ($line =~ m/^EC/){ #if line start with EC
                my @fields = (split /\>/,$line); #line is separated in fields with >
                my $ec = $fields[0]; #fields[1] is the EC number
                $ec =~ s/[EC:\s]//g; #delete from EC number the “EC:” and every whitespace
                        my @fields2 = (split /\;/, $fields[1]); fields[1] is separated in fields with ;
                        my $go = $fields2[1]; #fields2[1] is the GO id
                        $go =~ s/[\s]//g; #delete from GO id every whitespace character
        my $query = "INSERT INTO  ec2go (EC_number, GO_id) values (\'$ec\', \'$go\');"; #loading of
data in ec2go table
        my $sth = $dbh->prepare($query);$sth->execute();
        }
} close (FILE);
```

- **distinct_acc_numbers.pl**

```
#!/usr/bin/perl
use strict;
use DBI;

my $file_out=$ARGV[0];
my $db=$ARGV[1];
my $user_name=$ARGV[2];
my $password=$ARGV[3];

my $dbh=DBI->connect("DBI:mysql:$db:localhost", $user_name, $password) or die; #connection to
MySQL

open (FILE, ">$file_out");
my $sql1 = "SELECT DISTINCT(object_acc) FROM annotations;";
        my $acc= $dbh->selectall_arrayref($sql1,
        { Slice => {} }
        ); # Slice fetches an array of rows (@acc) where each row is stored as a hash { object_acc}
        foreach my $acc (@$acc) {
                print FILE $acc->{object_acc}, "\n";
        }close (FILE);
```

- **filter_length.pl**

```
#!/usr/bin/perl/
use strict;
use Bio::SeqIO;

my $file_in = $ARGV[0];

my $file_1 = "set_1_85";
my $file_2 = "set_86_300";
```

```
my $file_3 = "set_300+";
open (FILE_1, ">$file_1");
open (FILE_2, ">$file_2");
open (FILE_3, ">$file_3");

my $seqio_obj = Bio::SeqIO->new(-file => $file_in, -format => "fasta" ); #the input file (fasta format) is being transferred to object “$seqio_obj”
while (my $seq_obj = $seqio_obj->next_seq){  #seq_obj is each seq (entry) in seqio_obj, mext_seq is the function to find the next entry
        my $seq_name = $seq_obj->id; #id is the function to find protein id
        my $seq = $seq_obj->seq; #seq is the function to find protein sequence
        my $seq_length = $seq_obj->length; #length is the function to find sequence length
        if ($seq_length<=85) {
                print FILE_1 ">$seq_name","_length=$seq_length\n",$seq,"\n";
        } elsif ($seq_length>85 && $seq_length<=300) {
                print FILE_2 ">$seq_name","_length=$seq_length\n",$seq,"\n";
        } else {
                print FILE_3 ">$seq_name","_length=$seq_length\n",$seq,"\n";
        }
}
```

- **blastp_for_GO.pl**

```
#!/usr/bin/perl
use strict;
use DBI;
use Bio::SearchIO;

my $file_in = $ARGV[0]; #file for blastp scanning
my $file_out = $ARGV[1]; #result file and MySQL table
my $blastp_db = $ARGV[2]; #database for blastp
my $evalue = $ARGV[3]; #evalue threshold, 0.00001
my $matrix = $ARGV[4]; #BLOSUM62, BLOSUM80, PAM250 etc...
my $db = $ARGV[5]; #mysql schema
my $user_name = $ARGV[6];
my $password = $ARGV[7];

open (FILE_PRO, ">process_for_$file_in") or die "cannot open process file";

##############STEP 1: Run blastp program
my $blastp_run = "blastp -query $file_in -db $blastp_db -out $file_out -evalue $evalue -matrix $matrix";
system ($blastp_run); #system function execute the above command line query
print FILE_PRO "-blastp scanning for $file_in terminated.\n\n";

###############STEP 2: Blastp results parser to MySQL table
my $dbh = DBI->connect("DBI:mysql:$db:localhost", $user_name, $password) or die "cannot connect to MySQL"; #connection to MySQL

my $table_blastp_1= "DROP TABLE IF EXISTS $file_out;";
my $table_blastp_2= "CREATE TABLE $file_out (
        ind INTEGER NOT NULL AUTO_INCREMENT,
        query_name VARCHAR(100),
        hit_name VARCHAR(100),
        hsp_length INTEGER,
        hsp_evalue REAL,
        hsp_percent_identity FLOAT,
        PRIMARY KEY(ind)
        );";
my $sth = $dbh->prepare($table_blastp_1);$sth->execute();
my $sth = $dbh->prepare($table_blastp_2);$sth->execute();

my $in=Bio::SearchIO->new(-file=>$file_out, -format=>"blast"); #the output file (blast format) is being transferred to object “$in”
while (my $result = $in -> next_result) {
        my $query_accession = $result->query_accession();
        my $query_name = $result->query_name();
```

```
	while (my $hit = $result->next_hit) {
		my $hit_name = $hit->name();
		while (my $hsp = $hit->next_hsp) {
			my $hsp_length = $hsp->length;
			my $hsp_evalue = $hsp->evalue;
			my $hsp_percent_identity=$hsp->percent_identity;
			my $query="insert into $file_out (query_name, hit_name, hsp_length,
hsp_evalue, hsp_percent_identity) values
(\'$query_name\',\'$hit_name\',\'$hsp_length\',\'$hsp_evalue\',\'$hsp_percent_identity\');";
			my $sth = $dbh->prepare($query);$sth->execute();
		}
	}
}print FILE_PRO "-The loading of blastp results terminated.\n\n";

###############STEP 3: Find GO-terms and creation of a draft table GO_$file_out
my $table_1= "DROP TABLE IF EXISTS GO_$file_out;";
my $table_2= "CREATE TABLE GO_$file_out (
	ind INTEGER NOT NULL AUTO_INCREMENT,
	query_name VARCHAR(100),
	hit_name VARCHAR(100),
	hsp_length INTEGER,
	hsp_evalue REAL,
	hsp_percent_identity FLOAT,
	GO_id VARCHAR(15),
	PRIMARY KEY(ind)
	);";
my $sth = $dbh->prepare($table_1);$sth->execute();
my $sth = $dbh->prepare($table_2);$sth->execute();

my $query = "SELECT a.*, b.GO_id FROM $file_out a, annotations b WHERE a.hit_name REGEXP
b.object_acc;"; #the query which correlates “$file_out” and “annotations” tables
my $sth = $dbh->prepare($query); $sth->execute();

while (my @line = $sth->fetchrow_array()) {
	my @array = ($line[1], $line[2], $line[3], $line[4], $line[5], $line[6]);
	my $query="insert into GO_$file_out (query_name, hit_name, hsp_length, hsp_evalue,
hsp_percent_identity, GO_id) values (\'$line[1]\',\'$line[2]\',\'$line[3]\',\'$line[4]\',\'$line[5]\', \'$line[6]\');";
	my $sth = $dbh->prepare($query);$sth->execute();
	print FILE_OUT "\n";
}

###############STEP 4: Find GO_ids-EC_numbers and print the final file
open (DRAFT, ">draft_$file_out.csv");
print DRAFT
"query_name,hit_name,hsp_length,hsp_evalue,hsp_percent_identity,GO_id,EC_number\n\n"; #the first
line of draft file
my $query = "SELECT * FROM GO_$file_out";
my $sth = $dbh->prepare($query); $sth->execute();
while (my @line = $sth->fetchrow_array()) {
	foreach my $line (@line[1..$#line]) {
		print DRAFT "$line,"; #print the data of GO_$file_out table to draft file
	}print DRAFT "no_EC_number\n"; #in seventh column print “no_EC_number”
}close DRAFT;

my %go_ec;
my $query = "SELECT * FROM ec2go;";
my $sth = $dbh->prepare($query); $sth->execute();
while (my @element = $sth->fetchrow_array()) {
	my $ec = $element[1];
	my $go = $element[2];
	$go_ec{$go} = $ec;
}

open (DRAFT, "<draft_$file_out.csv");
open (FILE_OUT, ">final_results_$file_out.csv");
print FILE_OUT
"query_name,hit_name,hsp_length,hsp_evalue,hsp_percent_identity,GO_id,EC_number\n\n";
```

```
while (my $line = <DRAFT>) {
	my @fields = (split /\,/, $line);
	if ($line =~ m/^ma/) { #all entries start with “ma”
		my $i=0;
		while ((my $go_id, my $ec) = each %go_ec) {
			if ($fields[5] eq $go_id) { #fields[5] is the GO id of each line
				print  FILE_OUT
"$fields[0],$fields[1],$fields[2],$fields[3],$fields[4],$fields[5],$ec\n"; # in seventh column the EC number
				$i=1;
			}
		}
		if ($i==0) {
			print FILE_OUT "$line"; #i=0 means that there is not an EC number
		}
	}
}close DRAFT;
my $drop = "DROP TABLE IF EXISTS GO_$file_out;";
my $sth = $dbh->prepare($drop); $sth->execute();

my $command = "rm draft_$file_out.csv"; #remove the draft file
system($command);
print FILE_PRO "-final_results_$file_out file has been created! Check the results...";
close FILE_OUT;
close FILE_PRO;
```

- **calculate_percentage.pl**

