ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΒΙΟΛΟΓΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ - ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΛΟΓΙΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και υλοποίηση υπολογιστικής μεθόδου για την ταξινόμηση πεπτιδίων με βάση τη βιοδραστηριότητά τους»

Ευσταθία Δ. Κολοτούρου

Α.Μ.: 268807

Αναπληρωτής Καθηγητής Αλέξανδρος Γεωργακίλας (Επιβλέπων)
Τομέας Φυσικής, Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, Εθνικό Μετσόβειο Πολυτεχνείο

ΑΘΗΝΑ
ΜΑΡΤΙΟΣ 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και υλοποίηση υπολογιστικής μεθόδου για την ταξινόμηση πεπτιδίων με βάση τη βιοδραστηριότητά τους»

Ευσταθία Δ. Κολοτούρου

Α.Μ.: 268807

ΕΠΙΒΛΕΠΩΝ

Αναπληρωτής Καθηγητής Αλέξανδρος Γεωργακίλας
Τομέας Φυσικής, Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, Εθνικό Μετσόβειο Πολυτεχνείο

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Αναπληρωτής Καθηγητής Αλέξανδρος Γεωργακίλας
Τομέας Φυσικής, Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, Εθνικό Μετσόβειο Πολυτεχνείο

Καθηγητής Κωνσταντίνος Βοργιάς
Τομέας Βιοχημείας και Μοριακής Βιολογίας, Τμήμα Βιολογίας, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Καθηγητής Παντελεήμων Μπάγκος
Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική, Πανεπιστήμιο Θεσσαλίας

ΑΘΗΝΑ
ΜΑΡΤΙΟΣ 2019

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, η ανακάλυψη νέων θεραπειών κατά των λοιμώξεων έχει καταστεί επείγουσα εξαιτίας της ταχέως αυξανόμενης αντίστασης που εμφανίζεται έναντι στα συμβατικά αντιβιοτικά. Το γεγονός οτι τα περισσότερα δυνητικά αντιβιοτικά αποτυγχάνουν να σκοτώσουν παθογόνους οργανισμούς έχει αναπτύξει την ανάγκη σχεδιασμού νέων αντιμικροβιακών φορέων και η κλινική έρευνα πλέον επενδύει στην ταυτοποίηση νέων, μη συμβατικών θεραπειών κατά των μολύνσεων. Τα αντιμικροβιακά πεπτίδια (AMPs) έχουν κεντρίσει το ενδιαφέρον των ερευνητικών ομάδων ως νέα υποψήφια φάρμακα και χάρη στην ικανότητά τους να προστατεύουν το ξενιστή από ποικίλα παθογόνα βακτήρια είναι γνωστά και ως πεπτίδια άμυνας του ξενιστή. Πρόκειται για ολιγοπεπτίδια όπου ο αριθμός των αμινοξικών καταλοίπων τους κυμαίνεται συνήθως από πέντε ως εκατό και δρουν αποτελεσματικά έναντι ποικίλων στόχων, όπως είναι τα βακτήρια, οι ιοι, οι μύκητες και τα παράσιτα. Τα φυσικά αντιμικροβιακά πεπτίδια απαντώνται τόσο σε προκαρυωτικούς (π.χ. βακτήρια) όσο και σε ευκαρυωτικούς οργανισμούς (π.χ. πρωτόζωα, μύκητες, φυτά, έντομα και ζώα) και τα περισσότερα εμφανίζουν κατιονική και αμφιπαθητική ιδιότητα. Η αμφιπαθητικότητα τους τα καθιστά ιδανικά για το σχηματισμό διεπαφών και τους δίνει την ικανότητα διατάραξης της φυσικής ακεραιότητας της μικροβιακής μεμβράνης. Επιπλέον, οι ηλεκτροστατικές δυνάμεις μεταξύ της αρνητικά φορτισμένης βακτηριακής μεμβράνης και των κατιονικών αντιμικροβικακών πεπτιδίων, είναι ακόμα ένας καθοριστικός παράγοντας αυτής της αλληλεπίδρασης πεπτιδίου – μεμβράνης. Το ευρύ φάσμα των ιδιοτήτων που χαρακτηρίζουν τα αντιμικροβιακά πεπτίδια καθιστά τον *in-silico* σχεδιασμό νέων συνθετικών AMPs μια σοβαρή πρόκληση. Αρκετές δημόσιες βάσεις δεδομένων έχουν ηδη δημιουργηθεί με αποθηκευμένη πληροφορία για εκατοντάδες AMPs. Η ταυτοποίηση και ο χαρακτηρισμός των AMPs και των λειτουργικών τους τύπων έχει οδηγήσει σε πολλές μελέτες και ένας μεγάλος αριθμός μεθόδων έχει προταθεί για αυτόν τον σκοπό. Οι περισσότερες μέθοδοι έχουν εξάγει πληροφορία από την στοίχιση αλληλουχιών, ωστόσο, η αμινοξική σύνθεση δεν μπορεί να επεξηγήσει πάντα πλήρως τους μηχανισμούς αλληλεπίδρασης των AMPs. Πρόσφατα, η μηχανική εκμάθηση έχει συντελέσει στην ανάπτυξη διάφορων μεθόδων πρόβεψης, οι οποίες βασίζονται στα συνθετικά χαρακτηριστικά της αμινοξικής αλληλουχίας των AMPs.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η υλοποίηση μιας καινοτόμου υπολογιστικής μεθοδολογίας μηχανικής μάθησης που θα προσφέρει τη δυνατότητα χαρακτηρισμού των πεπτιδίων με βάση την αντιφλεγμονώδη ή/και την αντικαρκινική τους δράση. Το πρώτο βήμα ήταν η συλλογή πεπτιδίων από διάφορες κατηγορίες βιοδραστηριότητας, αντικαρκινική, αντιμικροβιακή, αντιβακτηριακή, αντιμυκητιακή, αντιϊική και εντομοκτόνα από τις υπάρχουσες βάσεις δεδομένων DRAMP και DAMPD, ως το θετικό δείγμα εκπαίδευσης. Παράλληλα, από τη Uniprot συλλέχθησαν πεπτίδια μήκους 10 ως 100 καταλοίπων με το κριτήριο να μην ανήκουν σε κάποια κατηγορία βιοδραστηριότητας, ως το αρνητικό δείγμα εκπαίδευσης. Η τεχνική της τυχαίας υπερδειγματοληψίας και υποδειγματοληψίας χρησιμοποιήθηκε για την αύξηση ή τη μείωση, αντίστοιχα, των δειγμάτων και το τελικό σύνολο πεπτιδίων διαμορφώθηκε σε 1491 πεπτίδια, 213 από κάθε μελετώμενη κατηγορία. Στη συνέχεια, υπολογίστηκε, για

κάθε ένα πεπτίδιο, ένας αριθμός (44) από φυσικοχημικά και ακολουθιακά χαρακτηριστικά, τα οποία εχει αποδειχθεί να αποτελούν τα πιο αντιπροσωπευτικά στον χαρακτηρισμό των αντιμικροβιακών πεπτιδίων. Με αυτόν τον τρόπο δημιουργήθηκε το σύνολο δεδομένων εκπαίδευσης του μοντέλου πρόβλεψης. Το δεύτερο βήμα ήταν η εκτέλεση της EnsembleGASVR μεθοδολογίας ταξινόμησης, μιας υπολογιστικής μεθόδου που προτάθηκε στο παρελθόν για την πρόβλεψη ουδέτερων και παθογενών πολυμορφισμών κατηγοριοποιώντας τους Σημειακούς Μονονουκλεοτιδικούς Πολυμορφισμούς (SNPs) σε ουδέτερους και σε αυτούς που σχετίζονται με κάποια ασθένεια. Η εν λόγω μεθοδολογία συνδυάζει έναν ταξινομητή Support Vector Regression (SVR) και έναν Γενετικό Αλγόριθμο (GA). Επιπροθέτως, η μέθοδος αυτή έχει βελτιωθεί κατόπιν αντικατάστασης του γενετικού αλγόριθμου με έναν εξελικτικό multi-objective αλγόριθμο, σε μια προσπάθεια επίτευξης αυξημένης απόδοσης. Η μεθοδολογία αυτή έχει το πλεονέκτημα ότι χειρίζεται με πολύ αποδοτικό τρόπο τις ελλιπείς τιμές ενώ επίσης τα μοντέλα που εξάγονται από αυτήν παρέχουν και ένα βαθμό εμπιστοσύνης για κάθε πρόβλεψη τους. Επιπλέον κατασκευάστηκε και ένα δοκιμαστικό σύνολο δεδομένων, που εφαρμόστηκε ως είσοδος στο μοντέλο για την αξιολόγηση της απόδοσής του με βάση το αν προέβλεψε σωστά ή όχι την κατηγορία στην οποία ανήκει κάθε πεπτίδιο. Η εκτέλεση του αλγόριθμου επαναλήφθηκε πέντε φορές και ο μέσος όρος της ακρίβειας πρόβλεψης υπολογίστηκε 66.77 ενώ η διακύμανση υπολογίστηκε 31.07. Απώτερος σκοπός είναι η βελτίωση της απόδοσης και τελικά η εφαρμογή των εκπαιδευμένων μοντέλων σε ένα αυξημένο σύνολο από αχαρακτήριστα πεπτίδια που περιλαμβάνονται σε φυτικούς οργανισμούς έτσι ώστε να προταθούν καινούρια πεπτίδια και φυτά που μπορούν να χρησιμοποιηθούν για την καταπολέμηση των φλεγμονών αλλά και την πρόληψη κατά του καρκίνου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μεθοδολογία μηχανικής μάθησης με επίβλεψη για την ταξινόμηση πεπτιδίων με αναγνωρισμένη βιοδραστηριότητα.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αντιμικροβιακό πεπτίδιο, βιοδραστηριότητα, αντιμικροβιακές ιδιότητες, Μηχανική Εκμάθηση, Support Vector Regression, Ταξινόμηση

ABSTRACT

In recent years, the discovery of new infection therapeutics has become an urgent due to the rapidly increasing infection resistance toward conventional antibiotics. The fact that most potential antibiotics fail to kill pathogens has grown the need to design new antimicrobial agents and the clinical research is now invested into identification of new, non-conventional anti-infective therapies. Antimicrobial peptides (AMPs) have captured the research attention as novel drug candidates and they are also known as host defense peptides because they can protect the host from various pathogenic bacteria. They are oligopeptides with a varying number of amino acids, five to one hundred, and they are effective towards a wide variety of targets, such as bacteria, viruses, fungi and parasites. Natural AMPs can be found in both prokaryotes (e.g., bacteria) and eukaryotes (e.g., protozoan, fungi, plants, insects, and animals) most of which are cationic and amphipathic. This amphipathicity gives them the ability to disrupt the physical integrity of the microbial membrane. Moreover, the electrostatic forces between the negatively charged bacterial surface and the cationic AMPs is another determinant for this interaction between peptides and microbial membrane. The design of novel synthetic AMPs, in-silico, has turned to be a serious challenge because of the broad spectrum of properties that characterize AMPs. Public databases have been already developed and store useful information for hundreds of natural AMPs. Identifying and characterizing AMPs and their functional types has led to many studies and a number of methods have been proposed. Most of them have extracted information from sequence alignment, however, the amino acid composition cannot fully explain the interaction mechanisms of AMPs. Recently, various predictors using machine learning have been developed, based on the compositional characteristics of AMPs amino acid sequences. In this study, a novel computational methodology has been implemented which will offer the ability to characterize the peptides depending on their anti-inflammatory and/or anticancer bioactivity. The first step was the collection of antimicrobial, anticancer, antibacterial, antifungal, antiviral and insecticidal AMPs from DRAMP and DAMPD databases, as the positive sample. Peptides that do not appear to present any bioactivity were collected from Uniprot, as the negative sample. The technique of boosting and random oversampling increased the number of the samples and the final peptide amount was 1491; 213 from each studied category. A number of physicochemical and sequential features (44), which have been proved to be the most indicative for characterizing the antimicrobial peptides, was calculated for each peptide and constituted the final training dataset. The second step was the execution of EnsembleGASVR classification methodology; a computational framework that was proposed in the past for predicting neutral and pathogenic polymorphism variations by classifying missense SNPs (Single Nucleotide Polymorphism) to neutral and disease associated. This methodology facilitates a two-step algorithm, which combines a Support Vector Regression (SVR) classifier with a Genetic Algorithm. Additionally, the method was further improved after replacing the Genetic Algorithm, in its second step, with an evolutionary multiobjective framework, in an attempt to achieve higher performance. The advantage of this method is the effective handling of missing values and the produced models provide a confidence score on every prediction. The training dataset was used to train the aforementioned

algorithm. Finally, the quality of the method was evaluated after applying the trained models on a test dataset which predicted the category of the bioactivity that a peptide could belong to. The algorithm was executed five times and the mean of accuracy was calculated to 66.77, while the variance was calculated to 31.07. The ultimate purpose of this algorithm is to apply the best trained models on a superset of uncharacterized peptides, which are met in plant organisms, in order to propose new peptides and plants that can be used in the cancer prevention and the fight against inflammation.

SUBJECT AREA: Machine learning methodology for antimicrobial peptides classification.

KEYWORDS: antimicrobial peptide, bioactivity, antimicrobial features, Machine Learning, Support Vector Regression, Evolutionary algorithms, Classification.

Στο φύλακα άγγελο μου,

Κωνσταντίνο Καλούρη

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο της φοίτησης μου στο «ΜΔΕ Βιοπληροφορική – Υπολογιστική Βιολογία» του τμήματος Βιολογίας του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών σε συνεργασία με την εταιρεία InSyBio Ltd και τον Δρ. Κωνσταντίνο Θεοφιλάτο.

Θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνησή της και ιδιαίτερα:

Τον επιβλέποντα καθηγητή μου, Αναπληρωτή Καθηγητή Αλέξανδρο Γεωργακίλα, για την εμπιστοσύνη που μου έδειξε στην ανάθεση αυτής της διπλωματικής εργασίας και τη μέγιστη συμβολή του στη διαμόρφωση ενός υγιούς κλίματος συνεργασίας ως προς την ολοκλήρωσή της.

Τον Δρ. Θεοφιλάτο Κωνσταντίνο και την εταιρεία InSyBio Ltd για την πολύτιμη βοήθεια και την καθοδήγηση τους. Η παροχή των απαραίτητων εργαλείων σε συνδυασμό με τις κατάλληλες υποδείξεις συνετέλεσαν στην ανάπτυξη και την ομαλή διεξαγωγή της εργασίας.

Τον Καθηγητή Κωνσταντίνο Βοργιά και τον Αναπληρωτή Καθηγητή Παντελεήμων Μπάγκο που δέχτηκαν να αποτελέσουν μέλη της Τριμελούς Επιτροπής Αξιολόγησης της διπλωματικής μου εργασίας.

Τους συναδέλφους και το επιστημονικό προσωπικό του Μεταπτυχιακού προγράμματος «Βιοπληροφορική – Υπολογιστική Βιολογία» για την άψογη συνεργασία, τις συμβουλές και την υποστήριξη τους στη διάρκεια των σπουδών μου.