```
#!/usr/bin/perl/

#Arguments-input elements
my $file_in = $ARGV[0];
my $column = $ARGV[1]; #EC or GO
my $code = $ARGV[2]; # 3. , 3.1, 3.1.1.1

open (FILE, "<$file_in") or die "could not open file";
my $i=0;
my $j=0;

if ($column eq 'EC') {
	while (defined(my $line = <FILE>)){
		chomp $line;
		if ($line =~ m/^ma/) {
			my @fields = (split /\,/,$line);
			my $ec = $fields[6];
			$i=$i+1;
				if ($ec =~ m/^$code/) {
					$j=$j+1;
			}
		}
	}print "\ntotal hits : $i\nhits with ec_number '$code' : $j\n percentage = ",$j/$i,"\n";
}

if ($column eq 'GO') {
	while (defined(my $line = <FILE>)){
		chomp $line;
		if ($line =~ m/^ma/) {
			my @fields = (split /\,/,$line);
			my $go = $fields[5];
			$i=$i+1;
			if ($go =~ m/^$code/) {
				$j=$j+1;
			}
		}
	}print "\ntotal hits : $i\nhits with go_id '$code' : $j\nPercentage = ",$j/$i,"\n";
}
```

ΠΑΡΑΡΤΗΜΑ (Α) 2ης ΕΝΟΤΗΤΑΣ

- **parser_non.pl**

```
#!usr/bin/perl
use DBI;
use Bio::SeqIO;

#Arguments-input elements
my $file_in = $ARGV[0];
my $db=$ARGV[1];
my $db_user_name = $ARGV[2];
my $db_password = $ARGV[3];
my $table_name = $ARGV[4];

my $dsn = "DBI:mysql:$db:localhost";
my $dbh = DBI->connect($dsn, $db_user_name, $db_password); #connection to MySQL

my $sql_table_1= "DROP TABLE if exists $table_name;";
my $sql_table_2="CREATE TABLE $table_name(
        ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        cluster_name VARCHAR(100),
        PRIMARY KEY(ind),
        seq MEDIUMTEXT,
        INDEX(cluster_name)
);";
my $sth = $dbh->prepare($sql_table_1); $sth->execute();
my $sth = $dbh->prepare($sql_table_2); $sth->execute();

$seqio_obj = Bio::SeqIO->new(-file => $file_in, -format => "fasta" ); #the input file (fasta format) is being transferred to object “$seqio_obj”
while ($seq_obj = $seqio_obj->next_seq){
        my $seq_name = $seq_obj->id;
        my $seq = $seq_obj->seq;
        my $query="insert into $table_name (cluster_name, seq) values (\'$seq_name\',\'$seq\');";
        my $sth = $dbh->prepare($query); $sth->execute();
}

close FILE;
```

- **select_random.pl**

```
#!usr/bin/perl
use strict;
use DBI;

#Arguments-input elements
my $db = $ARGV[0];
my $user_name = $ARGV[1];
my $password = $ARGV[2];
my $table_name = $ARGV[3];
my $N = $ARGV[4];

my $dsn = "DBI:mysql:$db:localhost";
my $dbh = DBI->connect($dsn, $user_name, $password); #connection to MySQL

my $query = "SELECT * FROM $table_name ORDER BY RAND() LIMIT $N;"; #select $N sequences randomly
my $sth = $dbh->prepare($query); $sth->execute;

my $file = "random_".($table_name)[0]."_".$N;
open (FILE, ">$file");
while (my @line = $sth->fetchrow_array()) {
```

```
                my $name = $line[1];
                my $seq = $line[2];
                print FILE ">$name\n$seq\n";
        }close FILE;
```

- **hmmer_process.pl**

```
#!/usr/bin/perl
use strict;
use lib "$ENV{HOME}/my_packages/";
use Hmmer_Process;

#Arguments-input elements
my $db = $ARGV[0]; # MySQL database
my $user_name = $ARGV[1];
my $password = $ARGV[2];
my $table = $ARGV[3]; # MySQL preferable table name, hyd_3, hyd_3_1 etc.
my $hmmer_db = $ARGV[4]; # Pfam-A database
my $N = $ARGV[5]; # the number of input fasta files

my %hash;
my $i;

for ($i=6; $i<2*$N+6; $i=$i+2) {
        my $file_in = $ARGV[$i]; #fasta file
        my $class= $ARGV[$i+1]; #class of fasta file entries'
        $hash{$file_in} = $class;
}

my $dir_extra = "./process_archives";
my $command = "mkdir $dir_extra";# command line query: make the directory $dir_extra
system("$command");# system function execute the above command line query


Hmmer_Process::process_file; #use Hmmer_Process subroutine process_file
Hmmer_Process::mysql_table($table, $db, $user_name, $password); #use Hmmer_Process subroutine
mysql_table with input variables: $table, $username, $password

while ((my $file_in, my $class) = each %hash) {

        my $hmmer_input = "$file_in"; # $hmmer_input will be the hmmer input fasta file.
        my $hmmer_output = "hmmer_out_$file_in"; # hmmer-out file, it will contain hmmer results in
table-out format. The subroutine hmmer_parser will use this name for the MySQL table (without ".").
        Hmmer_Process::del_names ($file_in, $hmmer_input, $dir_extra);
        Hmmer_Process::run_hmmer ($hmmer_input, $hmmer_output, $hmmer_db, $dir_extra);
        Hmmer_Process::hmmer_parser ($hmmer_output, $class, $table, $db, $user_name,
$password, $dir_extra);
}
Hmmer_Process::pfam_tables ($table, $db, $user_name, $password);
Hmmer_Process::create_csv ($table, $db, $user_name, $password);
Hmmer_Process::pfam_for_prof ($table, $db, $user_name, $password);
```

- **Hmmer_Process.pm**

```
package Hmmer_Process;
use strict;
use DBI;

# This package has 8 subroutines: process file, del_names, run_hmmer, mysql_table, hmmer_parser,
pfam_tables, create_csv, pfam_for_prof

######## 1st SUBROUTINE: Creation of the process file.
########
######## When the process will have been terminated without problems, the file must have the
following format:
```

```
########  -The file seqs contains 920 entries.
########  -Hmmer scan terminated! Here the results:
########            * 1832 hits
########            * 298 distinct hits, i.e. 298 Pfam ids.
########
########  -The file seqs_non contains 1500 entries.
########  -The file seqs_non divided into 2 files of 1000-entries for the PROFEAT scan.
########  -Hmmer scan terminated! Here the results:
########            * 2738 hits
########            * 1046 distinct hits, i.e. 1046 Pfam ids.
########
#########  -Finally 1238 distinct hits.
#########
#########  -The csv file has been created.
#########
#########  ---------Hmmer process terminated!---------

sub process_file {

    open (FILE_PRO, ">process_file") or die "cannot open process file";
}

######## 2nd SUBROUTINE: Dropping of some extra features in fasta file:  Example:
>UniRef50_Q9SJF0 Serine/threonine-protein phosphatase BSL2 n=34 Tax=Embryophyta
RepID=BSL2_ARATH, retains only >UniRef50_Q9SJF0

sub del_names {

    (my $file_in, my $hmmer_input, my $dir) = @_;

    open (FILE_1, "<$file_in") or die "cannot open $file_in";
    open (FILE_2, ">$dir/$hmmer_input") or die "cannot open $dir/$hmmer_input";

    my $n=0; # this is the number of entries

    while (defined(my $line = <FILE_1>)) {
            chomp $line;
            if ($line =~ m/^\>/){ ### if line starts with >, print $line,"\n";
                    $n = $n+1;
                    my @fields = (split /\s{1}/,$line);
                    my $name = $fields[0];
                    print FILE_2 $name,"\n";
            } else {
                    print FILE_2 $line,"\n";
            }
    }print FILE_PRO "-The file $hmmer_input contains $n entries.\n";
    close FILE_1;
    close FILE_2;

    my $command_remove = "rm $file_in";
    system("$command_remove");
}

######## 3rd SUBROUTINE: HMMER scan will be executed: The scan will have "--cut_ga" filtering
(use profile's GA gathering cutoffs to set all thresholding) and the result will be written in a default-
format file and in a tableout-format file

sub run_hmmer {

    (my $hmmer_input, my $hmmer_output, my $hmmer_db, my $dir) = @_;

    my $command_run= "hmmscan -o $dir/default_$hmmer_output --tblout $dir/$hmmer_output --cut_ga
$hmmer_db $dir/$hmmer_input";
    system("$command_run"); # hmmer scanning

    my $command_delete = "rm $dir/default_$hmmer_output";
    system("$command_delete"); # delete the default_out file
    print FILE_PRO "-Hmmer scan terminated! Here the results:\n"
```