Τέλος, την οικογένεια μου για την ηθική συμπαράσταση και την υπομονή που υπέδειξε τόσο κατά διαδικασία μελέτης, διεξαγωγής και συγγραφής της παρούσας διπλωματικής εργασίας όσο και καθ’ όλη την διάρκεια της πραγματοποίησης των μεταπτυχιακών σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

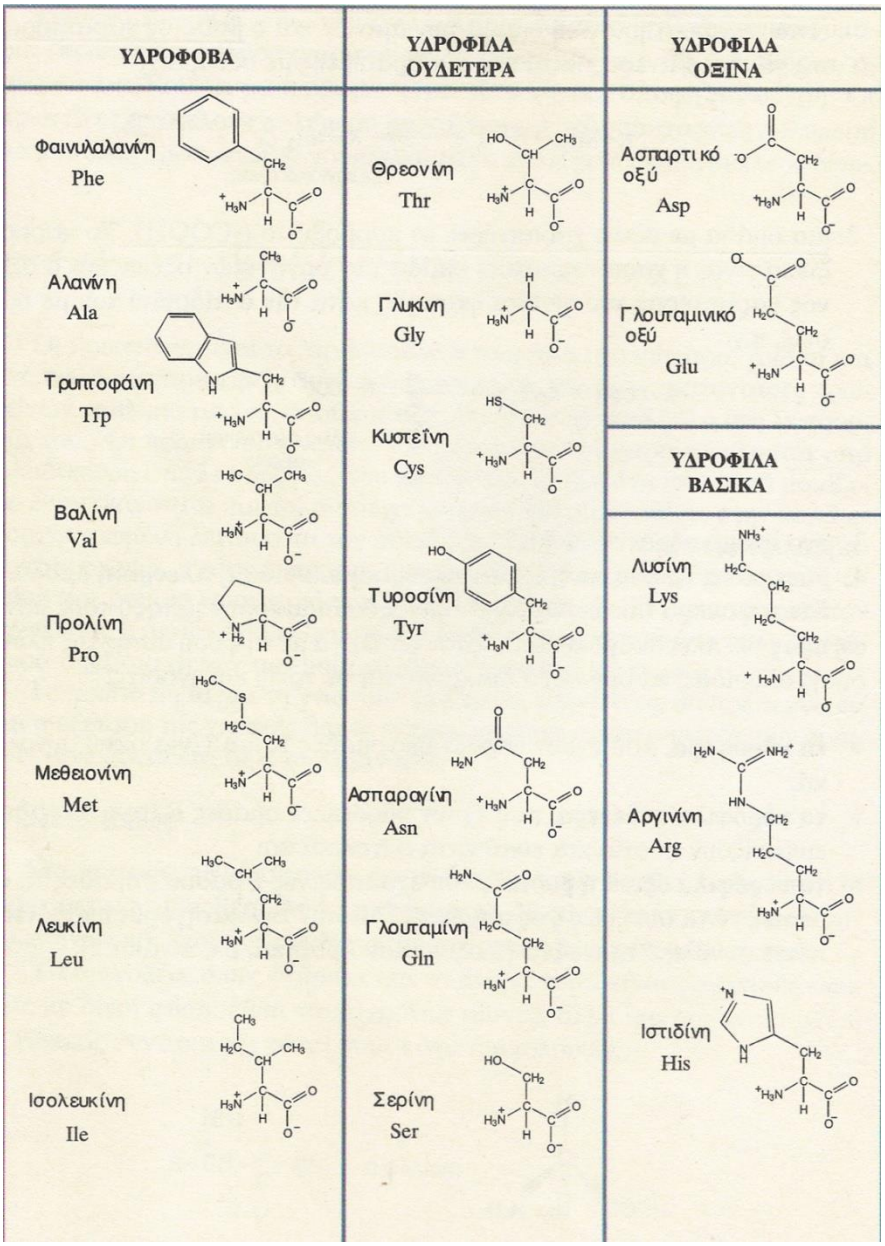
ΚΕΦΑΛΑΙΟ 1

1.1 Πεπτίδια

Οι πρωτεΐνες, τα πιο πολυδύναμα μακρομόρια στους ζωντανούς οργανισμούς, συμμετέχουν σχεδόν σε όλες τις βιολογικές διεργασίες εξυπηρετώντας τις βασικές λειτουργίες. Η σημασία τους μπορεί να συνοψιστεί στις παρακάτω λειτουργίες:

i. **Ενζυμική κατάλυση**. Οι πρωτεΐνες διαδραματίζουν το μοναδικό ρόλο της πορείας των χημικών αντιδράσεων που γίνονται στα βιολογικά συστήματα. Οι αντιδράσεις αυτές καταλύονται από ειδικά μακρομόρια που ονομάζονται ένζυμα και επιταχύνουν την ταχύτητα των αντιδράσεων. Όλα τα γνωστά ένζυμα είναι πρωτεΐνες.
ii. **Μεταφορά και αποθήκευση**. Πολλά μικρά μόρια και ιόντα μεταφέρονται από ειδικές πρωτεΐνες, όπως οι αιμοσφαιρίνη που μεταφέρει οξυγόνο στα ερυθροκύτταρα και η μυοσφαιρίνη στους μυς. Ο σίδηρος μεταφέρεται στο πλάσμα του αίματος από την τρανσφερίνη και αποθηκεύεται στο συκώτι σαν σύμπλοκο με φερριτίνη, μια διαφορετική πρωτεΐνη.
iii. **Κίνηση**. Τα κύρια συστατικά των μυών είναι πρωτεΐνες. Η συστολή των μυών επιτυγχάνεται με την ολισθητική κίνηση δύο ειδών πρωτεϊνικών νηματίων.
iv. **Μηχανική στήριξη**. Η μεγάλη αντοχή του δέρματος και των οστών οφείλονται στην παρουσία του κολλαγόνου, μιας πρωτεΐνης που σχηματίζει ίνες.
v. **Ανοσοπροστασία**. Τα αντισώματα έχουν μεγάλο ζωτικό ρόλο καθώς είναι πολύ ειδικές πρωτεΐνες που αναγνωρίζουν και συνενώνονται με ξένες ουσίες όπως οι ιοί, τα βακτήρια και κύτταρα από άλλους οργανισμούς.
vi. **Δημιουργία και μετάδοση νευρικών παλμών**. Η απόκριση των νευρικών κυττάρων σε ειδικά ερεθίσματα υποβοηθείται από πρωτεΐνες-υποδοχείς. Μόρια υποδοχείς που μπορούν να «διεγερθούν» με ειδικά μόρια, όπως η ακετυλοχολίνη, είναι υπεύθυνα για τη μετάδοση των νευρικών παλμών στις συνάψεις, τις επαφές δηλαδή, μεταξύ νευρικών κυττάρων.
vii. **Έλεγχος της ανάπτυξης και διαφοροποίησης**. Η ελεγχόμενη έκφραση των γενετικών πληροφοριών είναι ουσιώδης για την κανονική ανάπτυξη και διαφοροποίηση των κυττάρων. Μόνο ένα μικρό κλάσμα του γονιδιώματος ενός κυττάρου εκφράζεται κάθε φορά.

Οι πρωτεΐνες δομούνται από ένα σύνολο 20 αμινοξέων. Τα αμινοξέα έχουν ένα κεντρικό άτομο άνθρακα C_a, στο οποίο συνδέονται ένα άτομο υδρογόνου H, μία αμινομάδα NH_2 και μία καρβοξυλομάδα COOH. Αυτό που διαφοροποιεί τα αμινοξέα είναι η πλευρική αλυσίδα R που συνδέεται στον άνθρακα C_a. Υπάρχουν 20 διαφορετικά ήδη πλευρικών αλυσίδων με διαφορετικό σχήμα, φορτίο, μέγεθος υδροφοβικότητα, δεσμευτική συγγένεια οξυγόνου και χημική αντιδραστικότητα.



Εικόνα 1 Αμινοξικά κατάλοιπα (Amino acids)

Οι πρωτεΐνες χτίζονται σταδιακά από αμινοξέα που ενώνονται μεταξύ τους με ένα δεσμό που ονομάζεται **πεπτιδικός δεσμός**. Κατά το σχηματισμό ενός πεπτιδικού δεσμού, η

καρβοξυλομάδα του ενός αμινοξέος ενώνεται με την αμινομάδα του επόμενου αμινοξέος και η αντίδραση αυτή συνοδεύεται με την απώλεια ενός μορίου νερού H_2O. Το μόριο που προκύπτει ονομάζεται **διπεπτίδιο**. Καθώς η διαδικασία επαναλαμβάνεται, το αποτέλεσμα είναι η δημιουργία μιας πολυπεπτιδικής αλυσίδας, με κάθε ομάδα αμινοξέος να ονομάζεται **κατάλοιπο**, και παρουσιάζει πολικότητα δεδομένου ότι τα δύο άκρα της είναι διαφορετικά. Το αμινο-τελικό άκρο θεωρείται η αρχή της πολυπεπτιδικής αλυσίδας και το καρβοξυ-τελικό άκρο το τέλος της.

Επομένως, οι πρωτεΐνες είναι γραμμικά πολυμερή, που αποτελούνται κάτα μέσο όρο από 50 εως 2000 αμινοξικά κατάλοιπα που σχηματίζουν μία πολυπεπτιδική αλυσίδα. Βασικός παράγοντας διαχωρισμού μεταξύ πρωτεΐνης και πεπτιδίου είναι το μέγεθος. Τα πεπτίδια που έχουν μικρότερο αριθμό αμινοξέων (2 εως 50) ονομάζονται **ολιγοπεπτίδια** ή απλώς **πεπτίδια**.

1.2 Αντιμικροβιακά πεπτίδια

Μια σημαντική κατηγορία αποτελούν τα πεπτίδια που εμφανίζουν κάποια βιοδραστηριότητα, αντιμικροβιακή, αντιφλεγμονώδη κοκ και αναφέρονται στη βιβλιογραφία, γενικά, ως **αντιμικροβιακά** (**AMPs, Antimicrobial Peptides**). Πρόκειται για ολιγοπεπτίδια, με μικρό μοριακό βάρος, που αποτελούνται κατά μέσο όρο από 5 εως 100 αμινοξικά κατάλοιπα. Ένα βασικό χαρακτηριστικό που τα διακρίνει είναι η κατιονική, αμφιπαθική δομή που εμφανίζουν (Bahar & Ren, 2013a). Στην πεπτιδική αλυσίδα στην πλειοψηφία των αντιμικροβιακών πεπτιδίων έχουν εντοπιστεί κυρίως θετικά φορτισμένα (κατιονικά) αμινοξικά κατάλοιπα, όπως η Λυσίνη Lys (K) και η Αργινίνη Arg (R) καθώς και κατάλοιπα που έχουν τόσο πολικό όσο και μη πολικό χαρακτήρα (αμφιπαθητικά), Λυσίνη Lys (K), Μεθειονίνη Met (M), Τυροσίνη Tyr (Y), Τρυπτοφάνη Trp (W). Η δομή αυτή συνεισφέρει στην αλληλεπίδραση με την ανιονική λιπιδική κυτταρική μεμβράνη.

Η λυσοζύμη ήταν από τις πρώτες πρωτεΐνες με αντιμικροβιακές ιδιότητες που ταυτοποιήθηκε το 1922 από τον Alexander Fleming στη ρινική βλέννα. Αργότερα, η ανακάλυψη της πενικιλίνης το 1928 σε συνδυασμό με την «Χρυσή Εποχή των αντιβιοτικών» τη δεκαετία του 1940 οδήγησε στην απώλεια του ενδιαφέροντος για τις δυνητικά θεραπευτικές ιδιότητες των φυσικών AMPs, όπως η λυσοζύμη. Ωστόσο, το

γεγονός ότι πολλοί παθογενείς μικροβιακοί οργανισμοί άρχισαν να εμφανίζουν όλο και περισσότερο ανθεκτικότητα σε πολλά φάρμακα επανέφερε το ενδιαφέρον για τα AMPs ως μόρια άμυνας του ξενιστή.

Τα αντιμικροβιακά πεπτίδια εντοπίζονται στο ανοσοποιητικό σύστημα σε ποικιλία ειδών στη φύση, όπως τα θηλαστικά, τα αμφίβια, τα έντομα κλπ (Boohaker, Lee, Vishnubhotla, Perez, & Khaled, 2012). Δραστηριοποιούνται κατά των Gram - θετικών και Gram – αρνητικών βακτηρίων, των μυκήτων, των πρωτοζώων, των παρασίτων, των καρκινικών κυττάρων καθώς και κατά διαφόρων ειδών ιών. Χάρη στην ικανοτητά τους να καταπολεμούν μικρόβια μελετώνται ευρέως ως αντιβιοτικά και ειδικά στην αποτελεσματική καταπολέμηση μικροβιών που παρουσιάζουν πολλαπλή αντίσταση σε φαρμακευτική αγωγή (MDR, multi-drug resistant) (Gordon, Romanowski, & McDermott, 2005; Hancock & Scott, 2000; Khamis, Essack, Gao, & Bajic, 2015; Seshadri Sundararajan et al., 2012a). Τα αντιμικροβιακά πεπτίδια αλληλεπιδρούν με τα μικρόβια – στόχους διεισδύοντας στη κυτταροπλασματική μεμβράνη τους, και τελικά επηρεάζουν ή ακόμα και καταστρέφουν τη διαμόρφωση της μεμβράνης.

Ανάλογα με τη αλληλουχία και τη δομή τους τα πεπτίδια κατατάσσονται σε συγκεκριμένες οικογένειες και εμφανίζουν ένα ή και περισσότερα είδη βιοδραστηριότητας, τα οποία αναλύονται στη συνέχεια.

1.2.1 Οικογένειες αντιμικροβιακών πεπτιδίων

Οι ακόλουθες οικογένειες και υπο – οικογένειες είναι οι πιο χαρακτηριστικές στις οποίες έχει ταυτοποιηθεί οτι ανήκουν τα περισσότερα αντιμικροβιακά πεπτίδια.

1. Bacteriocins

Οι βακτηριοκίνες είναι βακτηριακά πεπτίδια που συντίθενται στο ριβόσωμα από συγκεκριμένα βακτήρια. Στην πλειοψηφία τους τα βακτήρια αυτά είναι θετικά κατά Gram, χωρίς να αποκλείεται η πιθανότητα παραγωγής τους και στα αρνητικά κατά Gram βακτήρια. (Ben Lagha, Haas, Gottschalk, & Grenier, 2017; Hammami, Zouhir, Le Lay, Ben Hamida, & Fliss, 2010)

Οι βακτηριοκίνες εμφανίζουν ποικιλία ως προς το μέγεθος, τη δομή, τους τρόπους δράσης, το εύρος δραστηριότητας και τους υποδοχείς των κυττάρων – στόχους. Η συνεχής ανάπτυξη της δομής και των τρόπων δράσης τους επιφέρει συνεχείς τροποποιήσεις στην ταξινόμησή τους (Ben Lagha et al., 2017).

Έχει παρατηρηθεί ότι πολλές βακτηριοκίνες δρουν συνεργατικά με τα συμβατά αντιβιοτικά (Ben Lagha et al., 2017; Cavera, Arthur, Kashtanov, & Chikindas, 2015; Wolska, Grześ, & Kurek, 2012) επιτρέποντας τη μείωση των βακτηριακών συγκεντρώσεων και τελικά τη μείωση των ανεπιθύμητων παρενέργειών τους.

2. Bombinins

Σε πολλές χώρες το δέρμα του βατράχου χρησιμοποείται για ιατρικούς σκοπούς ανα τους αιώνες. Το 1962 οι Kiss and Michl εντόπισαν την παρουσία αντιμικροβιακών και αιμολυτικών πεπτιδίων στο δέρμα του είδους *Bombina variegata*, κάτι το οποίο οδήγησε στην απομόνωση ενός αντιμικροβιακού πεπτιδίου μήκους 24 αμινοξικών καταλοίπων που το ονόμασαν «bombinin» (Csordas A & Michl H, 1970). Η απομόνωση της βομβινίνης σε συνδυασμό με την απομόνωση, αργότερα, των magainins από το είδος *Xenopus* (Zasloff, 1987) οδήγησε στην εξερεύνηση των πεπτιδίων σε όλα τα είδη αμφιβίων. Τα πεπτίδια αυτά τείνουν μάλιστα να εμφανίζουν και μικρό ποσοστό ομολογίας (Hancock & Chapple, 1999).

3. Cathelicidin

Οι κατηλιδινίνες είναι μικρά, κατιονικά αντιμικροβιακά πεπτίδια που διαφέρουν ως προς τη δομή, το μέγεθος και την αμινοξική ακολουθία. Το μήκος τους κυμαίνεται από 12 σε 80 αμινοξικά κατάλοιπα. Βρίσκονται τόσο στον άνθρωπο όσο και σε άλλα είδη όπως διάφορα είδη ψαριών αλλά και ζώα του αγροκτήματος (άλογα, γουρούνια, πρόβατα, κοτόπουλα, κουνέλια κλπ). Τα πεπτίδια αυτά, τα οποία ενεργοποιούνται πρωτεολυτικά, αποτελούν τμήμα του ανοσοποιητικού συστήματος πολλών σπονδυλωτών και δρουν ενάντια σε βακτήρια, ιούς και μύκητες. Παράλληλα, μπορούν να προκαλέσουν ειδικές αμυντικές αποκρίσεις στον ξενιστή. Αποθηκεύονται σε εκκριτικά κυστίδια ουδετερόφιλων και μακροφάγων και η εξωκυττάρια απελευθέρωσή τους πραγματοποιείται με την ενεργοποίηση των λευκοκυττάρων (Kościuczuk et al., 2012).