```
}

######## 4rth SUBROUTINE: Creation of the MySQL results table: This table will be filled with the
hmmer results by the hmmer_parser subroutine

sub mysql_table {

    (my $table, my $db, my $user_name, my $password) = @_;

my $dsn = "DBI:mysql:$db:localhost";
my $dbh  = DBI->connect($dsn, $user_name, $password) or die "cannot connect to MySQL";

    my $sql_table_1= "DROP TABLE if exists $table;";
    my $sql_table_2="CREATE TABLE $table(
            ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
            hit_name VARCHAR(100),
            hit_accession VARCHAR(50),
            query_name VARCHAR(100),
            evalue VARCHAR(50),
            score VARCHAR(50),
            description VARCHAR(500),
            ec_number VARCHAR(20),
            PRIMARY KEY(ind)
            );" or die "cannot create hmmer results table";
    my $sth = $dbh->prepare($sql_table_1);$sth->execute();
    my $sth = $dbh->prepare($sql_table_2);$sth->execute();
}

######## 5th SUBROUTINE: HMMER results parser: The necessary results will be retrieved from the
file_out and will be places in the MySQL-table.

sub hmmer_parser {

    (my $hmmer_output, my $class, my $table, my $db, my $user_name, my $password, my $dir) = @_;

    my $dsn = "DBI:mysql:$db:localhost";
    my $dbh  = DBI->connect($dsn, $user_name, $password) or die "cannot connect to MySQL";

    my $i = 0; # this variable will calculate the number of hits.
    open (FILE, "<$dir/$hmmer_output") or die "cannot open hmmer results file";
            while (defined(my $line=<FILE>)){
                    chomp $line ;
                    if ($line !~ /^\#/){
                            $i = $i+1;
                            my @fields=(split /\s{1,}/,$line);
                            my $hit_name=$fields[0]; # domain name
                            my $hit_accession=$fields[1]; # pfam_id
                            my $query_name=$fields[2]; # entry's accession_number
                            my $evalue=$fields[4];
                            my $score=$fields[5];
                            my $description=$fields[18];
                            $description =~ s/'//g; # (') creates problems
                            $class; # ec number
                            my $query="insert into $table
(hit_name,hit_accession,query_name,evalue,score,description,ec_number) values
(\'$hit_name\',\'$hit_accession\',\'$query_name\',\'$evalue\',\'$score'\,\'$description\',\'$class\');" or die
"problem in loading the hmmer results";
                            my $sth = $dbh->prepare($query);$sth->execute();
                    }
            }print FILE_PRO "\t* $i hits\n";
    close FILE;

    my $query = "SELECT count(DISTINCT(hit_accession)) FROM $table where ec_number like
'$class';" or die "problem in query"; # selection of distinct hits for a class
    my $sth = $dbh->prepare($query); $sth->execute();

    while (my @line = $sth->fetchrow_array()) {
            my $d_number_pfam = $line[0];
```

```
        print FILE_PRO "\t* $d_number_pfam distinct hits, i.e. $d_number_pfam Pfam ids.\n";
    }
}
```

######## 6th SUBROUTINE: Pfam_ids manipulation: Firstly, the creation of two tables, $inter_table and $final_table. Secondly, loading data: The $inter_table will include all the pfam_ids from the file_out-table (it will be dropped after the subroutine). The $final_table table will include the pfam-ids 'DISTINCT' (its data will be loaded based on the "DISTINCT SELECTION" of pfam_ids from $inter_table). This is the necessary table)

```
sub pfam_tables {

    (my $table, my $db, my $user_name, my $password) = @_;
    my $dsn = "DBI:mysql:$db:localhost";
    my $dbh  = DBI->connect($dsn, $user_name, $password) or die "cannot connect to MySQL";

    my $inter_table = ($table)[0]."_inter_pfam_ids";
    my $final_table = ($table)[0]."_pfam_ids";

    my $sql_table_1 = "DROP TABLE IF EXISTS $inter_table;" or die "no table $inter_table";
    my $sql_table_2 = "CREATE TABLE $inter_table (
            ind INTEGER NOT NULL AUTO_INCREMENT,
            pfam_id VARCHAR(30),
            PRIMARY KEY(ind)
    );" or die "cannot create table $inter_table";

    my $sth = $dbh->prepare($sql_table_1); $sth->execute();
    my $sth = $dbh->prepare($sql_table_2); $sth->execute();

    my $sql_table_1 = "DROP TABLE IF EXISTS $final_table;" or die "no table $final_table";
    my $sql_table_2 = "CREATE TABLE $final_table (
            ind INTEGER NOT NULL AUTO_INCREMENT,
            pfam_id VARCHAR(30),
            PRIMARY KEY(ind)
    );" or die "cannot create table $final_table";

    my $sth = $dbh->prepare($sql_table_1); $sth->execute();
    my $sth = $dbh->prepare($sql_table_2); $sth->execute();

    my $query = "SELECT DISTINCT(hit_accession) FROM $table;" or die "problem in query";
    my $sth = $dbh->prepare($query); $sth->execute();

    while (my @line = $sth->fetchrow_array()) {
            my $pfam = $line[0];
            my $query ="INSERT INTO $inter_table (pfam_id) values (\'$pfam\');";
            my $sth = $dbh->prepare($query); $sth->execute();
    }

    my $query = "SELECT DISTINCT(pfam_id) FROM $inter_table;" or die "problem in query";
    my $sth = $dbh->prepare($query); $sth->execute();

    my $i = 0;
    while (my @line = $sth->fetchrow_array()) {
            my $pfam = $line[0];
            $i = $i+1;
            my $query ="INSERT INTO $final_table (pfam_id) values (\'$pfam\');";
            my $sth = $dbh->prepare($query); $sth->execute();
    }print FILE_PRO "-Finally $i distinct hits.\n";

    my $sql_table= "DROP TABLE if exists $inter_table;" or die "cannot find $inter_table table";
    my $sth = $dbh->prepare($sql_table); $sth->execute();
}
```

######## 7th SUBROUTINE: Creation of the csv file which include pfam_ids and classes as features. In the first part, the program print the first line which contains all the features names. In the second part, the program find the pfam_ids which characterise each protein and print them in one line. When a protein is characterised by a pfam_id the relevant score is printed in the cell, else the 0 is printed. After

```
the creation of the above csv file, another one is created, which contains only the scores of pfam-ids
and the class. This file is the rapidminer input.

sub create_csv {

    (my $table, my $db, my $user_name, my $password) = @_;

    my $dbh = DBI->connect("DBI:mysql:$db:localhost", $user_name, $password) or die "cannot
connect";

    my $draft_csv = "draft_".($table)[0]."_pfam.csv"; # the name of a draft csv file.
    open (DRAFT_CSV, ">$draft_csv") or die "cannot open file";

        my $pfam_ids_table = ($table)[0]."_pfam_ids"; # the table with distinct pfam ids

        my $query = "SELECT pfam_id FROM $pfam_ids_table;";
        my $sth = $dbh->prepare($query); $sth->execute();

        print DRAFT_CSV 'Feature,,[class],,';

        my $i=0;
        my @pfam_ids;

        while (my @line = $sth->fetchrow_array()) { # the line with all pfam_ids
                my $pfam = $line[0];
                $i=$i+1;
                push @pfam_ids,$pfam;
        }
                foreach my $pfam(@pfam_ids[0..$#pfam_ids]) {
                        if ($pfam eq $pfam_ids[$#pfam_ids]) {
                                print DRAFT_CSV "[$pfam]"; # ex. [PF00001]
                        } else {
                                print DRAFT_CSV "[$pfam],";
                        }
                }print DRAFT_CSV "\n";

        my $n_pfam = $i; # number of pfam_ids

        my $query1 = "SELECT DISTINCT(query_name),ec_number FROM $table;";
        my $sth1 = $dbh->prepare($query1); $sth1->execute();

        while (my @line1 = $sth1->fetchrow_array()) {
                my $query_name = $line1[0];
                my $class = $line1[1];
                print DRAFT_CSV "$query_name,,$class,,"; # ex. UniRefP12345,,3.1.-.-,,

                my $query2 = "SELECT b.ind, a.score FROM $pfam_ids_table b, $table a WHERE
a.hit_accession = b.pfam_id AND a.query_name = '$query_name';";
                my @pfam_indexes;
                my @scores;

                my $sth2 = $dbh->prepare($query2); $sth2->execute();

                        while (my @line2 = $sth2->fetchrow_array()) {
                                my $ind = $line2[0];
                                my $score = $line2[1];

                                push @pfam_indexes,$ind;
                                push @scores,$score;
                        }

                        my @csv_line = (0) x $n_pfam; # the line of each protein is an array with all
the elements 0.
                        my $position_number=0;
                        foreach my $position(@pfam_indexes) { # substitute the 0 with score
wherever is possible
                                $csv_line[$position-1]=$scores[$position_number]; # in position-1
cell the score which caracterise the protein-pfam_id relation is printed
```