4. Cecropins

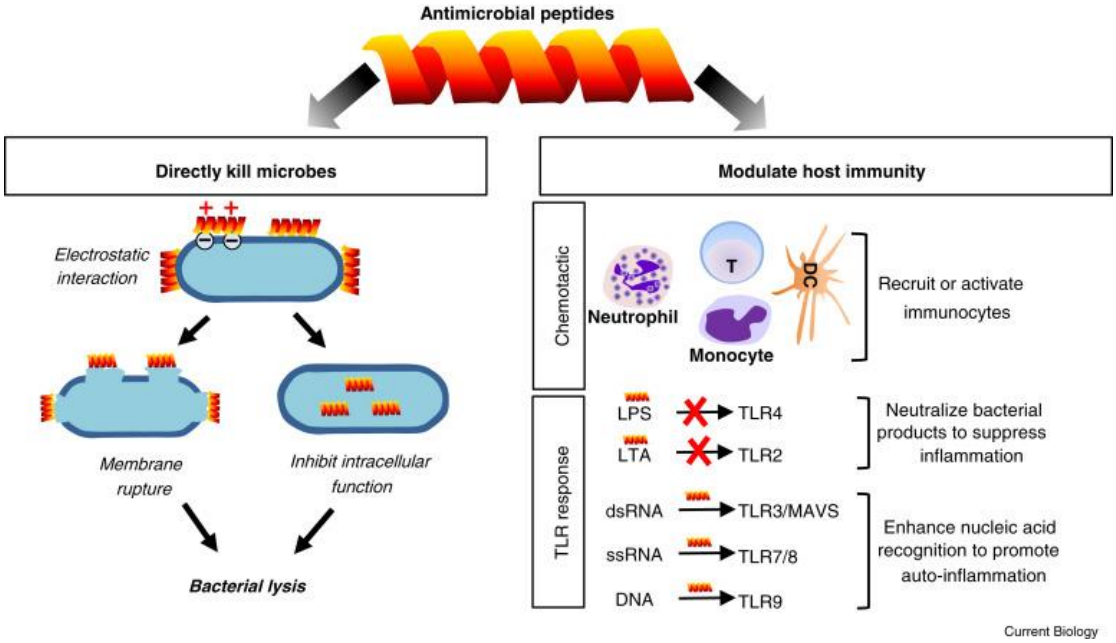
Οι κεκροπίνες περιλαμβάνουν δύο υπο – οικογένειες Cecropin A και Cecropin B. Τα πεπτίδια αυτής της οικογένειας απομονώθηκαν από το μεταξοσκώληκα *Hyalophora cecropia* αλλά αργότερα βρέθηκε και στα θηλαστικά. Στοχεύουν μη πολικές λιπιδικές μεμβράνες και έχουν τη δυνατότητα να λύουν καρκινικά κύτταρα καθώς το αμφιπαθικό αμινο – τελικό άκρο εμφανίζει κυτοτοξική αντικαρκινική δραστηριότητα. (Boohaker et al., 2012)

5. Cyclotide

Τα κυκλοτίδια είναι μικρά κυκλικά πεπτίδια, σε αυτό οφείλουν και το όνομα τους, με μήκος που κυμαίνεται μεταξύ 28 και 37 αμινοξέων. Ένα ιδιαίτερο χαρακτηριστικό τους είναι το μοτίβο κυστεϊνών που σχηματίζουν έξι συντηρημένα αμινοξικά κατάλοιπα κυστεϊνης. Αυτή η μοναδική κυκλική τοπολογία και διάταξη τριών δισουλφιδικών δεσμών τα καθιστά εξαιρετικά σταθερά σε θερμική, ενζυμική και χημική διάσπαση σε σχέση με άλλα πεπτίδια ίδιου μεγέθους(Gould, Ji, Aboye, & Camarero, 2011) Αρχικά απομονώθηκαν από τα φυτά αλλά μπορούν να παραχθούν και συνθετικά. Η κυριότερη δραστηριότητά τους περιλαμβάνει τη συμμετοχή τους στον αμυντικό μηχανισμό κατά των ξενιστών, το οποίο επιτυγχάνεται μέσω της αλληλεπίδρασής τους με την κυτταρική τους μεμβράνη(Barbeta, Marshall, Gillon, Craik, & Anderson, 2008; Poth, Colgrave, Lyons, Daly, & Craik, 2011) Παράλληλα, εμφανίζουν φαρμακευτική δράση, αντιμικροβιακή, αντικαρκινική και αντι - HIV.

6. Defensins

Τα πεπτίδια σε αυτή την οικογένεια εμφανίζουν κυρίως δομή β πτυχωτής επιφάνειας (β - sheet). Πρόκειται για πεπτίδια μικρά, πλούσια σε κυστεϊνες και απαντώνται σε θηλαστικά, τα έντομα, τα φυτά και τους μύκητες.(Boohaker et al., 2012) Οι οικογένειες defensins και cathelicidins, που αναφέρθηκαν παραπάνω, συνιστούν τις πιο σημαντικές καθώς σε αυτές ανήκουν πεπτίδια που διαταράσσουν τη μεμβράνη και απαντώνται στα σπονδυλωτά. Μεταξύ των κατιονικών κατάλοιπων των defensins και των αρνητικά φορτισμένων φωσφολιπιδικών ομάδων αναπτύσσονται ηλεκτροστατικές αλληλεπιδράσεις. Το αποτέλεσμα είναι να σχηματίζονται πόροι στη βακτηριακή μεμβράνη, καταστρέφουν την ακεραιότητα της και τελικά προάγεται η λύση των μικροβιακών στόχων (Ling-juan Zhang & Gallo, 2016).



Εικόνα 2 Βιολογική λειτουργία AMPs

DC dendritic cell, LPS lipopolysaccharide, LTA lipoteichoic acid, MAVS mitochondrial antiviral signalling protein

Μέσω ηλεκτροστατικών αλληλεπιδράσεων τα AMPs προσδένονται στη βακτηριακή μεμβράνη είτε για να την καταστρέψουν είτε για να εισέλθουν στο βακτήριο και να αναστείλουν την ενδοκυτταρική του λειτουργία. Ορισμένα AMPs προχωρούν σε ενεργοποίηση των ανοσοκυττάρων για να διαμορφώσουν την ανοσία του ξενιστή ή επηρεάζουν τους υποδοχείς αναγνώρισης (TLR) των μικροβιακών προϊόντων και των νουκλεϊκών οξέων που απελευθερώνονται μετά από βλάβη των ιστών.

1.2.2 Βιοδραστηριότητα πεπτιδίων

Στα μελετώμενα πεπτίδια έχει αποδοθεί ο γενικός χαρακτηρισμός «**αντιμικροβιακά**», ωστόσο, αποδίδονται και οι κάτωθι, πιο ειδικοί, χαρακτηρισμοί ανάλογα με το στόχο τους και την παρατηρούμενη βιοδραστηριότητά τους, η οποία μπορεί να είναι (Bahar & Ren, 2013b):

1. Αντιβακτηριακή (antibacterial)

Τα αντιβακτηριακά πεπτίδια είναι τα πιο ευρέως μελετημένα AMPs, στην πλειοψηφία τους είναι κατιονικά και αποτελούνται από 15 – 45 αμινοξικά κατάλοιπα. Στόχος τους είναι οι κυτταρικές μεμβράνες των βακτηρίων προκαλώντας αποσύνθεση στη δομή της λιπιδικής διπλοστιβάδας (Bahar & Ren, 2013b; Shai, 2002; L. Zhang, Rozek, & Hancock, 2001). Παράλληλα, έχουν την ικανότητα να προσδένονται τόσο στα λιπιδικά συστατικά, όσο και στις ομάδες φωσφολιπιδίων χάρη στην αμφιπαθική δομή που επίσης εμφανίζουν, με υδρόφοβες και υδρόφιλες περιοχές, αντίστοιχα (Bahar & Ren, 2013b; Jenssen, Hamill, & Hancock, 2006). Ένας επιπλέον τρόπος δράσης τους που έχει παρατηρηθεί κατά των βακτηρίων, εκτός από την απευθείας αλληλεπίδραση με τη μεμβράνη τους, είναι η αναστολή σημαντικών μονοπατιών μέσα στο κύτταρο, όπως η αντιγραφή του DNA και η πρωτεϊνοσύνθεση (Brogden, 2005).

2. Αντικαρκινική (anticancer)

Τα θεραπευτικά πεπτίδια που λειτουργούν ως αντικαρκινικοί παράγοντες αναμένεται να είναι επιλεκτικά εναντίον καρκινικών κυττάρων αφήνοντας ανεπηρέαστες τις υγιείς, φυσιολογικές σωματικές λειτουργίες (Gaspar, Veiga, & Castanho, 2013). Σε υγιή κύτταρα το συνολικό φορτίο της μεμβράνης είναι ουδέτερο λόγω της παρουσίας φωσφολιπιδίων διπολικού ιόντος (zwitterionic) όπως η φωσφατιδυλοχολίνη και η σφιγγομυελίνη, με τη φωσφατιδυλοσερίνη (phosphatidylserine PS) και τη φωσφατιδυλαιθανολαμίνη (phosphatidylethanolamine PE) να βρίσκονται στο εσωτερικό της πλασματική μεμβράνης. Σε αντίθεση, στα καρκινικά κύτταρα απουσιάζει αυτη η συμμετρία, εκθέτοντας την ανιονική φωσφατιδυλοσερίνη (PS) στο εξωτερικό της πλασματικής μεμβράνης, αυξάνοντας έτσι το συνολικό αρνητικό φορτίο (Bevers, Comfurius, & Zwaal, 1996). Τα σιαλικά αμινοξικά κατάλοιπα που συνδέονται στα γλυκολιπίδια και τις γλυκοπρωτεΐνες είναι μια επιπλέον πηγή αύξησης του αρνητικού φορτίου. Αρκετά AMPs διαθέτουν

κατιονική, αμφιφιλική δομή και έχουν έμφυτη κυτοτοξικότητα έναντι καρκινικών κυττάρων, γεγονός που τους επιτρέπει να στοχεύουν τις αρνητικά φορτισμένες μεμβράνες τους και επίσης τους παρέχει την ικανότητα δημιουργίας πόρων. Τα πεπτίδια αυτά συνδέονται στις μεμβράνες – στόχους, προκαλώντας αποπόλωση και κατά αυτόν τον τρόπο επιτυγχάνουν τα κυττοτοξικά τους αποτελέσματα (Boohaker et al., 2012). Τα αντικαρκινικά AMPs επιφέρουν το θάνατο των καρκινικών κυττάρων μέσω δύο βασικών μηχανισμών, την απόπτωση και τη νέκρωση. Η απόπτωση μπορεί να προκληθεί από τη διαταραχή της μεμβράνης των μιτοχονδρίων ενώ η νέκρωση μπορεί να είναι το αποτέλεσμα της στόχευσης των αρνητικά φορτισμένων μορίων που βρίσκονται στη μεμβράνη των καρκινικών κυττάρων η οποία και καταλήγει στην κυτταρική λύση (Boohaker et al., 2012).

3. Αντιμυκητιακή (antifungal)

Τα αντιμυκητιακά AMPS συντίθενται από πολικά και ουδέτερα αμινοξικά κατάλοιπα αλλά η δομή τους δε φαίνεται να σχετίζεται με τον τύπο των κυττάρων που στοχέυουν. Η αντιμυκητιακή δράση ορισμένων πεπτιδίων έγκειται στην εξολόθρευση των μυκήτων είτε στοχεύοντας το κυτταρικό τους τοίχωμα (A. J. De Lucca, Bland, Jacks, Grimm, & Walsh, 1998; Anthony J. De Lucca & Walsh, 1999) είτε τα συστατικά του ενδοκυττάρου (Lee et al., 1999). Ένα από τα βασικότερα συστατικά του κυτταρικού τοιχώματος των μυκήτων το οποίο δεσμεύουν τα αντιμυκητιακά AMPs είναι η χιτίνη. Τελικά, η καταστροφή των κυττάρων στόχων επιτυγχάνεται μέσω της αύξησης της διαπερατότητας της πλασματικής μεμβράνης (van der Weerden, Hancock, & Anderson, 2010), μέσω της διαταραχής της μεμβρανικής τους ακεραιότητας (Lehrer, Szklarek, Ganz, & Selsted, 1985; Terras et al., 1992), είτε μέσω της απευθείας δημιουργίας πόρων (Moerman et al., 2002).

4. Αντιϊκή (antiviral)

Τα αντιϊκά AMPs στοχεύουν τόσο DNA όσο και RNA ιούς. Ενσωματώνονται είτε στο περίβλημα του ιού είτε στην μεμβράνη του κυττάρου – ξενιστή, με αποτέλεσμα την αδρανοποίησή του(Bahar & Ren, 2013b; Bastian & Schäfer, 2001; Horne et al., 2005) και οι μέθοδοι με τις οποίες δρουν έναντι των ιών ποικίλουν. Σύμφωνα με πρόσφατες μελέτες, τα πεπτίδια αυτά θεωρούνται θεραπευτικά χάρη στην ικανότητά τους να αποτρέπουν την προσκόληση των ιών, να προλαμβάνουν την ενσωμάτωσή τους σε κύτταρα – ξενιστές, να διακόπτουν τις σηματοδοτικές διεργασίες των ιών καθώς επίσης, έχει παρατηρηθεί να

αναστέλλουν την αντιγραφή τους σε κύτταρα – ξενιστές με διαδικασίες που μπορεί να εμπλέκουν σειρές ενζύμων όπως η DNA πολυμεράση, η αντίστροφη μεταγραφάση, η ιντεγκράση και η πρωτεάση (Castel, Chtéoui, Heyd, & Tordo, 2011; Chang & Yang, 2013).

5. Εντομοκτόνα (insecticidal)

Τα επιβλαβή έντομα είναι υπεύθυνα για τη μεταφορά και τη μετάδοση νοσών. Μια απο τις μεγαλύτερες και πιο διαφοροποιημένες ομάδες ευκαρυωτών που καταστρέφουν την αγροτική παραγωγή, κατα ένα μεγάλο ποσοστό, και προκαλούν μεγάλες απώλειες σε όλο τον κόσμο είναι τα Κολεόπτερα (*Coleoptera*). Κατα συνέπεια ο έλεγχος των επιβλαβών αυτών οργανισμών με βιολογικές μεθοδους είναι μια σημαντική εναλλακτική μέθοδος για την αποτελεσματική διαχείριση τους.

Συγκεκριμένα, εκτατεμένες έρευνες επικεντρώνονται στην εύρεση μεθόδων και κατάλληλων πεπτιδίων, οι οποίες να είναι φιλικές προς το περιβάλλον (De Lima et al, 2007). Η τεχνολογία του ανασυνδυασμένου DNA, για παράδειγμα, επιτρέπει την εκμετάλλευση των εντομοκτόνων ιδιοτήτων των εντομοπαθογόνων οργανισμών. Σε αντίθεση με τα χημικά προϊόντα, κατάλληλα εντομοκτόνα πεπτίδια ή πολυπεπτίδια αναζητώνται ώστε να εμφανίζουν τοξικότητα μόνο στους επιβλαβείς οργανισμούς και να μην καταστρέφουν τη χλωρίδα και την πανίδα. Μεταβολίτες, φερομόνες, μικροβιολογικοί παράγοντες, μηχανισμοί άμυνας των φυτών και γονίδια που μεταφράζουν τοξικά πεπτίδια και πρωτεΐνες περιλαμβάνονται στα βιοεντομοκτόνα.

Υπάρχουν δυο κατηγορίες εντομοκτόνων τοξινών. Η πρώτη αφορά τις peptide-like τοξίνες που προέρχονται από το δηλητήριο κάποιων σκορπιών και αραχνών με μοριακό βάρος 3-10 kDa. Τα πεπτίδια αυτά αλληλεπιδρούν με τα ιοντικά κανάλια, Na^+, K^+, Ca^+, Cl^-, πάνω στις κυτταρικές μεμβράνες και αποτελούνται από μια αλυσίδα με πολλά κυστεϊνικά κατάλοιπα που σχηματίζουν ενδομοριακές δισουλφυδικές γέφυρες (De Lima et al, 2007). Η δεύτερη αφορά τοξίνες υψηλού μοριακού βάρους, που αγγίζει και τα 1000 κατάλοιπα, όπως για παράδειγμα οι λατροξίνες από το δηλητήριο της αράχνης *Latrodectus*.

Οι σκορπιοί είναι μια ενδιαφέρουσα ομάδα οργανισμών με τοξίνες που αποτελούντα απο 23-78 κατάλοιπα. Στη συνήθη στερεοδιάταξη αποτελούνται απο μια α-έλικα πακεταρισμένη απέναντι από μια β-πτυχωτή επιφάνεια με 3 κλώνους που σταθεροποιούνται απο 4 δισουλφυδικούς δεσμούς. Ανάλογα με το μηχανισμό δράσης

τους χωρίζονται σε α- και β-τοξίνες, με τις πρώτες να προσδένονται σε κανάλια ιωδίου με μεγάλη συγγένεια και τις δεύτερες να μεταβάλλουν την τάση από την οποία εξαρτάται η ενεργοποίηση του καναλιού.

Εκτός από τους σκορπιούς, τα φυτά επίσης παράγουν τοξικές χημικές ενώσεις υπεύθυνες για τους μηχανισμούς αυτοάμυνας εντόμων. Οι φυτικές αμυντοσίνες (Plant Defensins) είναι αντιμικροβιακές πρωτεΐνες με οχτώ συντηρημένες κυστεϊνες και τέσσερις δισουλφυδικές γέφυρες. Οι πρωτεΐνες αυτές επιτίθενται στις αμυλάσες των λεπιδόπτερων αναστέλλοντας την τροφή (Kanchiswamy et al., 2010). Επίσης, μέλη της οικογένειας των λεπιδόπτερων και των κολεόπτερων βλάπτονται και από τις κυκλοτίδες των φυτών (plant cyclotides), οι οποίες αποτελούνται απο 30 αμινοξέα με τρεις συντηρημένους δισουλφυδικούς δεσμούς που συνδένται σε ένα μοτίβο «κυστεϊνικού κόμπου».