```
                                        $position_number++;
                                }
                        print  DRAFT_CSV join(',',@csv_line),"\n";
                }
                close DRAFT_CSV;

    my $command = "sed 's/\,\,/\,/g' $draft_csv >pfam_$table.csv"; # creation of the final csv file
    system("$command");
    print FILE_PRO "-The csv file has been created.\n\n-The sql format file has been created.\n\n";
}

########8th SUBROUTINE: Creation of the pfam_table which will be used in profeat_process.pl

sub pfam_for_prof {

    (my $table, my $db, my $user_name, my $password) = @_;

    my $dbh = DBI->connect("DBI:mysql:$db:localhost", $user_name, $password) or die "cannot
connect";

    my $draft_csv = "draft_".($table)[0]."_pfam.csv"; # the draft csv
    my $feat_table = ($table)[0]."_pfam"; # the name of mysql table, where the data of drast file will be
loaded

    my $sql_table_1 = "DROP TABLE IF EXISTS $feat_table" or die "cannot drop table $feat_table;";
    my $sql_table_2 = "CREATE TABLE $feat_table(
                        ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
                        query_name VARCHAR(100),
                        vector_1 LONGTEXT,
                        vector_2 LONGTEXT,
                        PRIMARY KEY(ind)
                );" or die "cannot create table $feat_table";
    my $sth = $dbh->prepare($sql_table_1);$sth->execute();
    my $sth = $dbh->prepare($sql_table_2);$sth->execute();

    open (FILE_FEAT, "<$draft_csv") or die "cannot read $draft_csv file";
    while (defined(my $line = <FILE_FEAT>)){
            chomp $line;
            if ($line !~ /^Feat/){ ## if line does not start with "Feat"
                        my @fields=(split /\,\,/, $line);
                        my $query_name = $fields[0]; # protein name
                        my $vector_1 = $fields[1]; # in pfam table is the class and in profeat the features
                        my $vector_2 = $fields[2]; # in pfam table is the features and in profeat is empty
                        my $query="insert into $feat_table (query_name, vector_1, vector_2) values
(\'$query_name\',\'$vector_1\', \'$vector_2\');" or die "problem in loading the $draft_csv file";
                        my $sth = $dbh->prepare($query);$sth->execute();
            }
    }
    close FILE_FEAT;
    print FILE_PRO "-The $feat_table table has been created in mysql, so it can be used for profeat
process.\n\n\n---------Hmmer process terminated!---------";
    close FILE_PRO;

    my $command = "rm $draft_csv";
    system("$command");
}
1;
```

- **divide_fasta.pl**

```
#!/usr/bin/perl
use strict;

my $file_in = $ARGV[0];

my $n = 0;
```

```
open (FILE_IN, "<$file_in") or die "cannot open fasta file for dividing";
while (my $line = <FILE_IN>) { # calculation of entries
        chomp $line;
        if ($line =~ m/^\>/) {
                $n = $n+1;
        }
}
if ($n > 1000) { # if true, dividing
        my $i = 1000;
        my $j; # this is for the name of part noX file.
        open (FILE_IN, "<$file_in");
        while (my $line = <FILE_IN>) {
                chomp $line;
                if ($line =~ m/^>/) { # if true, this line is the name of an entry
                        $i = $i+1;
                        if ($i =~ m/^([0-9])+001$/) { # checking of $i, if $i=[XaXb...Xn][00]1 --->
creation of new part.
                                close FILE_PART;
                                $j = ($i-1)/1000; ### ex: $j = 1,2,3...
                                open (FILE_PART, ">$file_in-part$j"); # open a new part
                        }
                } print FILE_PART $line, "\n";
        }
        close FILE_PART;
        close FILE_IN;
}
```

- **profeat_process.pl**

```
#!/usr/bin/perl
use strict;
use lib "$ENV{HOME}/my_packages/";
use Profeat_Process;

#Arguments-input elements
my $class = $ARGV[0]; # hyd_3_1, hyd_3_2 etc
my $db = $ARGV[1];
my $user_name = $ARGV[2];
my $password = $ARGV[3];

my $draft_file_1 = "draft1_".($class)[0]."_profeat.csv";
open (DRAFT_1, ">$draft_file_1");

my @FILES = glob("./profeat_files/*"); # connect all profeat files
my $first_file = $FILES[0];

open (FILE, $first_file);
        while (my $line = <FILE>) {
                print DRAFT_1 $line;
        } print DRAFT_1 “\n”;
close FILE;

foreach my $file(@FILES[1..$#FILES]) {
open (FILE, $file);
        while (my $line = <FILE>) {
                if ($line !~ /^Feat/) {
                        print DRAFT_1 $line;
                }
        }
}close FILE;

my $draft_file_2 = "draft2_".($class)[0]."_profeat.csv"; # this file will be used by mysql parser. in
prof_shaping subroutine it will be written (based on $prof file) with a specific format.

Profeat_Process::prof_shaping ($draft_file_1, $draft_file_2);
Profeat_Process::mysql_table ($draft_file_2, $db, $user_name, $password);
```

```
Profeat_Process::final_csv ($class, $draft_file_1, $db, $user_name, $password);
my $delete = "rm $draft_file_1 $draft_file_2";
system("$delete");
close DRAFT_1;
```

- **Profeat_Process.pl**

```
package Profeat_Process;
use strict;
use DBI;

######## 1st SUBROUTINE: Creation of a modified profeat file, which will be used for the better organisation of the results. This file will consist of two fields, the field of protein names and the field of profeat features.

sub prof_shaping {

	(my $initial_file, my $final_file) = @_;

	open (FILE, "<$initial_file") or die "cannot open draft1_profeat file";
	open (FILE_NEW, ">$final_file") or die "cannot open draft2_profeat file"; ### this file will be used by mysql parser

	while (my $line = <FILE>) {
		chomp $line;
		next if ($line =~ m/^Feat/);
		my @fields=(split /\,/, $line);
		my $query_name = $fields[0];
		$query_name =~ s/ //g;
		print FILE_NEW $query_name,"\,\,"; # this is the 1st field, only the protein name.
							# the 2nd column is empty...the ",," will be the delimiter for mysql parser, not the ",".
		foreach my $feature(@fields[1..$#fields]) { # this is the 2nd field, all features.
			if ($feature==$fields[$#fields]) { # the final feature in a line
				print FILE_NEW "$feature";
			} else { # any other feature in a line should be followed by a ","
				print FILE_NEW "$feature,";
			}
		}print FILE_NEW "\n";
	}
	close FILE;
	close FILE_NEW;
}

######## 2nd SUBROUTINE: Creation of MySQL table, with 2 columns, from a csv file. In the first column the protein name and in the second all the features

sub mysql_table {

	(my $feat, my $db, my $user_name, my $password) = @_;

	my $feat_table = $feat; # the name of $feat_table must be "hyd_X_M_profeat"
	$feat_table =~ s/.csv//;
	$feat_table =~ s/draft2_//;

	my $dsn = "DBI:mysql:$db:localhost";
	my $dbh  = DBI->connect($dsn, $user_name, $password) or die "cannot connect to MySQL";

	my $sql_table_1 = "DROP TABLE IF EXISTS $feat_table" or die "cannot drop table;";
	my $sql_table_2 = "CREATE TABLE $feat_table(
				ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
				query_name VARCHAR(100),
				vector_1 LONGTEXT,
				PRIMARY KEY(ind)
			);" or die "cannot create table";
	my $sth = $dbh->prepare($sql_table_1);$sth->execute();
```

```
	my $sth = $dbh->prepare($sql_table_2);$sth->execute();

	open (FILE_FEAT, "<$feat") or die "cannot open csv file";

	while (defined(my $line = <FILE_FEAT>)){
		chomp $line;
		if ($line !~ /^Feat/){ # if line does not start with "Feat"
			my @fields=(split /\,\,/, $line);
			my $query_name = $fields[0]; # protein name
			my $vector_1 = $fields[1]; # in pfam table is the class and in profeat the
features

			my $query="insert into $feat_table (query_name, vector_1) values
(\'$query_name\',\'$vector_1\');" or die "problem in loading the $feat file";
			my $sth = $dbh->prepare($query);$sth->execute();
		}
	}
	close FILE_FEAT;
}

######### 3rd SUBROUTINE: Creation of a csv from the above table. This subroutine will use the pfam
and profeat tables with the view to create the final csv file.