6. <u>Αντιπαρασιτική (antiparasitic)</u>

Τα αντιπαρασιτικά πεπτίδια αποτελούν τη μικρότερη ομάδα των αντιμικροβιακών πεπτιδίων. Ο τρόπος με τον οποίο δρουν είναι παρόμοιος με τις παραπάνω κατηγορίες, σκοτώνοντας παρασιτικούς μικροοργανισμούς αλληλεπιδρώντας απευθείας με την κυτταρική τους μεμβράνη (Park, Jang, Lee, & Hahm, 2004).

ΚΕΦΑΛΑΙΟ 2

Τα τελευταία χρόνια το ενδιαφέρον για τις θεραπευτικές ιδιότητες των αντιμικροβιακών πεπτιδίων και τη φαρμακευτική τους χρήση συνεχώς αυξάνεται. Ωστόσο, η πειραματική ταυτοποίηση και ο σχεδιασμός τους είναι ακριβές, χρονοβόρες διαδικασίες και απαιτούν την κατανάλωση αρκετών πόρων. Το γεγονός αυτό δημιούργησε την ανάγκη ανάπτυξης υπολογιστικών μεθόδων πρόβλεψης AMPs με βάση τα ιδιαίτερα χαρακτηριστικά που εμφανίζουν. Οι περισσότεροι ερευνητές επικεντρώνονται στην ανακάλυψη νέων εργαλείων in silico καθώς μέσω της υπολογιστικής προσέγγισης μπορεί να επιταχυνθεί η διαδικασία εφεύρεσης και σχεδιασμού αντιμικροβιακών φαρμάκων.

Οι μέθοδοι αφορούν την εκπαίδευση ενός μοντέλου ταξινόμησης, το οποίο με βάση ένα γνωστό σύνολο δεδομένων εκπαίδευσης (training dataset) να μπορεί να προβλέπει τη λειτουργία ενός άλλου συνόλου δεδομένων που είναι άγνωστο (test dataset). Υπάρχουν και μέθοδοι πρόβλεψης που στηρίζονται στη στοίχιση ακολουθιών οι οποίες επιτυγχάνουν υψηλή ακρίβεια πρόβλεψης. Το αρνητικό σε αυτή την περίπτωση είναι οτι δεν μπορούν να προβλέψουν όλες τις αλληλουχίες γιατί η ταξινόμηση στην στοίχιση ακολουθιών βασίζεται στα HSPs scores, τα οποία αντιπροσωπεύουν το βαθμό ομοιότητας μεταξύ των ακολουθιών με τη χρήση του BLASTP. Οπότε, αν η αλληλουχία στο test dataset δεν έχει καμια σχέση με κάποια αλληλουχία από το training dataset, τα HSPs scores δε θα παραχθούν και κατ' επέκταση η ταξινόμηση δε θα εφαρμοστεί σε αυτή τη συγκεκριμένη αλληλουχία (Ng, Rosdi, & Shahrudin, 2015). Στην πλειοψηφία τους οι υπολογιστικές μέθοδοι βασίζονται στις αρχές της τεχνητής νοημοσύνης (artificial intelligence) και στην ανάπτυξη αλγόριθμων μηχανικής εκμάθησης (machine learning) με κύριο αντικείμενο μελέτης τα μοναδικά χαρακτηριστικά που εμφανίζουν τα αντιμικροβιακά πεπτίδια. Η απόδοση και η αξιοπιστία της κάθε μεθόδου εκτιμάται με βάση κάποιες συνήθεις μετρικές αξιολόγησης οι οποίες συνοψίζονται παρακάτω.

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	Αληθώς Θετικά (**TP**)	Αρνητικώς Θετικά (**FP**)
Αρνητική παρατήρηση	Αρνητικώς Αρνητικά (**FN**)	Αληθώς Αρνητικά (**TN**)

- **Ευαισθησία (Sensitivity)**. Το ποσοστό των παρατηρηθέντων θετικών που προβλέφθηκαν σωστά να είναι θετικά.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Ειδικότητα (Specificity)**. Το ποσοστό των παρατηρηθέντων αρνητικών που προβλέφθηκαν σωστά να είναι αρνητικά.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

- **Ακρίβεια (Accuracy)**. Η ακρίβεια καθορίζεται με βάση το πόσο κοντά βρίσκεται μια μέτρηση σε μια υπάρχουσα τιμή, δηλαδή το κατά πόσον η μέτρηση ανταποκρίνεται στην αλήθεια. Το ποσοστό του λάθους μεταξύ της αποδεκτής (accepted) και της πειραματικής (experimented) τιμής ορίζεται ως:

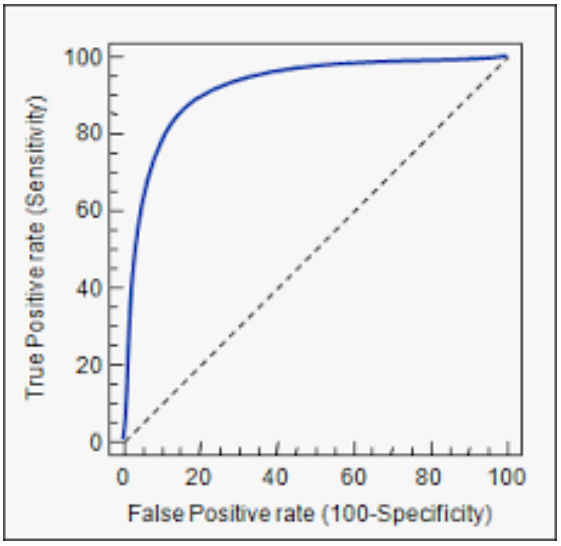
  %error = (accepted – experimented) / accepted * 100%.

- **Συντελεστής συσχέτισης Matthew's (Matthew's correlation coefficient, MCC)**. Η μετρική MCC, εκφράζει την ποιότητα ενός δυαδικού ταξινομητή και λαμβάνει υπόψη τα αληθώς και ψευδώς θετικά και αρνητικά αποτελέσματα (true positives TP, true negatives TN, false positives FP, false negatives FN) επιστρέφοντας μια τιμή μεταξύ -1 και +1 βάσει της εξίσωσης:

$$MCC = \frac{TP * TN - FP * FN}{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))^{1/2}}$$

Η τέλεια πρόβλεψη αντιπροσωπεύεται με MCC = +1, και η απόλυτη διαφωνία μεταξύ πρόβλεψης και παρατήρησης αντιπροσωπεύεται με MCC = -1.

- Καμπύλη ROC (Receiver Operating Characteristic Curve)



Εικόνα 3 Καμπύλη ROC – AUC

Η καμπύλη ROC αναφέρεται ως η καμπύλη που συνδέει τα σημεία που έχουν συντεταγμένες την ευαισθησία (Sensitivity) και τη συμπληρωματική τιμή της ειδικότητας (Specificity) για συγκεκριμένες τιμές που λαμβάνει η μεταβλητή που εξετάζεται. Όσο μεγαλύτερη είναι η επιφάνεια κάτω από την καμπύλη τόσο μεγαλύτερη είναι και η διαγνωστική αξία της μεθόδου πρόβλεψης. Η περιοχή αυτή χαρακτηρίζεται ως Area Under the Curve (AUC).

2.1 Μηχανική μάθηση (Machine Learning)

Η δημιουργία μοντέλων ή προτύπων από ένα σύνολο δεδομένων, από ένα υπολογιστικό σύστημα, ονομάζεται **Μηχανική Μάθηση (Machine Learning)**. Πρόκειται για μια διαδικασία όπου μια μηχανή εκπαιδεύεται και βελτιώνει την απόδοση της κατά την εκτέλεση μιας συγκεκριμένης λειτουργίας χωρίς να χρειάζεται να προγραμματιστεί εκ νέου, αξιοποιώντας προηγούμενη γνώση και εμπειρία. Η μηχανική μάθηση μελετά αλγόριθμους που βελτιώνουν τη συμπεριφορά τους σε κάποια εργασία που τους έχει ανατεθεί χρησιμοποιώντας την εμπειρία τους. Ένας από τους επίσημους ορισμούς που έχουν αποδοθεί είναι ο εξής:

«Ένα υπολογιστικό πρόγραμμα θεωρείται οτι μαθαίνει από την εμπειρία E σε σχέση με μια κατηγορία εργασιών T και μια μετρική απόδοσης P, αν η απόδοσή του σε εργασίες της T, όπως μετράται από την P, βελτιώνεται με την εμπειρία E» (Tom M. Mitchell, 1997)

Ανάλογα με τη φύση του εκάστοτε προβλήματος έχουν αναπτυχθεί πολλές τεχνικές μηχανικής μάθησης και χωρίζονται σε κυρίως σε δυο κατηγορίες:

- Μάθηση χωρίς επίβλεψη (unsupervised machine learning)
- Μάθηση με επίβλεψη (supervised machine learning)

2.1.1 Μηχανική εκμάθηση χωρίς επίβλεψη (Unsupervised machine learning)

Ένας αλγόριθμος, στη μη επιβλεπόμενη μάθηση, κατασκευάζει ένα μοντέλο όπου υπάρχουν μόνο οι παρατηρήσεις και αποτελούν τα δεδομένα εισόδου (X) χωρίς να γνωρίζει τις αντίστοιχες μεταβλητές εξόδου. Στόχος είναι να μοντελοποιήσει την υποκείμενη δομή και την κατανομή στα δεδομένα ώστε να «μάθει» περισσότερα για αυτά. Το σύστημα δηλαδή πρέπει μόνο του να ανακαλύψει ποιες συσχετίσεις υπάρχουν σε ένα σύνολο δεδομένων και ποιες ομάδες μπορούν πιθανόν να προκύψουν από αυτό, δημιουργώντας τελικά κάποια πρότυπα, τα οποία δεν είναι σίγουρο ούτε αν υφίστανται πραγματικά ούτε πόσα και ποια ακριβώς είναι. Από τον ορισμό γίνεται αντιληπτό ότι δεν υπάρχουν σωστές και λανθασμένες απαντήσεις, αντίθετα οι αλγόριθμοι ανακαλύπτουν και παρουσιάζουν την ενδιαφέρουσα δομή των δεδομένων ανάλογα με το δικό τους σχεδιασμό. Τα κυριότερα προβλήματα που βρίσκει εφαρμογή η μέθοδος αυτή είναι της Ομαδοποίησης (Clustering) και της Ανάλυσης Συσχετισμών (Association Analysis). Η

ομαδοποίηση επιτρέπει τον αυτόματο διαχωρισμό των δεδομένων σε ομάδες ανάλογα με το βαθμό ομοιότητάς του. Οι αλγόριθμοι ιεραρχικής ομαδοποίησης (hierarchical clustering) και k-means cluster analysis είναι δυο ευρέως εφαρμοζόμενες, μη επιβλεπόμενες προσεγγίσεις. Στην ανάλυση συσχετισμών στόχος είναι η εύρεση κάποιου κανόνα όπου περιγράφει κατάλληλα ένα μεγάλο ποσοστό των δεδομένων, όπως για παράδειγμα ότι άνθρωποι που τείνουν να εμφανίζουν μια πάθηση τείνουν επίσης να έχουν και κάποια κοινή καθημερινή συνήθεια. Ο αλγόριθμος Apriori, αντίστοιχα, είναι από τους πιο δημοφιλής εδώ.

2.1.2 Μηχανική εκμάθηση με επίβλεψη (Supervised machine learning)

Στη μηχανική μάθηση με επίβλεψη το σύστημα «μαθαίνει» μια συνάρτηση που προκύπτει από ένα σύνολο δεδομένων και η οποία περιγράφει ένα μοντέλο. Με χρήση του κατάλληλου αλγόριθμου γίνεται γνωστή η συνάρτηση Y = f(X) που αντιστοιχεί την είσοδο στην έξοδο, όπου X οι μεταβλητές εισόδου και Y η μεταβλητή εξόδου. Απώτερος στόχος είναι να επιτευχθεί όσο το δυνατόν καλύτερη προσέγγιση αυτής της συνάρτησης, ώστε με νέα άγνωστα δεδομένα εισόδου X να μπορούν να προβλεφθούν οι μεταβλητές εξόδου Y για αυτά τα δεδομένα. Οι σημασμένες μεταβλητές εισόδου X αποτελούν το **σύνολο δεδομένων εκπαίδευσης** (**training dataset**) ενώ οι άγνωστες μεταβλητές εισόδου X που καλείται να προβλεφθούν από το μοντέλο αποτελούν το **δοκιμαστικό σύνολο δεδομένων** (**test dataset**). Ο αλγόριθμος πραγματοποιεί επαναλαμβανόμενες προβλέψεις στα training data και, δεδομένου ότι οι σωστές απαντήσεις είναι γνωστές, συνεχώς βελτιώνεται μέχρι να επιτευχθεί το καλύτερο επίπεδο απόδοσης. Η πλειοψηφία των πρακτικών μηχανικής μάθησης είναι με επίβλεψη και χρησιμοποιούνται συχνά σε προβλήματα **Ταξινόμησης (Classification)** και **Παλινδρόμησης (Regression)**.

Στην ταξινόμηση η μεταβλητή εξόδου ανήκει σε μια κατηγορία πχ ασθενής ή υγιής. Συγκεκριμένα, τα δεδομένα εισόδου κατανέμονται σε δύο ή περισσότερες κλάσεις και το μοντέλο που κατασκευάζει η μηχανή τοποθετεί τα δεδομένα σε μια ή περισσότερες κλάσεις. Η παλινδρόμηση είναι μια τεχνική μοντελοποίησης με ευρεία εφαρμογή στη στατιστική, όπου ερευνάται η συσχέτιση μεταξύ μιας εξαρτώμενης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών. Στην παλινδρόμηση η μεταβλητή εξόδου λαμβάνει συνεχείς και όχι διακριτές τιμές πχ βάρος. Τα πιο δημοφιλή παραδείγματα αλγορίθμων επιβλεπόμενης μηχανικής μάθησης είναι:

- **Γραμμική παλινδρόμηση (linear regression)**, για τα προβλήματα παλινδρόμησης. Απαίτηση του μοντέλου είναι η εξαρτημένη μεταβλητή Y να είναι γραμμικός συνδυασμός των ανεξάρτητων μεταβλητών. Η απλή γραμμική παλινδρόμηση είναι η απλούστερη περίπτωση όπου υπάρχει μόνο μια ανεξάρτητη μεταβλητή X, μια εξαρτημένη μεταβλητή Y και δύο παράμετροι β_0 , β_1 και το μοντέλο έχει τελικά τη μορφή:

  $Y = \beta_0 + \beta_1 * X$

- **Random Forest**, για προβλήματα τόσο παλινδρόμησης όσο και ταξινόμησης. Πρόκειται για αλγόριθμο ικανό να κατηγοριοποιεί μεγάλες ποσότητες δεδομένων με ακρίβεια. Βασίζεται στην κατασκευή δένδρων απόφασης (Decision Trees) κατά την διαδικασία εκπαίδευσης και στην έξοδο παράγει την κλάση που αποτελεί την πιο αντιπροσωπευτική κλάση εξόδου από τα μεμονωμένα δένδρα. Τα Random Forests συνδυάζουν δενδρικές προβλέψεις, όπου το κάθε δένδρο εξαρτάται από τις τιμές ενός τυχαίου διανύσματος το όποιο εχει δειγματιστεί ανεξάρτητα με την ίδια κατανομή για όλα τα δέντρα του δάσους. Τα μεμονωμένα δέντρα απόφασης εμφανίζουν υψηλή διακύμανση και μεροληψία ενώ αντίθετα τα Random Forests προσπαθούν να περιορίσουν τα προβλήματα αυτά λαμβάνοντας το μέσο όρο ώστε να βρουν μια ισορροπία μεταξύ των δυο άκρων. Πρόκειται για ένα απλό εργαλείο δεδομένου ότι δεν απαιτείται η ρύθμιση πολλών παραμέτρων, αφού μπορούν να χρησιμοποιηθούν έχοντας προεπιλεγμένες ρυθμίσεις παραμέτρων. Το γεγονός ότι παρέχουν αποτελεσματικές μεθόδους εκτίμησης δεδομένων που λείπουν και εξακολουθούν να διατηρούν την ακρίβεια των αποτελεσμάτων, αποτελεί ένα από τα σημαντικά πλεονεκτήματά τους.

- **Support Vector Machines (SVMs)**, για τα προβλήματα ταξινόμησης και παλινδρόμησης. Τα μοντέλα αυτά αναλύονται στη συνέχεια καθώς σε αυτά ανήκουν οι βασικότεροι αλγόριθμοι υπολογιστικής προσέγγισης για την πρόβλεψη της βιοδραστηριότητας των πεπτιδίων.

2.2 Support Vector Machine (SVM)

Support Vector Machines (SVMs) ή **Μηχανές Διανυσμάτων Υποστήριξης (ΜΔΥ)** είναι μια από τις πιο συνηθισμένες μεθόδους που εφαρμόζονται στη μηχανική μάθηση με επίβλεψη και συσχετίζεται με αλγόριθμους ανάλυσης δεδομένων για την επίλυση προβλημάτων ταξινόμησης και παλινδρόμησης. Προτάθηκαν τη δεκατία του 1990 από τον Vladimir Vapnik και τους συνεργάτες του όπου θεμελίωσαν τις βασικές αρχές της μεθόδου και στηρίζονται στη θεωρία της Στατιστικής Μάθησης. Τα τελευταία χρόνια προσελκύουν όλο και μεγαλύτερο ενδιαφέρον χάρη στην ικανότητά τους να παρέχουν καλύτερη απόδοση ως προς την ακρίβεια της ταξινόμησης σε σύγκριση με άλλους αλγόριθμους μάθησης. Η μέθοδος δυαδικής ταξινόμησης όπου δέχεται ως είσοδο ένα σύνολο δεδομένων εκπαίδευσης (training dataset) τα οποία ανήκουν σε μια από τις δυο κατηγορίες και κατασκευάζει ένα μοντέλο όπου νέα άγνωστα δεδομένα κατανέμονται σε μια από τις δυο κατηγορίες ονομάζεται γραμμικός δυαδικός ταξινομητής (binary linear classifier). Κάθε δείγμα των δεδομένων αντιπροσωπεύει ένα σημείο σε μια περιοχή μεγαλύτερων διαστάσεων και ο αλγόριθμος προσπαθεί να ορίσει το βέλτιστο διαχωριστικό υπερεπίπεδο (**hyperplane**) ανάμεσα στις δυο κλάσεις που ανήκουν τα δεδομένα, επιδιώκοντας η απόσταση των δειγμάτων και στις δυο κλάσεις από το επίπεδο αυτό να είναι η μέγιστη δυνατή. Η τιμή αυτή αναφέρεται ως **margin**. Σε ένα χώρο δυο διαστάσεων τα δεδομένα χωρίζονται χρησιμοποιώντας απλά μια γραμμή, σε ένα χώρο τριών διαστάσεων τα δεδομένα χωρίζονται χρησιμοποιώντας διαφορετικά επίπεδα κοκ.

Κάθε σημείο είναι της μορφής

$$\{(x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4), ..., (x_n,y_n)\}$$

όπου y_n = 1 / -1 μια σταθερά που υποδεικνύει την κλάση που ανήκει το σημείο x_n και

n = αριθμός των δειγμάτων.

Κάθε x_n είναι ένα p-διάστατο διάνυσμα. Σκοπός είναι να βρεθεί η μέγιστη τιμή για το margin που χωρίζει την ομάδα των σημείων x_i όπου το y_i = 1 από την ομάδα όπου το y_i = -1, και ορίζεται κατά τέτοιο τρόπο ώστε η απόσταση μεταξύ της διαχωριστικής γραμμής και του πλησιέστερου σημείου x_i από κάθε ομάδα να μεγιστοποιείται. Κάθε hyperplane μπορεί να γραφεί ως ενα σύνολο από σημεία x που ικανοποιούν την παρακάτω εξίσωση:

$$w * x + b = 0$$

όπου b είναι ένα βαθμωτό μεγεθος και w είναι το κάθετο διάνυσμα στο hyperplane (DURGESH K. SRIVASTAVA & LEKHA BHAMBHU, n.d.). Για κάθε σημείο πάνω στο hyperplane η απόσταση μεταξύ του αρχικού και του hyperplane είναι b/‖w‖. Η προσθήκη της παραμέτρου μετατόπισης b επιτρέπει την αύξηση του margin.Τα στοιχεία που ανήκουν στην κλάση -1 θεωρείται οτι ικανοποιούν την εξίσωση

$$w * x + b = -1$$

και οριζουν το H_1 hyperplane, ενω η απόσταση μεταξύ του αρχικού και του hyperplane είναι |-1-b| / ‖w‖.

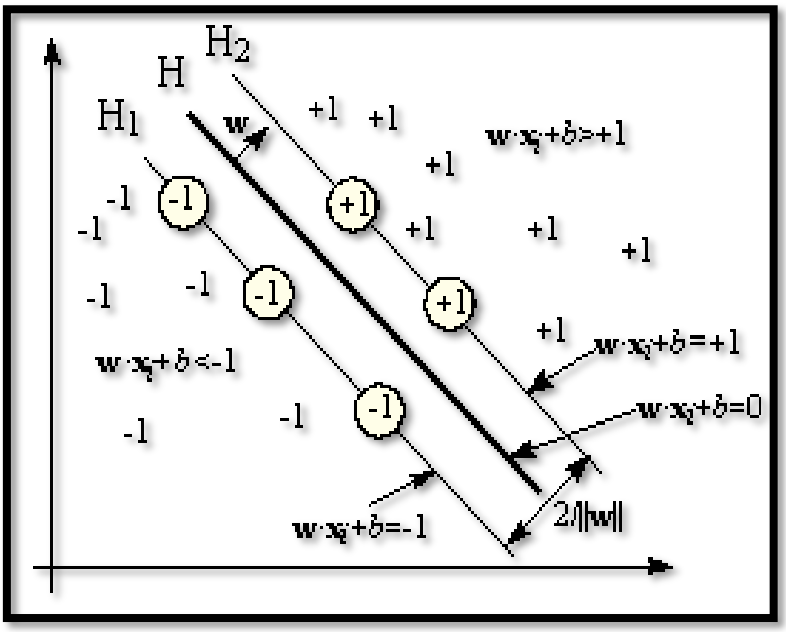
Ομοίως, τα μοτίβα των στοιχείων που ανήκουν στην κλάση +1, θεωρείται οτι ικανοποιούν την εξίσωση

$$w * x + b = +1$$

και οριζουν το H_2 hyperplane, ενω η απόσταση μεταξύ το αρχικού και του hyperplane είναι

$$\frac{|+1 - b|}{||w||}$$

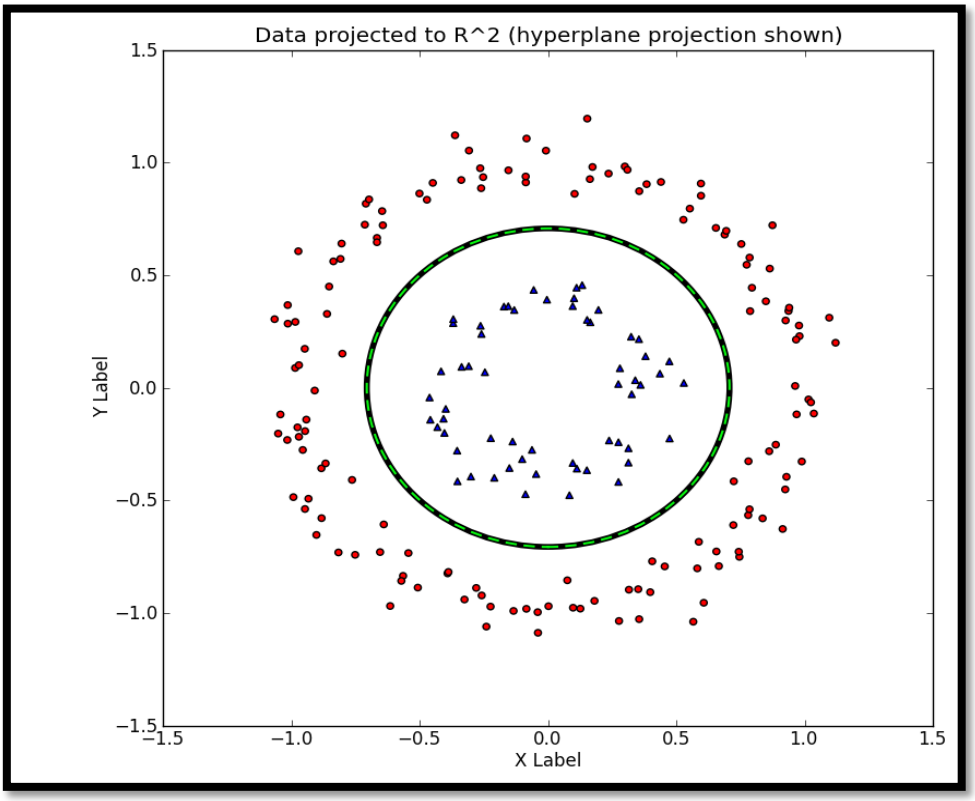
Τα H, H_1, H_2 είναι παράλληλα και δεν υπάρχουν μοτίβα από το δεδομένα εκπαίδευσης που να βρίσκονται μεταξύ των H_1, H_2. Τελικά, σύμφωνα με τα παραπάνω η απόσταση μεταξύ των hyperplanes H_1, H_2, η τιμή margin δηλαδή, είναι $\frac{2}{||w||}$.



Εικόνα 4 Ο βέλτιστος διαχωρισμός υπερεπιπέδου (hyperplane)

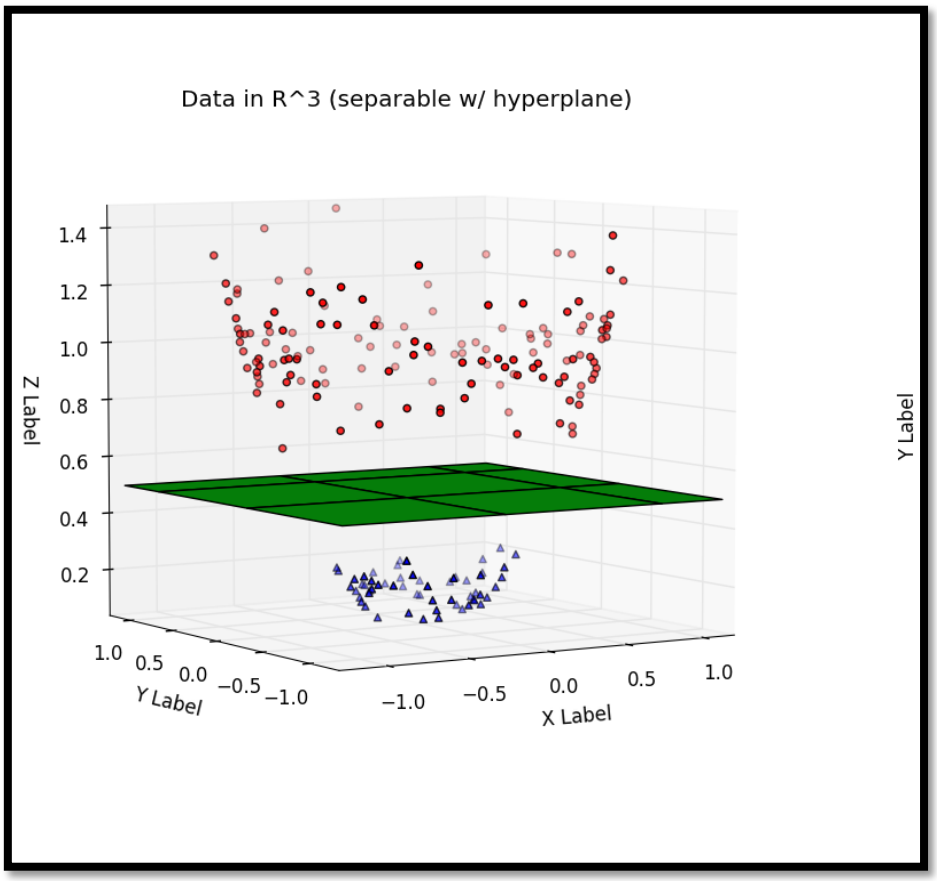
Kernel Trick

Υπάρχουν περιπτώσεις ωστόσο όπου τα δεδομένα δεν μπορούν να χωριστούν σε 2 κατηγορίες γραμμικά.



Εικόνα 5 Μη γραμμικώς διαχωρίσιμο σύνολο δεδομένων

Οπότε, ένας τρόπος διαχείρισης αυτής της κατάστασης είναι να μεταφερθούν τα δεδομένα σε ένα χώρο τριών διαστάσεων και να χωριστούν γραμμικά από ένα επίπεδο.



Εικόνα 6 Μεταφορά συνόλου δεδομένων σε 3D χώρο

Αυτό συμβαίνει με τη χρήση της λεγόμενης **συνάρτησης kernel** (kernel function). Για δυο δείγματα x και x', η συνάρτηση kernel ανάλογα με τον τύπο της μπορεί να χαρακτηριστεί ως

- Γραμμική : $K(x,x') = x^T x'$
- Μη γραμμική
- Πολυωνυμική : $K(x,x') = (1+ x^T x')^d$ για κάθε d>0 όπου το d εκφράζει το βαθμό του πολυωνύμου.
- Σιγμοειδής
- Συνάρτηση βασιζόμενη στην ακτίνα (radial basis function, RBF)

$$K(x, x') = \exp\left(-\frac{\left||x-x'|\right|^2}{2\sigma^2}\right), \text{ για } \sigma>0$$

με την τελευταία να είναι και η πιο δημοφιλής σε χρήση καθώς ο αντίστοιχος χώρος διανυσμάτων (feature vector) έχει άπειρη διάσταση.

2.3 Υπολογιστικές μέθοδοι προσδιορισμού βιοδραστηριότητας πεπτιδίων

❖ AmPEP

Η AmPEP (Bhadra, Yan, Li, Fong, & Siu, 2018) είναι μια μέθοδος ταξινόμησης που αναπτύχθηκε με τον αλγόριθμο Random Forest (RF) και βασίζεται στη μελέτη του τρόπου με τον οποίο είναι κατανεμημένα τα αμινοξικά κατάλοιπα κατά μήκος της ακολουθίας. Το σύνολο των θετικών δεδομένων συλλέχθηκε από τις βάσεις δεδομένων APD3, CAMPR3, LAMP και το σύνολο των αρνητικών δεδομένων δημιουργήθηκε συλλέγοντας ακολουθίες απο τη Uniprot, μήκους 5 ως 255 αμινοξικών καταλοίπων, και αφαιρώντας εν συνεχεία όσες περιείχαν ως χαρακτηρισμό οποιαδήποτε από τις λέξεις-κλειδιά «AMP», «antibiotic», «antifungal», «anticancer», «antiviral», «toxic», «membrane» και «defensive». Για τις ανάγκες της μηχανικής μάθησης, οι αμινοξικές ακολουθίες μετατράπηκαν σε αριθμητικούς περιγραφείς που ονομάζονται descriptors, και οι οποίοι χαρακτηρίζουν διαφορετικές ιδιότητες των πεπτιδίων προκειμένου να βρεθεί το πρότυπο που ακολουθούν τα χαρακτηριστικά των αντιμικροβιακών πεπτιδίων καθώς και οι βέλτιστες παράμετροι του μοντέλου που θα εξασφαλίζουν υψηλή ακρίβεια. Οι εφτά φυσικοχημικές ιδιότητες που λήφθησαν υπόψη ήταν η υδροφοβικότητα, η κανονικοποιημένη ένταση van der Waals, η πολικότητα, το φορτίο, η δευτεροταγής δομή, η πολωσιμότητα και η προσβασιμότητα σε διαλύτη. Τελικά, συνολικά αξιολογήθηκαν 19 ταξινομητές RF με διαφορετική αναλογία θετικών:αρνητικών δειγμάτων με 10-fold cross-validation. Το βέλτιστο μοντέλο είχε αναλογία 1:3 και τα αποτελέσματα των μετρικών αξιολόγησης ηταν:

Accuracy	MCC	AUC-ROC	Kappa statistic
96%	0.9	0.99	0.9

- **AMPA**

AMPA (Torrent et al., 2012) ονομάζεται μια διαδικτυακή εφαρμογή που αναπτύχθηκε έχοντας ως επίκεντρο το σχεδιασμό νέων αντιμικροβιακών φαρμάκων μέσω της αξιολόγησης των αντιμικροβιακών domains των πρωτεϊνών. Ο αλγόριθμος βασίζεται σε μια αντιμικροβιακή κλίμακα που προέκυψε μελετώντας την AMP bactenecin 2A, στην οποία έχουν προσδιοριστεί οι αντιμικροβιακές τιμές IC_{50} για όλες τις αμινοξικές αντικαταστάσεις σε κάθε θέση. IC_{50} (inhibitory concentration), το μισό της μέγιστης ανασταλτικής συγκέντρωσης, είναι ένα μέτρο που εκφράζει την ισχύ μιας ουσίας στην αναστολή μιας συγκεκριμένης βιοχημικής ή βιολογικής λειτουργίας. Όσο χαμηλότερη είναι η τιμή IC_{50}, τόσο πιο ισχυρό είναι ένα μόριο. Από τα δεδομένα υπολογίστηκε ένα αντιμικροβιακός δείκτης που παρέχει μια ικανοποιητική εκτίμηση για την τάση που έχουν τα αμινοξικά αυτά κατάλοιπα να βρίσκονται σε μια αντιμικροβιακή ακολουθία, με τα αμινοξέα με χαμηλό δείκτη να παρουσιάζουν την υψηλότερη πιθανότητα, δεδομένου ότι οι χαμηλές τιμές IC_{50} αντιστοιχούν σε υψηλή δραστηριότητα. Τελικά, ο αλγόριθμος σχεδιάζει ένα αντιμικροβιακό προφίλ με τη μέθοδο κυλιόμενου παραθύρου (sliding-window method). Οι περιοχές μήκους λιγότερο από 12 κατάλοιπα που βρίσκονται κάτω από το κατώφλι θεωρούνται υποθετικά αντιμικροβιακά domains. Η αξιοπιστία του αλγόριθμου επιβεβαιώθηκε in silico και φάνηκε να ταυτοποιεί σωστά 80-90% των αντιμικροβιακών πρωτεϊνών και να προβλέπει το domain με τις αντιμικροβιακές ιδιότητες. Σε συνδυασμό με το εργαλείο στοίχισης ακολουθιών T-coffee, στο οποίο ενσωματώθηκε η εφαρμογή, οι αντιμικροβιακές περιοχές δύνανται να ελεχθούν ώστε να διαπιστωθεί η ύπαρξη πιθανών συντηρημένων αντιμικροβιακών domains.