sub final_csv {

	(my $class, my $initial_file, my $db, my $user_name, my $password) = @_;

	my $dsn = "DBI:mysql:$db:localhost";
	my $dbh  = DBI->connect($dsn, $user_name, $password) or die "cannot connect to MySQL";

	my $prof_table = ($class)[0]."_profeat";
	my $final_file = "profeat_".($class)[0].".csv";
	open (FINAL_CSV, ">$final_file");
	open (FILE, "<$initial_file");

	my $line_prof; # this will be a part of the 1st line in the final csv file
	while (my $line = <FILE>) {
		if ($line =~ m/^Feat/) {
			$line_prof = $line;
			$line_prof =~ s/Feature,//g; # delete the "Feature,"
			print FINAL_CSV "Feature,[class],$line_prof"; # so the 1st line is:
Feature,class,F1,F2,F3,...,FN\n
		}
	}
	my $pfam_table = ($class)[0]."_pfam"; # the name of pfam table in mysql

	my $query = "SELECT a.query_name, b.vector_1, a.vector_1 FROM $prof_table a,
$pfam_table b WHERE a.query_name = b.query_name;";
	my $sth = $dbh->prepare($query); $sth->execute();
	while (my @line = $sth->fetchrow_array()) { # print the data of mysql query
		my $query_name = $line[0];
		my $class = $line[1];
		my $profeat = $line[2];
		print FINAL_CSV "$query_name,$class,$profeat\n"; # the data will be printed like the
1st line!
	}close FINAL_CSV;
}
1;
```

- **rm_multiple_validation.pl**

```
#!/usr/bin/perl
use strict;

#Arguments-input elements
my $rmp_initial = $ARGV[0];
```

```
my $N = $ARGV[1];
my $feat_class = $ARGV[2]; # pfam_hyd_3_2, profeat_hyd_3_1

my $i;
my %files;

for ($i=3; $i<2*$N+3; $i=$i+2) {
	my $file_in = $ARGV[$i]; # fasta file
	my $x= $ARGV[$i+1]; # class of fasta file entries'
	$files{$file_in} = $x;
}

while ((my $file_in, my $x) = each %files) {

	my $PER_per = "Perceptron_performance";
	my $SVM_per = "Support_Vector_Machine_performance";
	my $DT_per = "Decision_Tree_performance";
	my $NBK_per = "Naive_Bayes_Kernel_performance";
	my $kNN_per = "kNN_performance";

	my $PER_log = "Perceptron_log";
	my $SVM_log = "Support_Vector_Machine_log";
	my $DT_log = "Decision_Tree_log";
	my $NBK_log = "Naive_Bayes_Kernel_log";
	my $kNN_log = "kNN_log";

	my $rmp_final = "multiple_validation.rmp"; # rapidminer process file
	my $command = "cp $file_in analysis_file && sed -e 's/file_name/analysis_file/g' $rmp_initial
>$rmp_final";
	system("$command");

	my $execution = "java -cp /home/thodoris/PROGRAMS-DBs/rapidminer/lib/rapidminer.jar
com.rapidminer.RapidMinerCommandLine -f $rmp_final"; # query in command line with the view to run
rapidminer rmp file
	system ("$execution");

	my $final_per = ($feat_class)[0]."_".($x)."_rapidminer_performances";
	my $final_log = ($feat_class)[0]."_".($x)."_rapidminer_log.csv";

	my $cat_rm = "cat $PER_per $SVM_per $DT_per $NBK_per $kNN_per >$final_per && cat
$PER_log $SVM_log $DT_log $NBK_log $kNN_log >$final_log && rm $PER_per $SVM_per $DT_per
$NBK_per $kNN_per $PER_log $SVM_log $DT_log $NBK_log $kNN_log analysis_file $rmp_final &&
sed -i 's/ /,/g' $final_log";
	system("$cat_rm");
}
```

ΠΑΡΑΡΤΗΜΑ (Β) 2ης ΕΝΟΤΗΤΑΣ

- **all_data_parser.pl**

```
#!usr/bin/perl
use strict;
use DBI;
use Bio::SeqIO;

#Arguments-input elements
my $file_data = $ARGV[0];
my $file_seqs = $ARGV[1];
my $table = $ARGV[2];
my $db=$ARGV[3];
my $db_user_name = $ARGV[4];
my $db_password = $ARGV[5];

my $dsn = "DBI:mysql:$db:localhost";
my $dbh = DBI->connect($dsn, $db_user_name, $db_password);

my $sql_table_1= "DROP TABLE if exists data;";
my $sql_table_2="CREATE TABLE data(
        ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        protein_id VARCHAR(30),
        ec_number VARCHAR(50),
        organism MEDIUMTEXT,
        PRIMARY KEY(ind)
);" or die "cannot create data";
my $sth = $dbh->prepare($sql_table_1); $sth->execute();
my $sth = $dbh->prepare($sql_table_2); $sth->execute();

open (FILE, "<$file_data");
while (my $line = <FILE>) {
        chomp $line;
        if ($line !~ /^Entry/) {
                my @fields = (split /\t/, $line);
                my $protein_id = $fields[0];
                my $ec = $fields[3];
                $ec =~ s/;/ &/;
                my $organism = $fields[2];
                        my @org_fields = (split /\s+/, $organism);
                        my $org_name = "$org_fields[0]_$org_fields[1]";
                        $org_name =~ tr/[A-Z]/[a-z]/;
                my $query="insert into data (protein_id, ec_number, organism) values (\'$protein_id\',
\'$ec\', \'$org_name\');" or die "problem in loading the data";
                my $sth = $dbh->prepare($query);$sth->execute();
        }

}close FILE;

my $sql_table_1= "DROP TABLE if exists seqs_table;";
my $sql_table_2="CREATE TABLE seqs_table(
        ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        protein_id_ext VARCHAR(50),
        PRIMARY KEY(ind),
        seq MEDIUMTEXT,
        INDEX(protein_id_ext)
);" or die "cannot create seqs_table";
my $sth = $dbh->prepare($sql_table_1); $sth->execute();
my $sth = $dbh->prepare($sql_table_2); $sth->execute();

my $seqio_obj = Bio::SeqIO->new(-file => $file_seqs, -format => "fasta" );
while (my $seq_obj = $seqio_obj->next_seq){
        my $protein_id = $seq_obj->id;
        my $seq = $seq_obj->seq;
```

```
	my $query="insert into seqs_table (protein_id_ext, seq) values (\'$protein_id\',\'$seq\');" or die
"problem in loading the sequences";
	my $sth = $dbh->prepare($query); $sth->execute();
}

my $sql_table_1= "DROP TABLE if exists $table;";
my $sql_table_2="CREATE TABLE $table(
	ind INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
	protein_id VARCHAR(30),
	ec_number VARCHAR(50),
	organism MEDIUMTEXT,
	PRIMARY KEY(ind),
	seq MEDIUMTEXT,
	INDEX(protein_id)
);" or die "cannot create $table table";;
my $sth = $dbh->prepare($sql_table_1); $sth->execute();
my $sth = $dbh->prepare($sql_table_2); $sth->execute();

my $query = "SELECT a.protein_id,a.ec_number,a.organism,b.seq FROM data a, seqs_table b
WHERE b.protein_id_ext REGEXP a.protein_id;";
my $sth = $dbh->prepare($query); $sth->execute();

while (my @line = $sth->fetchrow_array()) {
		my $protein_id = $line[0];
		my $ec = $line[1];
		my $organism = $line[2];
		my $seq = $line[3];
		my $query="insert into $table (protein_id, ec_number, organism, seq) values
(\'$protein_id\', \'$ec\', \'$organism\', \'$seq\');" or die "problem in loading the $table";
		my $sth1 = $dbh->prepare($query); $sth1->execute();

}
my $drop_1= "DROP TABLE if exists data;";
my $drop_2= "DROP TABLE if exists seqs_table;";
my $sth = $dbh->prepare($drop_1); $sth->execute();
my $sth = $dbh->prepare($drop_2); $sth->execute();
```

- **temperature_parser.pl**

```
#!/usr/bin/perl
use DBI;
use strict;

#Arguments-input elements
my $file_in=$ARGV[0];
my $table = $ARGV[1];
my $db=$ARGV[2];
my $db_user_name = $ARGV[3];
my $db_password = $ARGV[4];

my $dsn = "DBI:mysql:$db:localhost";
my $dbh = DBI->connect($dsn, $db_user_name, $db_password);

my $sql_table_1= "DROP TABLE if exists $table;" or die "cannot drop table $table";
my $sql_table_2="CREATE TABLE $table(
	ind INTEGER NOT NULL AUTO_INCREMENT,
	organism VARCHAR(60),
	temperature VARCHAR(5),
	category VARCHAR(20),
	PRIMARY KEY(ind)
);" or die "cannot create table $table";
my $sth = $dbh->prepare($sql_table_1);$sth->execute();
my $sth = $dbh->prepare($sql_table_2);$sth->execute();

open (FILE, "<$file_in");
```

```
while (my $line = <FILE>) {
	chomp $line;
	if ($line !~ m/^\#/) {
		my @fields = (split /\,/, $line);
		my $name = $fields[0];
		my $temperature = $fields[1];
		my $category = $fields[2];

		my $query="insert into $table (organism, temperature, category) values (\'$name\',
\'$temperature\', \'$category\');" or die "problem in loading the $table";
		my $sth = $dbh->prepare($query);$sth->execute();
	}
}close FILE;
```

- **select_thermo.pl**