Η εφαρμογή είναι προσβάσιμη μέσω του συνδέσμου http://tcoffee.crg.cat/apps/ampa

- **Πρόβλεψη αντιμικροβιακών πεπτιδίων με βάση τη στοίχιση ακολουθιών και μεθόδους επιλογής χαρακτηριστικών**

Η μελέτη (P. Wang et al., 2011) περιλαμβάνει την υλοποίηση μιας μεθοδολογίας για την πρόβλεψη AMPs η οποία ενσωματώνει τη μέθοδο στοίχισης ακολουθιών (sequence alignment) και τη μέθοδο επιλογής χαρακτηριστικών (feature selection). Σε ένα νέο σύνολο δεδομένων το συνολικό jackknife ποσοστό επιτυχίας ήταν πάνω από 80.23% και η τιμή MCC ήταν 0.73, κάτι το οποίο υποδηλώνει μια καλή πρόβλεψη. Παράλληλα,

βρέθηκε οτι τα αποτελέσματα ήταν σύμφωνα με την πρότερη γνώση όπου μερικά αμινοξέα προτιμώνται σε AMPs και διαδραματίζουν σπουδαίο ρόλο στην αντιμικροβιακή δραστηριότητα.

Στη μέθοδο επιλογής χαρακτηριστικών, κάθε πεπτίδιο κωδικοποιήθηκε με 270 χαρακτηριστικά, στα οποία συμπεριλαμβάνονται η αμινοξική και η ψευδο-αμινοξική σύνθεση, που ενσωματώνει το ηλεκτροστατικό φορτίο, το μοριακό όγκο, την πολικότητα, τη δευτεροταγή δομή και την ποικιλία ως προς τα κωδικόνια. Η μέθοδος στοίχισης αλληλουχιών βασίστηκε στο BLASTP και η πρόβλεψη διεξήχθη αναθέτοντας το πεπτίδιο που αναζητάται στην κατηγορία των πεπτιδίων που έχει τη μεγαλύτερη ομοιότητα με το ζητούμενο. Στη συνέχεια, οι παραπάνω μέθοδοι, συμπεριλαμβανομένων των μεθόδων Maximum Relevance Minimum Redundancy (mRMR) (Peng, Long, & Ding, 2005) και Incremental Feature Selection (IFS) (Kohavi R, 1997), εφαρμόστηκαν στην επιλογή των βέλτιστων χαρακτηριστικών για την πρόβλεψη των AMPs έναντι των non-AMPs. Το μοντέλο πρόβλεψης αναπτύχθηκε με τη χρήση του αλγόριθμου Nearest Neighbor (NNA). Το σύνολο εκπαίδευσης αποτέλεσαν ακολουθίες που συλλέχθησαν από τη βάση δεδομένων CAMP. Στην ανάλυση των βέλτιστων χαρακτηριστικών παρατηρήθηκε ότι τα αντιμικροβιακά πεπτίδια είναι πλούσια σε κυστεϊνη, τρυπτοφάνη, αργινίνη και ιστιδίνη ενώ η προλίνη δεν εμφάνιζε εμφανή διαφορά μεταξύ AMPs και non-AMPs. Η τρυπτοφάνη είναι σημαντική για τη δέσμευση λιπιδίων και προνομιακή στη διεπαφή πρωτεΐνης - μεμβράνης.

2.4 Βάσεις δεδομένων αντιμικροβιακών πεπτιδίων

Ο αριθμός των πεπτιδίων που έχουν ταυτοποιηθεί και χαρακτηριστεί είναι μεγαλύτερος από 5000, αριθμός που αφορά τόσο φυσικά πεπτίδια που έχουν ανακαλυφθεί σε ευκαρυωτικούς και προκαρυωτικούς οργανισμούς όσο και πεπτίδια που έχουν συντεθεί με τεχνητούς τρόπους. Επόμενως η οργάνωσή τους σε βάσεις δεδομένων κρίνεται αναγκαία.

Υπάρχουν αρκετές δημόσιες βάσεις δεδομένων από τις οποίες μπορεί να αντληθεί επαρκής πληροφορία αναφορικά με τα χαρακτηρισμένα AMPs. Οι πιο σημαντικές εξ αυτών αναφέρονται αναλυτικά στη συνέχεια.

❖ Data Repository of AntiMicrobial Peptides (DRAMP)

Η **DRAMP** (Fan et al., 2016) βάση δεδομένων αντιμικροβιακών πεπτιδίων περιέχει 17349 αντιμικροβιακές ακολουθίες, στις οποίες συμπεριλαμβάνονται 4571 γενικά AMPs, 12704 ακολουθίες που προέρχονται από πατέντες και 74 πεπτίδια που έχουν αναπτυχθεί από εταιρείες ως υποψήφια φάρμακα σε προκλινικό ή κλινικό δοκιμαστικό στάδιο. Η συλλογή των αντιμικροβιακών πεπτιδίων πραγματοποιήθηκε από τη PubMed, τη Uniprot και τη Lens χρησιμοποιώντας τις αντιμικροβιακές δράσεις, «antimicrobial, antifungal, antiviral, anticancer, antitumor, antiviral, insecticidal,antiparasitic, antibacterial peptides» ως λέξεις κλειδιά. Από τις ακολουθίες που ανακτήθηκαν καταχωρήθηκαν στη βάση όσες είχαν διευκρινιστεί και είχαν μήκος μικρότερο από 100 αμινοξικά κατάλοιπα, είχαν πιστοποιημένη αντιμικροβιακή δραστηριότητα και οι σηματοδοτικές περιοχές είχαν απομακρυνθεί.

Όλες οι εγγραφές στη βάση περιέχουν αναλυτικό σχολιασμό, με συνδέσμους που παραπέμπουν απευθείας στην PubMed, τη Uniprot και τη Protein Data Bank (PDB). 253 από τα πεπτίδια της DRAMP έχουν γνωστή δομή στη PDB. Παράλληλα, σε κάθε εγγράφη υπάρχει λεπτομερής περιγραφή της αντιμικροβιακής δράσης και της δομικής πληροφορίας.

Η πρόσβαση στην παραπάνω βάση πραγματοποιείται μέσω του συνδέσμου http://dramp.cpu-bioinfor.org/ και παρέχει λειτουργίες εύκολης περιήγησης και στοχευμένης αναζήτησης πεπτίδιων. Τέλος, μέσω διαθέσιμων εργαλείων δίνει τη δυνατότητα στοίχισης ακολουθιών, αναζήτησης ομοιότητας ακολουθιών και συντηρημένων περιοχών καθώς και υπολογισμού φυσικοχημικών ιδιοτήτων.

❖ Antimicrobial Peptide Database (APD)

Η **APD** (Z. Wang & Wang, 2004) βάση δεδομένων δημιουργήθηκε βασιζόμενη σε βιβλιογραφική έρευνα και περιέχει πληροφορίες για 525 πεπτίδια εκ των οποίων 498 είναι αντιβακτηριακά, 155 αντιμυκητιακά, 28 αντιϊϊκά και 18 αντικαρκινικά. Πρόκειται για βάση χρήσιμη στη μελέτη της σχέσης μεταξύ δομής κ λειτουργίας των AMPs.

Στη συνέχεια, η APD ανανεώθηκε και επεκτάθηκε με αποτέλεσμα τη δημιουργίας της πιο ενημερωμένης πλέον **APD2** (G. Wang, Li, & Wang, 2009). Η APD2 περιλαμβάνει 1228

καταχωρήσεις με 65 αντικαρκινικά, 76 αντιϊκά, 944 αντιβακτηριακά και 327 αντιμυκητιακά AMPs. Στη νέα αυτή έκδοση ο χρήστης έχει τη δυνατότητα να αναζητήσει τον οργανισμό από τον οποίο προέρχονται τα πεπτίδια, τους στόχους στους οποίους προσδένονται (μεμβράνες, σάκχαρα, DNA/RNA, ή πρωτεΐνες) καθώς επίσης και τις οικογένειες στις οποίες ανήκουν (bacteriocins, cyclotides κοκ).

Σήμερα, η συγκεκριμένη βάση έχει επίσης ανανεωθεί με τελευταία τη δημιουργία της **APD3** (G. Wang, Li, & Wang, 2016), η οποία επικεντρώνεται σε φυσικά πεπτίδια, μήκους μικρότερου από 100 αμινοξικά κατάλοιπα, με καθορισμένη ακολουθία και βιοδραστηριότητα. Συνολικά, διαθέτει πληροφορίες για 2619 AMPs εκ των οποίων 2169 είναι αντιβακτηριακά, 959 αντιμυκητιακά, 185 αντικαρκινικά, 172 αντικϊκά, 80 αντιπαρασιτικά και 105 αντι – HIV. Παράλληλα, υπάρχει πιο εκτενής κατηγοριοποίηση και ως προς τον οργανισμό από τον οποίο προέρχονται, οπότε έχουν καταγραφεί 13 AMPs με προέλευση από μύκητες, 321 από φυτά, 4 από αρχαία, 7 από πρώτιστα, 261 είναι βακτηριοσίνες από βακτήρια και 1972 είναι πεπτίδια που άμυνας ξενιστών σε ζώα.

Η πρόσβαση τελικά σε αυτήν τη βάση παρέχεται μέσω του συνδέσμου http://aps.unmc.edu/AP/main.php

- **Collection of Anti-Microbial Peptides (CAMP)**

Η **CAMP** (Thomas, Karnik, Barai, Jayaraman, & Idicula-Thomas, 2010) είναι μια επίσης δημόσια βάση δεδομένων μεταγενέστερη της πρώτης έκδοσης της APD και αναπτύχθηκε προκειμένου να εξελίξει την υπάρχουσα κατανόηση της λειτουργίας των αντιμικροβιακών πεπτιδίων. Περιλαμβάνει 3782 ακολουθίες AMPs, οι οποίες είτε έχουν προσδιοριστεί με πειραματικές διαδικασίες, είτε έχουν προβλεφθεί σύμφωνα με τις βιβλιογραφικές αναφορές.

Το σύνολο των δεδομένων που έχει επαληθευθεί πειραματικά χρησιμοποιήθηκε αργότερα και στην εκπαίδευση αλγορίθμων μάθησης, πχ Support Vector Machines (SVM), Random Forests (RF), για την ανάπτυξη εργαλείων πρόβλεψης. Η πρόσβαση στη βάση CAMP είναι ελεύθερη μέσω του συνδέσμου http://www.camp.bicnirrh.res.in/

- **Dragon Antimicrobial Peptide Database (DAMPD)**

Η **DAMPD** (Seshadri Sundararajan et al., 2012b) είναι μία βάση δεδομένων που προέκυψε ως ενημέρωση και αντικατάσταση της **ANTIMIC** που προϋπήρχε (Brahmachary et al., 2004)**.** Η DAMPD είναι εμπλουτισμένη με εργαλεία που μπορούν να βελτιώσουν τις μελέτες των αντιμικροβιακών πεπτιδίων. Με τη χρήση Support Vector Machines αναπτύχθηκε εργαλείο πρόβλεψης ώστε να επιτυγχάνεται όσο το δυνατόν πιο ακριβής κατάταξη ενός πεπτιδίου σε γνωστές αντιμικροβιακές οικογένειες. Το πρωτοποριακό χαρακτηριστικό της DAMPD είναι ότι συνδυάζει τον αλγόριθμο αυτό με σημαντικά εργαλεία όπως το BLAST, το ClustalW, το Hydrocalculator, το SignalP, το HMMER. Σκοπός της DAMPD, που περιλαμβάνει 1232 AMPs, είναι να ενημερώνεται σε μηνιαία βάση. Η πρόσβαση στη βάση αυτή παρέχεται μέσω του συνδέσμου http://apps.sanbi.ac.za/dampd/.

- **PhytAMP**

Η **PhytAMP** είναι μία βάση δεδομένων αποκλειστικά αφιερωμένη σε αντιμικροβιακά πεπτίδια που προέρχονται από τα φυτά και περιλαμβάνει 271 AMPs (Hammami, Ben Hamida, Vergoten, & Fliss, 2009). Τα φυτά διαθέτουν έναν έμφυτο μηχανισμό άμυνας ενάντια στα παθογόνα χάρη σε μικρά, πλούσια σε κυστεϊνες, πεπτίδια που παράγουν. Η βάση αυτή περιέχει ταξονομικά, μικροβιολογικά και φυσικοχημικά δεδομένα σχετικά με αυτά τα πεπτίδια. Στόχος είναι να δημιουργηθεί μια συλλογή ώστε να μελετηθεί ο τρόπος με τον οποίο η παρατηρούμενη δομή τους δύναται να χρησιμοποιηθεί στη γενετική μηχανική για την αύξηση της αντίστασης των αντιβιοτικών ή των φυτών έναντι σε παθογόνους μικροοργανισμούς. Η πρόσβαση στη βάση είναι ελεύθερη μέσω του συνδέσμου http://phytamp.hammamilab.org/main.php.

- **BACTIBASE**

Πλέον οι διαθέσιμες βάσεις δεδομένων AMPs είναι πολυάριθμες και ανάμεσα τους βρίσκεται και η **BACTIBASE** (Hammami, Zouhir, Ben Hamida, & Fliss, 2007) η οποία είναι αποκλειστικά αφιερωμένη σε βακτηριοσίνες. Περιέχει γενικές πληροφορίες που

αφορούν τη οικογένεια bacteriocins, φυσικοχημικά και στατιστικά δεδομένα, BLAST διεπαφή καθώς και διεπαφή ανάλυσης ακολουθιών σύμφωνα με τις αναζητήσεις του χρήστη.

Η πρόσβαση στη βάση είναι ελεύθερη μέσω του συνδέσμου http://phytamp.hammamilab.org/main.php.

ΚΕΦΑΛΑΙΟ 3

3.1 Μεθοδολογία ανάπτυξης μοντέλου ταξινόμησης

Η υλοποίηση της μεθοδολογίας για το χαρακτηρισμό των πεπτιδίων με βάση την δράση τους ως αντικαρκινικά, αντιβακτηριακά, αντιμυκητιακά, αντιμικροβιακά κοκ αρχικά απαιτεί τη συλλογή, από υπάρχουσες βάσεις δεδομένων, ενός επαρκούς συνόλου εκπαίδευσης με ήδη χαρακτηρισμένα πεπτίδια που ανήκουν σε διάφορες κατηγορίες βιοδραστηριότητας. Για όλα αυτά τα πεπτίδια υπολογίστηκε ένα σύνολο ακολουθιακών και φυσικοχημικών ιδιοτήτων και στη συνέχεια προκειμένου να αυξηθεί τεχνικά ο αριθμός των δειγμάτων χρησιμοποιήθηκε ένας συνδυασμός των τεχνικών boosting και τυχαίας υπερδειγματοληψίας. Τέλος, εφαρμόστηκε η μεθοδολογία ταξινόμησης EnsembleGASVR, για την εκπαίδευση των μοντέλων ταξινόμησης με βάση τη βιοδραστηριότητά τους. Η μεθοδολογία αυτή είχε χρησιμοποιηθεί με επιτυχία στο παρελθόν στο πρόβλημα εντοπισμού παθογενών μονονουκλεοτιδικών πολυμορφισμών και έχει το πλεονέκτημα ότι χειρίζεται με αποδοτικό τρόπο τις ελλιπείς τιμές ένω, παράλληλα, τα μοντέλα που εξάγονται από αυτήν παρέχουν και ένα βαθμό εμπιστοσύνης για κάθε πρόβλεψή τους.

3.2 Αλγόριθμος EnsembleGASVR

Η **EnsembleGASVR** (Rapakoulia et al., 2014) είναι μια καινοτόμος μεθοδολογία Μηχανικής Εκμάθησης η οποία εισήχθη για την ταξινόμηση μη συνώνυμων σημειακών νουκλεοτιδικών πολυμορφισμών (nsSNPs) σε ουδέτερους ή παθογενείς μελετώντας ένα σύνολο χαρακτηριστικών που αποτελούν τα πιο ενδεικτικά που έχουν προταθεί στη βιβλιογραφία. Πρόκειται για μια τεχνική ταξινομήσης η οποία συνδυάζει έναν γενετικό αλγόριθμο Adaptive Genetic Algorithm (GA) με έναν αλγόριθμο nu-Support Vector Regression (nu-SVR), μέσω ενός συνόλου αλγοριθμικών πλαισίων. Η τεχνική EnsembleGASVR συνδυάζει οχτώ μεμονωμένα μοντέλα ταξινόμησης για να αντιμετωπίσει το πρόβλημα των ελλιπών τιμών στα σύνολα δεδομένων και να χειριστεί το γεγονός των μη ισοζυγισμένων, ανομοιόμορφων κατηγοριών που εντοπίζονται σε αυτά. Οι Ensemble τεχνικές αυξάνουν την αποτελεσματικότητα μεμονομένων

ταξινομητών. Η εκπαίδευσή τους και ο συνδυασμός των εξόδων τους σε μια μοναδική έξοδο μεγιστοποιεί την απόδοση και την ακρίβεια, σε σύγκριση με έναν μοναδικό ταξινομητή, δεδομένου οτι έχουν εκπαιδευτεί σε διαφορετικά υποσύνολα του αρχικού dataset. Η εκπαίδευση πολλαπλών μοντέλων και η χρήση μιας συνάρτησης που ειδικεύεται σε μη ισορροπημένα σύνολα δεδομένων βιοπληροφορικής, εκμεταλλεύεται την πλήρη κατανομή του συνόλου δεδομένων, ενω ταυτόχρονα, οι επιπτώσεις της ανισορροπίας μειώνονται. Τελικά, επιτυγχάνεται η απόκτηση μιας τεχνικής παλινδρόμησης αρκετά ισχυρής, που δύναται να χρησιμοποιηθεί στην ταξινόμηση αντιμικροβιακών πεπτιδίων και στην ανάθεση βαθμολογίας σε κάθε πρόβλεψη.

3.2.1 Support Vector Regression SVR

Ένα μοντέλο ταξινόμησης SVR διατηρεί τη βασική ιδέα του μοντέλου Support Vector Machine, SVM, μεταφέροντας, μη γραμμικά, τα δεδομένα σε ένα μεγαλύτερης διάστασης χώρο χαρακτηριστικών και εφαρμόζει γραμμική παλινδρόμηση σε αυτό το χώρο. Οι αρχές που ακολουθεί για την ταξινόμηση, δηλαδή, η ελαχιστοποίηση του λάθους, η εύρεση του βέλτιστου hyperplane που μεγιστοποιεί το margin και η αποδοχή ότι ένα ποσοστό λάθους είναι ανεκτό, είναι κοινές. Τα μοντέλα SVR βρίσκουν ευρεία εφαρμογή σε προβλήματα αναγνώρισης προτύπων καθώς εμφανίζουν υψηλή αποδόση και χαμηλή πολυπλοκότητα. Ωστόσο, η διαφορά τους με τα SVMs εντοπίζεται σε ένα περιθώριο ανοχής «epsilon» που ορίζεται κατά προσέγγιση προς το SVM, εξαιτίας της εξόδου που είναι ένας πραγματικός αριθμός και καθίσταται δύσκολη η πρόβλεψη της πληροφορίας.

Ανάλογα με τον τρόπο που χειρίζεται ο αλγόριθμος το περιθώριο ανοχής και την παράμετρο ποινής υπάρχουν δυο βασικές εκδοχές, το **epsilon-SVR** και το **nu-SVR**. Στο epsilon-SVR, δεν ελέγχεται το πλήθος των διανυσμάτων δεδομένων (data vectors), από το σύνολο, που μετατρέπονται σε διανύσματα υποστήριξης (support vectors). Σε αντίθεση, ο nu-SVR αλγόριθμος έχει μια συστηματοποιημένη παράμετρο C για τον έλεγχο του ποσοστού λάθους στο μοντέλο. Επιπλέον, αποκτά μια πιο ουσιαστική ερμηνεία εισάγοντας μια παράμετρο **«nu»,** η οποία επιτρέπει τον έλεγχο του αριθμού των support vectors. Αυτό συμβαίνει γιατί καθορίζει το ποσοστό των support vectors που διατηρούνται στη λύση σε σχέση με το συνολικό αριθμό δειγμάτων. Η παράμετρος **nu** αντιπροσωπεύει ένα ανώτατο όριο στα δείγματα εκπαίδευσης που έχουν προβλεφθεί λάθος και ένα κατώτατο όριο στα δείγματα που είναι support vectors(Langhammer, Česák,

Langhammer, & Česák, 2016). Οι έξοδοι των nu-SVR είναι πραγματικές τιμές που αποτελούν scores για την υποστήριξη των προβλέψεων. Επιπλέον, με βάση τις απαιτήσεις, τα μοντέλα αυτά μπορούν να αλλάξουν το κατώφλι απόφασης και να ρυθμίσουν με αυτόν το τρόπο τα μέτρα της ευαισθησίας και της ειδικότητας.

3.2.2 Γενετικοί Αλγόριθμοι

Η αρχική ιδέα της υπολογιστικής εξέλιξης εισήχθη το 1960 από τον *I.Rechenberg* και αποτυπώθηκε στο στο έργο του, "*Evolution strategies*". Η ιδέα αυτή αναπτύχθηκε περαιτέρω από μεταγενέστερους ερευνητές. Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms, GAs) είναι στοχαστικοί αλγόριθμοι βελτιστοποίησης, που βασίζονται στην εξελικτική θεωρία του Δαρβίνου και το γεγονός οτι κάθε είδος στη φύση, προκειμένου να μεγιστοποιήσει την πιθανότητα επιβίωσής του, οφείλει να προσαρμόζεται σε ένα περιβάλλον ευμετάβλητο και πολύπλοκο. Κάθε κύτταρο στους ζωντανούς οργανισμούς περιέχει χρωμοσώματα πάνω στα οποία βρίσκεται το σύνολο των γονιδίων που φέρουν κωδικοποιημένη όλη τη γενετική του πληροφορία. Κατά τη διάρκεια της αναπαραγωγής, τα χρωμοσώματα υφίστανται μετασχηματισμούς καθώς τα γονίδια από τους προγόνους σχηματίζουν το νέο χρωμόσωμα των απογόνων, το οποίο όμως στη συνέχεια μπορεί και να μεταλλαχθεί. Οι μεταλλάξεις προέρχονται από λάθη κατά τη διαδικασία της αντιγραφής των γονιδίων από τους γονείς. Ακόμα και αν μερικές αλλαγές δεν είναι ευεργετικές, με το πέρασμα των χρόνων τείνον να ευνοούν την ανάπτυξη ειδών που είναι πιο πιθανό να επιβιώσουν και να μεταφέρουν βελτιωμένα χαρακτηριστικά σε επόμενες γενιές.

Οι Γενετικοί Αλγόριθμοι είναι εφεύρεση του John Holland , στην προσπάθεια του να μιμηθεί μερικές από τις διεργασίες της φυσικής εξέλιξης και επιλογής, και δημοσιεύθηκαν στο βιβλίο του "*Adaptation in Natural and Artificial Systems*" το 1975. Συγκεκριμένα, προσπάθησε να προσομοιώσει το γενετικό αλγόριθμο της φύσης με τον εξής τρόπο:

1. Μια λύση του προβλήματος αναπαρίσταται ως μια συμβολοσειρά από γονίδια που μπορεί να λάβει μια συγκεκριμένη τιμή σε ενα πεπερασμένο δάστημα αριθμών ή αλφάβητο. Αυτή η συμβολοσειρά, που αντιπροσωπεύει μια λύση, ονομάζεται χρωμόσωμα (**chromosome**).
2. Ένας αρχικός πληθυσμός (**population**) από χρωμοσώματα συντίθεται τυχαία και αντιπροσωπεύει ένα σύνολο λύσεων.

3. Σε κάθε αναπαραγωγή, υπολογίζεται η καταλληλότητα (**fitness**) κάθε χρωμοσώματος στον πληθυσμό.
4. Οι απόγονοι (**offsprings**) της επόμενης αναπαραγωγής προέρχονται από την επιλογή των πιο κατάλληλων χρωμοσωμάτων των προγόνων, κληρονομώντας τα καλύτερα χαρακτηριστικά από τους δυο γονείς (parents). Με μια πιθανότητα διασταύρωσης, οι γονείς διασταυρώνονται ώστε να σχηματίσουν τον απόγονο, ενώ αν δεν υπάρξει διασταύρωση, ο απόγονος είναι ακριβές αντίγραφο των γονέων. Επίσης, με μια πιθανότητα μετάλλαξης, μεταλλάσεται κάποια θέση του χρωμοσώματος του απόγονου και τελικά προστίθεται στο νέο πληθυσμό. Η επιλογή των καταλληλότερων χρωμοσωμάτων συνεχίζεται για πολλές αναπαραγωγές με την ελπίδα ότι το αποτέλεσμα θα είναι ένας πληθυσμός που ουσιαστικά θα εφαρμόζει καλύτερα από τον αρχικό. Οι επαναλήψεις σταματούν όταν ικανοποιηθεί το κριτήριο τερματισμού που τίθεται εξαρχής στο εκάστοτε πρόβλημα, όπως, για παράδειγμα, όταν επιτευχθεί ο μέγιστος αριθμός αναπαραγωγών.

Οι βασικές συνιστώσες των γενετικών αλγόριθμων συνοψίζονται στη συνέχεια.

- Χρωμοσωμική αναπαράσταση

Ο τρόπος αναπαράστασης κάθε λύσης ως ένα χρωμόσωμα ποικίλει. Ανάλογα με την εφαρμογή, ένα γονίδιο μπορεί να αντιπροσωπευθεί είτε από το δυαδικό αλφάβητο {0,1} είτε ακόμα και από ακέραιους ή πραγματικούς αριθμούς.

Δυαδική κωδικοποίηση

Χρωμόσωμα Α	**11100101001011010010**
Χρωμόσωμα Β	**00110011010100111010**

Αριθμητική κωδικοποίηση

Χρωμόσωμα Α	**1.94 611.71 254.08 0.024**
Χρωμόσωμα Β	**1.11 261.78 134.58 0.38**

Η αριθμητική κωδικοποίηση χρησιμοποιείται σε πολύπλοκα προβλήματα πραγματικών αριθμών όπως είναι η εύρεση των βαρών σε ένα τεχνητό νευρωνικό δίκτυο.

- Αρχικός πληθυσμός

Μετά από την επιλογή της χρωμοσωμικής κωδικοποίησης δημιουργείται τυχαία ένας πληθυσμός ως το σημείο εκκίνησης του αλγόριθμου. Συνήθως στα περισσότερα optimization προβλήματα επιλέγεται ένας πληθυσμός μεταξύ 30 και 100 ο οποίος δείχνει τον αριθμό των χρωμοσωμάτων στον πληθυσμό, σε μια αναπαραγωγή.

- Αξιολόγηση καταλληλότητας

Ένας στόχος ή μια συνάρτηση καταλληλότητας δοκιμάζεται έναντι κάθε χρωμοσώματος στο υπό εξέταση περιβάλλον και αναμένεται ο ατομικός βαθμός καταλληλότητας κάθε χρωσοσώματος να αυξάνεται όσο προχωράει ο αλγόριθμος καθώς επίσης και η συνολική καταλληλότητα του πληθυσμού συνολικά.

- Επιλογή

Ανάλογα με το βαθμό καταλληλότητάς τους κάποια γονεϊκά χρωμοσώματα του τρέχοντος πληθυσμού επιλέγονται για την αναπαραγωγή χρωμοσωμάτων του νέου πληθυσμού. Όσο καλύτερο είναι ένα χρωμόσωμα, τόσες περισσότερες πιθανότητες έχει να επιλεχθεί. Μια προτινόμενη μέθοδος επιλογής είναι η Roulette Wheel Selection, η οποία, συνοπτικά, εκτελεί τα παρακάτω βήματα.

i. Υπολογίζει το fitness άθροισμα όλων των χρωμοσωμάτων στον πληθυσμό: S
ii. Επιλέγει ένα τυχαίο αριθμό στο διάστημα [0,S]: r
iii. Υπολογίζει το fitness άθροισμα στον πληθυσμό ξεκινώντας από το 0: s
iv. Επιστρέφει το χρωμόσωμα που βρίσκεται στη θέση όπου s > r.

- Χειριστής διασταύρωσης (crossover operator)

Έπειτα από την επιλογή ενός ζεύγους χρωμοσωμάτων για την επόμενη γενιά, εφαρμόζεται διασταύρωση ώστε να προκύψει ο απόγονος, με στόχο την βελτίωση των νέων χρωμοσωμάτων, διατηρώντας ένα καλό μέρος από τα προηγούμενα χρωμοσώματα. Ο χειριστής εφαρμόζεται με κάποια πιθανότητα, όπου πιθανότητα ίση με 1 σημαίνει οτι ο απόγονος προκύπτει εξ ολοκλήρου από διασταύρωση, ενώ πιθανότητα ίση με 0 υποδηλώνει οτι η επόμενη γενιά είναι ακριβές αντίγραφο των χρωμοσωμάτων

του προηγούμενου πληθυσμού. Ωστόσο, σύμφωνα με εμπειρικές μελέτες μια πιθανότητα μεταξύ 0.65 και 0.85 επιτυγχάνει καλύτερα αποτελέσματα.

- Χειριστής μετάλλαξης (mutation operator)

Αντίστοιχα με τον χειριστή διασταύρωσης, εφαρμόζεται κ ο χειριστής μετάλλαξης με κάποια πιθανότητα στη συνέχεια της διασταύωσης. Αν δεν υπάρξει μετάλλαξη, δηλαδή με πιθανότητα 0% ο απόγονος προκύπτει από τη διασταύρωση χωρίς αλλαγές, ενώ στην αντίθετη περίπτωση ένα μέρος του χρωμοσώματος του απογόνου αλλάζει, όπου πιθανότητα ίση με 100% υποδηλώνει αλλαγή ολόκληρου του χρωμοσώματος.

Υπάρχουν διάφοροι τρόποι διασταύρωσης και μετάλλαξης και εφαρμόζονται ανάλογα με τη χρωμοσωμική κωδικοποίηση.

Μέθοδος	Εφαρμογή
Single – point crossover	Επιλέγεται ένα σημείο διασταύρωσης. Το τμήμα του χρωμοσώματος από την αρχή ως το σημείο αυτό επιλέγεται από τον γονέα-Α για αντιγραφή, και το υπόλοιπο αντιγράφεται από το γονέα-Β. **1100111**001 + 0101010**101** = **1100111101**
Two – point crossover	Επιλέγονται δυο σημεία διασταύρωσης. Το τμήμα του χρωμοσώματος από την αρχή ως το πρώτο σημείο διασταύρωσης επιλέγεται από το γονέα-Α για αντιγραφή, από το σημείο αυτό ως το δεύτερο σημείο αντιγράφεται το τμήμα του γονέα-Β, και το υπόλοιπο αντιγράφεται και πάλι από το γονέα-Α. **1100**11100**1** + 0101**0101**11 = **1100010101**
Arithmetic – crossover	Εκτελείται κάποια αριθμητική πράξη πάνω στα γονεϊκά χρωμοσώματα για την παραγωγή του απογόνου. 1100111001 OR 1101010111 = **1101111111**

Mutation Binary	Κάποιο από τα bits στη συμβολοσειρά μπορεί να μετατραπεί από 0 σε 1 και αντίστροφα. **1101111111 - > 1101110101**
Mutation Value	Σε αριθμητική κωδικοποίηση, επιλέγεται ένας αριθμός για μετάλλαξη και σε αυτόν προστίθεται ή αφαιρείται κάποιος άλλος μικρός αριθμός. 6.73 8.24 55.44 124.12 78.21 6.73 8.24 55.44 **124.30 78.00**

Οι Γενετικοί Αλγόριθμοι εφαρμόζονται ευρέως σε υπολογιστικές μεθόδους τεχνητής νοημοσύνης χάρη στην αναγνωρισμένη τους ικανότητα να εξερευνούν μεγάλους χώρους αναζήτησης με στόχο τον εντοπισμό πιθανών λύσεων πολύ κοντά στη βέλτιστη λύση.

Στο EnsembleGASVR σύστημα υλοποιείται ένας γενετικός αλγόριθμος που ρυθμίζει τις nu-SVR παραμέτρους, τις οποίες αποτελούν α) το κατώφλι ταξινόμησης, β) οι παράμετροι ταξινόμησης C, nu και γ) το εύρος gamma της ακτινικής βάσης kernel.