```
#!/usr/bin/perl
use strict;
use lib "$ENV{HOME}/my_packages/";
use Select_Thermo;

my $db = $ARGV[0];
my $user_name = $ARGV[1];
my $password = $ARGV[2];
my $N = $ARGV[3];
my $ec = $ARGV[4]; # 3.1., 3.4, non 3.1.-.- or 3.4.-.-
my %hash = ();
my $i;

for ($i=5; $i<5*$N+5; $i=$i+5) {
	my $category = $ARGV[$i]; # fasta file
	my $temperature_low= $ARGV[$i+1]; # class of fasta file entries'
	my $temperature_high = $ARGV[$i+2];
	my $type_of_select = $ARGV[$i+3]; # all or limit
	my $Number_limit = $ARGV[$i+4]; # if all -> $Number_limit=all, if limit ->
$Number_limit=number
	push (@{$hash{$category}}, $temperature_low);
	push (@{$hash{$category}}, $temperature_high);
	push (@{$hash{$category}}, $type_of_select);
	push (@{$hash{$category}}, $Number_limit);
}

while ((my $category, my $elements) = each %hash) {

	my $dir = "./$category";
	my $command = "mkdir $dir";
	system($command);

	my $temp_low = @{$elements}[0];
	my $temp_high = @{$elements}[1];
	my $type_of_select = @{$elements}[2];
	my $Number_limit = @{$elements}[3];
	my $file_out = $category;
	my $final = "$dir/$file_out";

	if ($type_of_select eq 'limit') {
		Select_Thermo::select_seqs_limit ($final, $temp_low, $temp_high, $Number_limit,
$ec, $db, $user_name, $password);
	} else {
		Select_Thermo::select_seqs ($final, $temp_low, $temp_high, $ec, $db, $user_name,
$password);
	}
	Select_Thermo::cd_hit_clustering ($final);
}
```

- **Select_Thermo.pm**

```
package Select_Thermo;
use strict;
use DBI;

######## 1st SUBROUTINE

sub select_seqs {

	(my $file_out, my $temp_low, my $temp_high, my $ec, my $db, my $user_name, my $password) = @_;

	my $dsn = "DBI:mysql:$db:localhost";
	my $dbh = DBI->connect($dsn, $user_name, $password);

	my $query = "SELECT a.protein_id, a.organism, a.seq FROM full_data a, opt_temperature b WHERE ($temp_low <= b.temperature AND b.temperature <= $temp_high) AND a.organism like b.organism AND a.ec_number like '$ec%';" or die "cannot execute the MySQL command 'select'";
	my $sth = $dbh->prepare($query); $sth->execute;

	open (FILE, ">$file_out");
	while (my @line = $sth->fetchrow_array()) {
			my $protein_id = $line[0];
			my $organism = $line[1];
			my $seq = $line[2];
			print FILE ">",$protein_id," ",$organism,"\n",$seq,"\n";
	}close FILE;
}

######## 2nd SUBROUTINE

sub select_seqs_limit  {

	(my $file_out, my $temp_low, my $temp_high, my $N, my $ec, my $db, my $user_name, my $password) = @_;

	my $dsn = "DBI:mysql:$db:localhost";
	my $dbh = DBI->connect($dsn, $user_name, $password);

	my $query = "SELECT a.protein_id, a.organism, a.seq FROM full_data a, opt_temperature b WHERE ($temp_low <= b.temperature AND b.temperature <= $temp_high) AND a.organism like b.organism AND a.ec_number like '$ec%' ORDER BY RAND() LIMIT $N;";
	my $sth = $dbh->prepare($query); $sth->execute;

	open (FILE, ">$file_out");
	while (my @line = $sth->fetchrow_array()) {
			my $protein_id = $line[0];
			my $organism = $line[1];
			my $seq = $line[2];
			print FILE ">",$protein_id," ",$organism,"\n",$seq,"\n";
	}close FILE;
}

######## 3rd SUBROUTINE

sub cd_hit_clustering {

	(my $file_in) = @_;

	my $name = $file_in;
	$name =~ s/.fasta//;
	my $file_clustered = ($name)[0]."_clustered.fasta";
	my $command = "cd-hit -i $file_in -o $file_clustered -c 0.7"; # cd-hit query in command line
	system ("$command");
}
1;
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Alberts, B., Bray, D., Hopkin, K., Jonshon, A., Lewis, J., Raff, M., Roberts, K., Walter, P., 2006. Βασικές Αρχές Κυτταρικής Βιολογίας. Ιατρικές Εκδόσεις Π.Χ. Πασχαλίδης, Αθήνα.
2. Becker, W.M., Kleinsmith, L.J., Hardin, J., 2003. The World of Cell, 5th edition. ed. Benjamin Cummings, USA.
3. Γεωργάτσος, Ι. Γ., Γιουψάνης, Τ. Α., Κυριακίδης, Δ. Α., 2001. Ενζυμολογία. Εκδόσεις Ζήτη, Θεσσαλονίκη.
4. Λυμπεράτος, Γ., Παύλου, Σ., 2011. Εισαγωγή στη Βιοχημική Μηχανική. Εκδόσεις Τζίολα, Θεσσαλονίκη.
5. Πολυμένης, Σ. Α., 2009. Κινητική και Μηχανισμοί Ετερογενών Αντιδράσεων. Εκδόσεις ΕΜΠ, Αθήνα.
6. Schultz, R.M., 2001. Proteins and Protein Structure. Academic Press.
7. Fornes, O., Aragues, R., Espadaler, J., Marti-Renom, M.A., Sali, A., Oliva, B., 2009. ModLink+: improving fold recognition by using protein–protein interactions. BIOINFORMATICS 25, 1506–1512.
8. Sleator, R.D., Walsh, P., 2010. An overview of in silico protein function prediction. Arch Microbiol 192, 151–155.
9. Zhou, X.-X., Wang, Y.-B., Pan, Y.-J., Li, W.-F., 2008. Differences in amino acids composition and coupling patterns between mesophilic and thermophilic proteins. Amino Acids 34, 25–33.
10. Mizuguchi, K., Sele, M., Cubellis, M., 2007. Environment specific substitution tables for thermophilic proteins. BMC Bioinformatics 8, 1–10.
11. Kumar, S., Tsai, C.-J., Nussinov, R., 2000. Factors enhancing protein thermostabiblity. Protein Engineering 13, 179–191.
12. Henikoff, S., Henikoff, J.G., 2000. Amino Acid Substitution Matrices. Advances in Protein Chemistry 54, 74–87.
13. Mount, D.W., 2004. Bioinformatics, Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, USA.
14. Smith, T.F., Waterman, M.S., 1981. Identification of Common Molecular Subsequences. Mol. Bid 195–197.
15. Needleman, S.B., Wunsch, C.D., 1970. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Mol. Bio 48, 443–453.
16. Sean, E.R., 2004. Where did the BLOSUM62 alignment score matrix come from? Computational biology 22, 1035–1036.
17. Altschul, S.F., 1991. Amino Acid Substitution Matrices from an Information Theoretic Perspective. Mol. Biol 219, 555–565.
18. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C., 1978. Atlas of Protein Sequence and Structure, A Model of Evolutionary Change in Proteins 22, 345–352.
19. Henikoff, S., Henikoff, J.G., 1992. Amino Acid Substitution Matrices From Protein Blocks. Biochemistry 89, 10915–10919.
20. Churbanov, A., 2002. PAM matrix for BLAST algorithm [WWW Document]. URL http://bioinfo.ict.ac.cn/~dbu/AlgorithmCourses/Lectures/Lec6-PAM.pdf
21. 2Can Support Portal - Bioinformatics [WWW Document], 2013. . URL http://www.ebi.ac.uk/2can/home.html
22. Χρυσαφίνου, Ουρανία, 2004. Εισαγωγή στις Στοχαστικές Ανελίξεις, Σοφία. ed. Θεσσαλονίκη.

23. Cappe, O., Moulines, E., Ryden, T., 2005. Inference in Hidden Markov Models. Springer, USA.
24. Sean, E.R., 1998. Profile hidden Markov models. Bioinformatics 14, 755–763.
25. Kuznetsov, G., Nabavi, S., Azizi, E., 2010. Lecture 6: Hidden Markov Models I [WWW Document]. URL http://compbio.mit.edu/teaching.html
26. Fejzic, A., Azizi, E., 2010. Lecture 7: HMMs II - Posterior Decoding and Learning [WWW Document]. URL http://compbio.mit.edu/teaching.html
27. Rabiner, L.R., 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Presented at the IEEE, pp. 257–286.
28. Διαμαντάρας, Κωνσταντίνος. *Τεχνητά Νευρωνικά Δίκτυα*. Αθήνα: Κλειδάριθμος, 2007. Print.
29. De Ferrari, Luna et al. “EnzML: Multi-label Prediction of Enzyme Classes Using InterPro Signatures.” *BioMed Central* 13.1 (2012): 1–12.
30. Jung, Jaehee et al. “PoGO: Prediction of Gene Ontology Terms for Fungal Proteins.” *BioMed Central* 11 (2010): 1–9.
31. King, Ross D. et al. “Genome Scale Prediction of Protein Functional Class from Sequence Using Data Mining.” *ACM* (2000): 384–389.
32. Τζαφέστας, Σπύρος. *Έμπειρα Συστήματα Και Εφαρμογές, Με Εισαγωγή Στην Τεχνητή Νοημοσύνη*. Αθήνα: Εκδόσεις ΕΜΠ, 2005. Print.
33. Nilsson, J.Nils. *Introduction to Machine Learning*. Stanford,USA, 1998.
34. Witten, Ian H., and Eibe Frank. *Data Mining, Practical Machine Learning Tools and Techniques*. 2nd ed. USA: Morgan Kaufmann, 2005. Print.
35. Liu, H., Motoda, H., Setiono, R., Zhao, Z., 2010. Feature Selection: An Ever Evolving Frontier in Data Mining. Presented at the The Fourth Workshop on Feature Selection in Data Mining, pp. 4–13.
36. Theodoridis, Sergios, and Konstantinos Koutroumbas. *Pattern Recognition*. 2nd ed. USA: Elsevier Academic Press, 2003. Print.
37. Hofmann, Thomas, Bernhard Schölkopf, and Alexander J. Smola. “Kernel Methods in Machine Learning.” *Institute of Mathematical Statistics* 36 (2008): 1171–1174. Print.
38. Jäkel, Frank, Bernhard Schölkopf, and Felix A. Wichmann. “A Tutorial on Kernel Methods for Categorization.” *Elsevier* 51 (2007): 347–351. Print.
39. Souza, César R. “Kernel Functions for Machine Learning Applications.” 2010.
40. Akthar, Fareed, and Caroline Hahne. *RapidMiner 5, Operator Reference*. Rapid-I GmbH, 2012.
41. Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874.
42. Κολυβά-Μαχαίρα, Φ., and Ε. Μπόρα-Σέντα. *Στατιστική, Θεωρία Εφαρμογές*. Θεσσαλονίκη: εκδόσεις ΖΗΤΗ, 1998. Print.
43. Yang, Ying, and Geoffrey I. Webb. “A Comparative Study of Discretization Methods for Naive-Bayes Classifiers.” (2002): 1–4. Print.
44. Jiawei, Han, and Micheline Kamber. *Data Mining, Concepts and Techniques*. 2nd ed. USA: Morgan Kaufmann, 2006. Print.
45. Liu, Bin et al. “A Comparative Study of Bandwidth Choice in Kernel Density Estimation for Naive Bayesian Classification.” *Springer-Verlag* (2009): 302–303. Print.
46. Zucchini, Walter. “Applied Smoothing Techniques, Part 1: Kernel Density Estimation.” 2003.

47. Nonparametric Density and Regression Estimation [WWW Document], 2007. . URL http://athena.sas.upenn.edu/~petra/class222/nonpar.pdf
48. Density Estimation [WWW Document], n.d. . URL http://www.stat.duke.edu/~wjang/teaching/S05-293/lecture/ch6.pdf
49. Tan, P.-N., Steinbach, M., Kumar, V., 2006. Introduction to Data Mining. Addison-Wesley, USA.
50. Lior, R., Oded, M., n.d. Chapter 9, DECISION TREES [WWW Document]. URL http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
51. Gunn, Steve R. “Support Vector Machines for Classification and Regression.” 1998.
52. Igel, Christian. “Machine Learning: Kernel-based Methods, Lecture Notes.”
53. Διαμαντάρας, Κωνσταντίνος, n.d. Μηχανες Διανυσμάτων Υποστήριξης (Support Vector Machines) -Μάθημα 9ο [WWW Document]. URL http://aetos.it.teithe.gr/~kdiamant/MachineLearning/MachineLearningLesson09.pdf
54. Fahey, Colin. “Neural Network, with Learning by Backward Error Propagation.”
55. Cunningham, Padraig, and Sarah Jane Delany. “k-Nearest Neighbour Classifiers.” 2007.
56. Howley, Tom, and Michael G. Madden. “The Evolution of a Kernel-Based Distance Metric for k-NN Regression.” 2007.
57. UniProt Consortium, n.d. UniProt: the Universal Protein Resource [WWW Document]. URL http://education.expasy.org/Uniprot_flyer.pdf
58. Suzek, B.E., Huang, H., McGarvey, P., Mazumder, R., Wu, C.H., 2007. UniRef: comprehensive and non-redundant UniProt reference clusters. BIOINFORMATICS 23, 1282–1288.
59. Finn, R.D., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J.E., Gavin, O.L., Gunasekaran, P., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E.L.L., Eddy, S.R., Bateman, A., 2009. The Pfam protein families database. Nucleic Acids Research 38, 211–222.
60. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G., n.d. Gene Ontology: tool for the unification of biology. Nat Genet 25, 1–9.
61. lipid catabolic process [WWW Document], 2012. . URL http://amigo.geneontology.org/cgi-bin/amigo/term_details?term=GO:0016042#lineage
62. Barrell, D., Dimmer, E., Huntley, R.P., Binns, D., O’Donovan, C., Apweiler, R., 2009. The GOA database in 2009—an integrated Gene Ontology Annotation resource. Nucleic Acids Research 37, 396–403.
63. Moss, G.P., 2012. Enzyme Nomenclature [WWW Document]. Enzyme Nomenclature, Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes by the Reactions they Catalyse. URL http://www.chem.qmul.ac.uk/iubmb/enzyme/
64. McEntyre, J., Ostell, J., 2002. The BLAST Sequence Analysis Tool, Chapter 16, in: The NCBI Handbook. NCBI, USA, pp. 1–7.
65. The Statistics of Sequence Similarity Scores [WWW Document], n.d. . URL http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html

66. Eddy, S.R., 2010. HMMER User's Guide, Biological sequence analysis using profile hidden Markov models.
67. Bioinformatics & Drug Design group at NUS, 2011. Introduction to structural and physicochemical features calculated by PROFEAT.
68. AAindex, Amino acid indices, substitution matrices and pair-wise contact potentials [WWW Document], 2008. . URL http://www.genome.jp/dbget/aaindex.html
69. Chou, K.C. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. Biochemical and Biophysical Research Communications 2000, 278, 477-483.
70. Chou, K.C and Cai Yu-Dong. Prediction of Protein sucellular locations by GO-FunD-PseAA predictor, Biochemical and Biophysical Research Communications, 2004, 320, 1236-1239.
71. Schneider Gisbert and Wrede Paul. The Rational Design of Amino Acid Sequences by Artifical Neural Networks and Simulated Molecular Evolution: Do Novo Design of an Idealized Leader Cleavge Site. Biophys Journal, 1994, 66, 335-344.
72. Grantham,R. Amino acid difference formula to help explain protein evolution. Science, 1974, 185, 862-864
73. Chou, K.C., n.d. Type 1 pseudo amino acid composition [WWW Document]. URL http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm
74. Chou, K.C., n.d. Type 2 pseudo amino acid composition [WWW Document]. URL http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm
75. Gromiha M.Michael, Suwa Makiko. Influence of amino acid properties for discriminating outer membrane proteins at better accuracy. Biochimica of Biophysica Acta, 2006,1764,1493-1497.
76. Visualization Process - RapidMiner [WWW Document], n.d. . URL http://rapid-i.com/content/view/181/196/
77. Weizhong, L., 2011. CD-HIT User's Guide [WWW Document]. URL http://weizhong-lab.ucsd.edu/cd-hit/wiki/doku.php?id=cd-hit_user_guide
78. De Vendittis, E., Castellano, I., Cotugno, R., Ruocco, M.R., Raimo, G., Masullo, M., 2008. Adaptation of model proteins from cold to hot environments involves continuous and small adjustments of average parameters related to amino acid composition. Journal of Theoritical Biology 250, 156–171.
79. McDonald, J.H., 2010. Temperature Adaptation at Homologous Sites in Proteins from Nine Thermophile–Mesophile Species Pairs. Genome Biol. Evol 2, 267–276.
80. Panasik, N.J., Brenchley, J.E., Farber, G.K., 2000. Distributions of structural features contributing to thermostability in mesophilic and thermophilic K/L barrel glycosyl hydrolases. Biochimica et Biophysica Acta 1543, 189–201.
81. Sadeghi, M., Naderi-Manesh, H., Zarrabi, M., Ranjbar, B., 2006. Effective factors in thermostability of thermophilic proteins. Biophysical Chemistry 119, 256–270.
82. Singer, G.A.C., Hickey, D.A., 2003. Thermophilic prokaryotes have characteristic patterns of codon usage, amino acid composition and nucleotide content. Gene 317, 39–47.

```
>TR:K0IGE1_NITGG K0IGE1 Transposase, IS605 OrfB family OS=Nitrosospheara gargensis
          (strain Ga9.2) GN=Ngar_c09560 PE=4 SV=1
         Length = 370

 Score = 71.2 bits (173), Expect = 8e-13
 Identities = 36/90 (40%), Positives = 55/90 (61%)

Query: 30  SVDLGIVNLATVVVEDGTWYIFKGGGVLSRYEYYSKRVSEVQKVLARHGQKRSRKLKLLY 89
           S+DLGI NL   Y E+    I+ G  VLS Y Y +K+++E+Q  L +  Q  SRK+ L Y
Sbjct: 200 SIDLGICNLVAFYAENEQPVIYSGRAVLSDYVYRTKKIAELQSRLPQKQQHTSRKIGLSY 259

Query: 90  DKRRRFLKHALNSMVRKIMEELREKGVGEV 119
           KD R  KHA  +M++ + E ++ G+ +T
Sbjct: 260 RKRTRRFKHATRAMLKDLFRMKQIGTIKV 289



>TR:F0NBR8_SULIR F0NBR8 Probable translation initiation factor IF-2 OS=Sulfolobus
          islandicus (strain REY15A) GN=infB PE=3 SV=1
         Length = 600

 Score =  164 bits (414), Expect = 7e-44
 Identities = 89/235 (37%), Positives = 153/235 (65%), Gaps = 7/235 (2%)

Query: 17  MYIII-EKELKEIEEKLKVDIENIIFQTDKGIIVKSDTLGSLEVLVKMLRDRNIPIKKA 75
           +Y+I  E ++++ +++++ +I+ +   +D +GI++K+D+LG+LE L+  L+   IPI+ A
Sbjct: 330 LYVIEDESKVEQYKQQIEEIKEVRLYSDIDGTVLKADSLGTLEALIEALQRGGIPIRLA 389

Query: 76  DIGDINKEDVNIALSMKEKVPQYSVIVGFNIGINKNDEEYVKSSNNLIISNVTVDLIEK 135
           DIG I+K DV  A  + ++  +Y +I  F + +     + +S + ++ + +IV LIE
Sbjct: 390 DIGPISKRDVIEASIVAQRSKEYGIIAAFRVKLLQG----IDTSGVKILVNEIIVQLIED 445

Query: 136 IEKVINDIKV-HKERILRELPIGKIKILKGNIFRSNPAIVGVEVLGGEIRKDVVLINL 194
           I+K+IN+I+  K R   L  GKIKIL G +FRS+P +VG+EVL GGEIRKDVVLINL  LI
Sbjct: 446 IKKHINEIRSEAEKRTFDTLILPGKIKILPGVVFRRSDPVVVGIEVIGGVIRPKYPLIKE 505

Query: 195 DGKIIGRILAIQKEKKMNIDIAIKNDKVAVSI-DDAVVGRNLFEDDILVTFISEND 248
           DG+ +G +L IQ  KK+++ A K  +VA+SI  + ++G2++ E DILYT + + D
Sbjct: 506 DGRRVGEVLQIQDNKKSLERATKGMEVAISIKGNVMIGRHIHEGDILYTDVPKED 560



>TR:F6RMI1_XENTR F6RMI1 Uncharacterized protein (Fragment) OS=Xenopus tropicalis
          PE=3 SV=1
         Length = 882

 Score = 38.8 bits (81), Expect = 7.0
 Identities = 27/102 (26%), Positives = 48/102 (44%), Gaps = 7/102 (6%)

Query: 69  LPSYNIISITLPKPFSSFTSRITPTPFPSTFINFPSSKYLWESPAINNNKSGIPSPVFEE 128
           +P Y ++  TLP+ F     I   PFP    + K + +        G PS  +
Sbjct: 702 VPKYILLLCTLPSVFICALWLIISPPFPDYDTHSATGKIILQC------NEGSPSAFYIM 755

Query: 129 TGMIATCLLKSFTLSYLSESNPNSYNLATILYVYSSNLFITL 170
            G IA     SF ++YL+   P+ +N A  +  +S  LF ++
Sbjct: 756 VGYIALLAFVSFFVAYLARKLPDIFNEAQYI-TFSMLLFCSV 796



>TR:F8AJ59_PYRYC F8AJ59 IS element ISTsi3 orfB, probably transposase OS=Pyrococcus
          yayanosii (strain CH1 / JCM 16557) GN=PYCH_08150 PE=4
          SV=1
         Length = 432

 Score = 74.3 bits (181), Expect = 8e-14
 Identities = 37/96 (38%), Positives = 59/96 (61%)

Query: 25  GGLRASVDLGIVNLATVVVEDGTWYIFKGGGVLSRYEYYSKRVSEVQKVLARHGQKRSRK 84
           G L A +DLG+ NL  VVVE+G  +E  G  + S   Y+ K+++ Q  L +   K+SRK
Sbjct: 193 GSLTAGIDLGVNNLMAVVVENGKSFLVNGRPLKSIAFYWQKQIADVQSKLNKSSAMKSRK 252

Query: 85  LKLLYDKRRRFLKHALNSMVRKIMEELREKGVGEVV 120
           LK +Y K +   KH +N+ VR+ +E+L + GV  +V
Sbjct: 253 LKRMYQKAKLQAKHYINTAVRQTVEKLYQLGVSRIV 288



>TR:E6Q727_9ZZZZ E6Q727 Uncharacterized protein OS=mine drainage metagenome PE=4
          SV=1
         Length = 493

 Score = 80.9 bits (198), Expect = 3e-13
 Identities = 96/404 (23%), Positives = 159/404 (39%), Gaps = 49/404 (12%)

Query: 148 PLGVTLACGLLYATE----DSMPGSITAINPVTGKIVWMANGLAGPANNPVVWHGIVYV 203
           PLG  +  GL Y       DS    + ++N  TG+I W A  L G AWN PV+  G+V V
Sbjct: 107 PLGSKVNGGLSYGNGRIYVDSFDRHLYSLNAKTGRIEWSAP-LGGVAWNTPVLADGVVYV 165

Query: 204 SVGDUCFTSQFVWFEEHQYCKINPGRCGAVYAFNATDQQLLNRGRFTYGEAMPAPAI--- 260
           S    G     +  V  +    R     V AFNA  G  LW   T GE MP+  I
Sbjct: 166 GTG------TNRVMIDTPTQVLWGRPGGDNVLAFNANTGAPLWRYHTVGEDMPSAVIVMR 219

Query: 261 --YDGILAVVTGGGCFVGVNATTGQILWVDHFPGFMAAMASVWYYVLP-HGTPLFIAGFT 317
             D  + +  G G     ++  TG+ +W  + G +A M+S  Y    G   F  F
Sbjct: 220 PRQDPEVVFSNGDGYVRSLDLMTGKKIWKSYLNG-VAIMSSISYKSZLFGIENFSVSFI 278

Query: 318 YAAA--PVGYIIA-----VNGLNGHEAWNATMPAPVYGAMTGLGDVAPAVDQQLGIVVDN 370
           +A    P     A     +N  NG   W     +     +   G +     +  +VDN
Sbjct: 279 FANNMDPTRLRFATKTWSNMAKNGQLIWTIPYGSSDSPSASWGRLSFQTSKLVYPITTG 338

Query: 371 DIAHFSNGVVDMVTFALHATNGKPLHAVNTGRGPIPPAYKGGMPLIVGNVVYDGNPSLGT 430
           +  ++S    + +A++   G+ +N  +G G    +Y  G    V  V  D  G
Sbjct: 339 NTLSYS------IVYAVDINTGRIIWEKKSGTG--YRSLRGSNEDAIAGAVMDGYYLTGL 390

Query: 431 VNAIC-----IKTGKILWSTKLPCLQTPPKFPGGPRGSPTYYNG-LLMVSAGQVYVVINP 484
             C     +K G + W  K          G  + S     G +++     ++ ++VVINP
Sbjct: 391 PFERCFAVYVNVKNGDLRWKLKT---------TGPVMGSSVIVDGKVIFGDTHDFLYVVNF 441

Query: 485 KNGDILTFYWIGGRLGIMNPVIAGNTMFLSNSVGWIVAIPLSQI 528
           K ++T          +P++ G T++++N+   I AIP++ I
Sbjct: 442 NTGRLVTKVKFPSFFTTSSPIVVGKTLYIANAKS-IYAIPVNNI 484



>TR:F0QVS1_VULMT F0QVS1 IS element ISDka2 orfB OS=Vulcanisaeta moutnovskia (strain
          768-28) GN=VMUT_1894 PE=4 SV=1
         Length = 423

 Score = 76.6 bits (187), Expect = 1e-14
 Identities = 37/96 (38%), Positives = 59/96 (61%)

Query: 24  RGGLRASVDLGIVNLATVYVEDGTWYIFKGGGVLSRYEYYSKRVSEVQKVLARHGQKRSR 83
           +  LRA +D+GI NL  VY+ DG  ++  G  + +   Y+  R++  Q  L R+G + SR
Sbjct: 191 KASLRAGIDIGINNLFAVYIGDGRAFLVNGRPLKAISYYMKARIANYQSTLNRYGLRTSR 250

Query: 84  KLKLLYDKRRRFLKHALNSMVRKIMEELREKGVGEV 119
           +L+++Y K DR +KH +N+ VR++ EEL   GV  V
Sbjct: 251 RLRIMYKKKRQVKHYINTAVRRLAEELYNAGVSTV 286
```