Ένας νέος αλγόριθμος που χαρακτηρίζεται ως, **Evolutionary Multi-Objective Framework** ενσωματώθηκε στην τεχνική EnsembleGASVR**,** σε αντικατάσταση του γενετικού αλγόριθμου, με σκοπό την επίτευξη καλύτερης απόδοσης στην ταξινόμηση των πεπτιδίων ανάλογα με το αν εμφανίζουν κάποια βιοδραστηριότητα ή όχι.

3.3 Evolutionary Multi-Objective Framework

Ο **multi-objective optimization** αλγόριθμος, ο αλγόριθμος βελτιστοποίησης με πολλαπλά κριτήρια ή χαρακτηριστικά, αφορά στη διαδιακασία όπου δυο ή περισσότερα ζητήματα βελτιστοποιούνται ταυτόχρονα με διάφορους περιορισμούς. Η έλλειψη πρότερης γνώσης και το μεγάλο μέγεθος των χαρακτηριστικών που πρέπει να βελτιωθούν ευνόησε την επιλογή του (Corthésy et al., 2018). Ενας multi-objective evolutionary αλγόριθμος, όπως αποκαλείται διαφορετικά, στοχεύει στην επίλυση προβλημάτων στα οποία εμπλέκονται πολλαπλά κριτήρια που συγκρούονται μεταξύ τους και υπάρχει ένα σύνολο από Pareto βέλτιστες λύσεις, τις οποίες προσπαθεί να

προσεγγίσει. Ο όρος **Pareto set** ή **Pareto frontier** αναφέρεται στο σύνολο των βέλτιστων λύσεων όπου δεν μπορούν να βελτιωθούν ως προς ένα κριτήριο χωρίς να χειροτερέψουν σε τουλάχιστον ένα από τα υπόλοιπα κριτήρια. Οι λύσεις που συνθέτουν ενα Pareto front χαρακτηρίζονται ως **non-dominated,** δηλαδή καμια λύση δεν υπερέχει κάποιας άλλης.

Ο αλγόριθμος που προτάθηκε ξεκινάει αρχικοποιώντας ένα σύνολο από λύσεις για το υπο εξέταση πρόβλημα και στη συνέχεια, εφαρμόζει επαναληπτικά τους χειριστές των εξελικτικών αλγόριθμων, δηλαδή, της αξιολόγησης (evaluation operator), της επιλογής (selection operator), της διασταύρωσης (crossover operator) και της μετάλλαξης (mutation operator). Οι επαναλήψεις σταματούν όταν ικανοποιηθεί ένα κριτήριο τερματισμού. Η μέθοδος που υλοποιήθηκε είναι Pareto-based και χρησιμοποιώντας μια προσέγγιση ψευδο-τυχαίας αρχικοποίησης επιτρέπει τη γρήγορη σύγκλιση σε καλές λύσεις. Με τη στρατηγική αυτή αρχικοποιούνται οι λύσεις της πρώτης αναπαραγωγής. Ο αριθμός των αρχικών λύσεων προκαθορίζεται από το χρήστη μέσω μιας παραμέτρου. Οι επόμενες δυο μέθοδοι χρησιμοποιούνται με πιθανότητες τις οποίες ο χρήστης μπορεί να τροποποιήσει.

i. Οι μεταβλητές βελτιστοποίησης αρχικοποιούνται στις προεπιλεγμένες (default) τιμές. Οι τιμές αυτές αλλάζουν σύμφωνα με έναν τοπικό χειρισμό ο οποίος προσθέτει ένα τυχαίο νούμερο το οποίο λαμβάνεται από την κατανομή Gaussian μηδενικού μέσου όρου και διακύμανσης ίσης με το 10% του μεταβλητού διαστήματος των επιτρεπόμενων τιμών. Αυτός ο ψευδοτυχαίος μηχανισμός αρχικοποίησης επιτρέπει την επιλογή ενός αρχικού συνόλου καλών λύσεων και επιταχύνει την συνολική επίδοση της μεθόδου.

ii. Οι μεταβλητές βελτιστοποίησης αρχικοποιούνται τυχαία με τιμές από την κανονική κατανομή με μέσο όρο που ορίζεται ως :

$$mean = minValue + \frac{maxValue - minValue}{2}$$

και διακύμανση που ορίζεται ως :

$$variance = \frac{maxValue - minValue}{2}$$

όπου maxValue είναι η μέγιστη επιτρεπόμενη και minValue η ελάχιστη επιτρεπόμενη τιμή μιας μεταβλητής βελτιστοποίησης.

Στην επόμενη αναπαραγωγή, ένας multi-objective μηχανισμός χρησιμοποιείται για την επιλογή νέων λύσεων. Στο πρώτο βήμα υπολογίζεται ο αριθμός των Pareto frontiers.

Μία αρχική τιμή, fitness value, ανατίθεται έπειτα σε κάθε λύση, ίση με την αντίστροφη σειρά του parent-front στο οποίο είχε ανατεθεί κατά το προηγούμενο βήμα. Οι λύσεις ομαδοποιούνται με βάση την ομοιότητά τους και σχηματίζουν κόμβους λύσεων. Κατόπιν, οι τιμές μεταβάλλονται ανάλογα με τον αριθμό των λύσεων που ανήκουν στον κόμβο τους. Οι τιμές σε κάθε κόμβο διαιρούνται από την τιμή m (τη μέση ομοιότητα της κάθε λύσης), η οποία προκύπτει εφαρμόζοντας συγκρίσεις ανα ζεύγη σε όλες τις λύσεις σε έναν κόμβο, υπολογίζοντας τις γεωμετρικές τους αποστάσεις και τη μέση απόσταση ανα ζεύγη. Με αυτόν τον τρόπο αλγόριθμος ψάχνει πιο ενδελεχώς μη εξερευνημένες περιοχές στο χώρο αναζήτησης.

Για την επιλογή του πληθυσμού της επόμενης παραγωγής χρησιμοποιείται Roulette Wheel Selection, όπου σε κάθε πληθυσμό αποδίδεται η πιθανότητα να επιλεχθεί στην επόμενη παραγωγή αναλογικά με τις fitness τιμές του. Η καλύτερη λύση, αυτή με τη μέγιστη συνολική fitness τιμή, περνάει στην επόμενη αναπαραγωγή. Η τελική τιμή fitness υπολογίζεται από το ζυγισμένο μέσο των βελτιστοποιημένων στόχων με τα βάρη που έχει προκαθορίσει ο χρήστης. Όλοι οι στόχοι είναι το ίδιο σημαντικοί, οπότε όλα τα βάρη είναι προκαθορισμένα να ισούνται με 1.

Αναφορικά με τους χειριστές crossover, δυο crossover μέθοδοι χρησιμοποιούνται με πιθανότητες που επίσης παρέχει ο χρήστης. Οι προκαθορισμένες πιθανότητες είναι

- 45% για διασταύρωση 2-σημείων (two-point crossover operator). Ο χειριστής εδώ παράγει νέες λύσεις ανταλλάσοντας δυο μέρη από τις γονεϊκές λύσεις.
- 45% για την εφαρμογή αριθμητικής διασταύρωσης (arithmetic crossover). Ο χειριστής αυτός παράγει λύσεις συνδυάζοντας δυο λύσεις με βάση τις ακόλουθες ισότητες, όπου α είναι μια τυχαία μεταβλητή στο διάστημα [0,1]
    - $Offspring\ 1 = a * Parent1 + (1 - a) * Parent2$
    - $Offspring\ 2 = (1 - a) * Parent1 + a * Parent2$
- 10% για την μη εφαρμογή χειριστή διασταύρωσης.

Για τη μετάλλαξη, εφαρμόζεται ο Gaussian mutation operator, ως ο πιο κατάλληλος, για το float σχήμα αναπαράστασης που υιοθετείται από τον προτινόμενο αλγόριθμο. Η πιθανότητα μετάλλαξης είναι μια μεταβλητή που ορίζεται από τον χρήστη και αντιπροσωπεύει την πιθανότητα να επιλεχθούν τιμές στη λύση ώστε να μεταλλαχθούν.

Οι μεταβλητές αυτές που επιλέγονται μετατρέπονται με την προσθήκη μιας τυχαίας τιμής από την Gaussian κατανομή με μέσο ίσο του 0 και διακύμανση ίση με την:

$$variation = GausVar * (maxAllowed - minAllowed)$$

Όπου *maxAllowed* είναι η μέγιστη επιτρεπόμενη και *minAllowed* είναι η ελάχιστη επιτρεπόμενη τιμή της μεταβλητής.

Τα κριτήρα τερματισμού που πρέπει να ικανοποιηθούν είναι τα εξής:

- Κριτήριο σύγκλισης : Τερματισμός αν θεωρηθεί οτι ο πληθυσμός συγκλίνει, δηλαδή ότι η ομοιότητα μεταξύ των λύσεων ξεπερνά ένα όριο.
- Κριτήριο μέγιστου αριθμού αναπαραγωγής : Τερματισμός αν οι επαναλήψεις φθάσουν ένα μέγιστο αριθμό.

Με τον τερματισμό του αλγόριθμου, επιστρέφεται στο χρήστη η καλύτερη λύση αναφορικά με τη συνολική fitness λειτουργία μαζι με ένα σύνολο από λύσεις που συνιστούν τις Pareto βέλτιστες λύσεις, οι οποίες βρέθηκαν κατα την τελευταία αναπαραγωγή του αλγόριθμου.

ΚΕΦΑΛΑΙΟ 4

4.1 Πειραματικά αποτελέσματα

4.1.1 Συλλογή δεδομένων

Η συλλογή των δεδομένων που αποτέλεσαν το training dataset για το μοντέλο εκπαίδευσης πραγματοποιήθηκε από τις βάσεις δεδομένων DAMPD και DRAMP. Συγκεκριμένα, από τη DAMPD ανακτήθηκαν 783 πεπτίδια για την κατηγορία antimicrobial, ενώ από τη DRAMP συλλέχθησαν 75 antibacterial, 62 antifungal, 30 anticancer, 93 insecticidal και 136 antiviral. Ο λόγος που επιλέχθηκε η DRAMP μεταξύ των διαθέσιμων βάσεων δεδομένων είναι το γεγονός ότι υπήρχαν καταχωρημένα περισσότερα αντιπροσωπευτικά πεπτίδια από κάθε κατηγορία και κάθε ένα απο αυτά τα πεπτίδια συνοδεύεται από αναλυτική πληροφορία και συνδέσμους σε άλλες βάσεις δεδομένων, όπως η Uniprot και η PubMed. Επίσης, η DRAMP επιτρέπει τη μαζική επιλογή και ταυτόχρονη ανάκτηση των ζητούμενων ακολουθιών σε ποικίλες μορφές, fasta, HTML, XLS και XML. Από τα πεπτίδια που προέκυψαν μέσω της αναζήτησης στις παραπάνω κατηγορίες, τελικά, επιλέχθησαν μόνο όσα είχαν εγγραφή και τον απαραίτητο σχολιασμό «reviewed» στη Uniprot.

Η συλλογή των μη αντιμικροβιακών πεπτιδίων πραγματοποιήθηκε από τη Uniprot και αποτέλεσαν την κατηγορία non-AMPs. Η διαδικασία επιλογής ήταν η ακόλουθη (Παράρτημα ΙΙ):

1. Το μήκος της ακολουθίας ζητήθηκε να είναι μεταξύ 10 και 100 αμινοξικών καταλοίπων.
2. Από τις ακολουθίες αποκλείστηκαν όσες περιείχαν οποιαδήποτε από τις κάτωθι λέξεις-κλειδιά:
    - antibiotic
    - antimicrobial
    - antiviral defense
    - interferon antiviral system
    - antiviral protein

- tumor antigen
- insecticide resistance
- pharmaceutical use
- fungicide
- defensin
- plant defense
- amphibian defense peptide

3. Οι ακολουθίες επίσης ζητήθηκε να έχουν το σχολιασμό «reviewed».

Τελικά, συγκεντρώθηκαν 9463 non-AMPs πεπτίδια τα οποία, μέσω της Pfam, επιβεβαιώθηκε ότι δεν ανήκουν σε κάποια αντιμικροβιακή οικογένεια με χρήση του εργαλείου HMMER και κατάλληλου κώδικα που αναπτύχθηκε σε γλώσσα προγραμματισμού Python (Παράρτημα Ι: selectNegative.py)

Ωστόσο, λόγω της ύπαρξης των X, B, U, Z σε θέσεις αμινοξικών καταλοίπων σε ορισμένες πεπτιδικές ακολουθίες έγινε η απαραίτητη αντικατάσταση τους με κάθε ένα από τα 20 αμινοξέα με τη χρήση του κατάλληλου κώδικα που αναπτύχθηκε σε Python (Παράρτημα Ι: replace.py, replaceXBUZ.py), ενώ μερικές ακολουθίες από τα non-AMPs αφαιρέθηκαν λόγω του μεγάλου συνόλου των δεδομένων.

Τελικά το σύνολο δεδομένων για όλες τις κατηγορίες διαμορφώθηκε ως εξής:

Κατηγορία Βιοδραστηριότητας	**Σύνολο πεπτιδίων**
Antimicrobial	1408
Antibacterial	75
Antifungal	62
Anticancer	49
Insecticidal	131
Antiviral	213
Non-AMPs	8817

Στη συνέχεια, παρατηρήθηκε ότι τα δεδομένα σε κάθε κατηγορία παρουσίαζαν μεγάλες αποκλίσεις ως προς τον αριθμό τους, με συνέπεια την αδυναμία σωστής εκπαίδευσης του αλγόριθμου και τη διεξαγωγή ενός αξιόπιστου μοντέλου πρόβλεψης. Οπότε, έγινε μια ανακατανομή με βάση το μέσο όρο, φροντίζοντας η κάθε κλάση να έχει περίπου 200 ακολουθίες, και συγκεκριμένα 213, όσος δηλαδή είναι και ο αριθμός των antiviral πεπτιδίων. Πρόκειται για την τεχνική υπερδειγματοληψίας και υποδειγματοληψίας των κατηγοριών με τα λιγότερα (minority class) και τα περισσότερα (majority class) πεπτίδια, αντίστοιχα. Στις κλάσεις anticancer, antibacterial, antifungal, insecticidal επαναλήφθησαν κάποιες πεπτιδικές αλληλουχίες με τυχαίο τρόπο, ενώ αντίθετα στις κλάσεις antimicrobial και non-AMPs αφαιρέθηκαν αλληλουχίες.

Οι ακολουθίες που αφαιρέθηκαν από τις παραπάνω κατηγορίες αποτέλεσαν δεδομένα του **test dataset** για την επαλήθευση του αλγόριθμου σε άγνωστα δεδομένα και την αξιολόγηση της απόδοσής του. Παράλληλα, από τη βάση δεδομένων DRAMP επιλέχθησαν εκ νέου πεπτιδικές ακολουθίες που δεν είχαν συμπεριληφθεί στο αρχικό training dataset. Απαραίτητη προϋπόθεση ήταν να υπάρχει αντίστοιχη εγγραφή στη Uniprot, με τη διαφορά ότι ο χαρακτηρισμός «reviewed» δεν ήταν αναγκαίος, όπως συνέβη αντίθετα στο training dataset.

Τελικά το test dataset περιέχει:

Κατηγορία Βιοδραστηριότητας	**Σύνολο πεπτιδίων**
Antimicrobial	1183
Antibacterial	199
Antifungal	159
Anticancer	7
Antiviral	13
Non-AMPs	8601

4.2 Υπολογισμός χαρακτηριστικών πεπτιδίων

Η εκπαίδευση του αλγόριθμου για την πρόβλεψη της βιοδραστηριότητας των πεπτιδίων πραγματοποιήθηκε με βάση τη συλλογή και τη μελέτη των πιο σχετικών χαρακτηριστικών των αντιμικροβιακών πεπτιδίων. Για τον υπολογισμό των χαρακτηριστικών αυτών αναπτύχθηκαν κώδικες προγραμματισμού με χρήση της γλώσσας Python 2.7. Στα προγράμματα αυτά, scripts , η υλοποίηση των οποίων παρατίθεται στο Παράρτημα Ι, καλούνται τα απαραίτητα εργαλεία προκειμένουν να κατασκευαστεί το υπο μελέτη αρχείο με όλα τα υπολογισμένα χαρακτηριστικά για κάθε πεπτίδιο. Τα εργαλεία αυτά είναι τα εξης:

1. SignalP 4.1

   Το εργαλείο αυτό προβλέπει τις σηματοδοτικές θέσεις διάσπασης ενός πεπτιδίου.

2. ProtParam Biopython library

   Η βιβλιοθήκη αυτή παρέχει εργαλεία για την υπολογιστική Μοριακή Βιολογία στη γλώσσα προγραμματισμού Python και ανήκει στο Expasy Proteomics Server. Συγκεκριμένα με τη βοήθεια της βιβλιοθήκης αυτής υπολογίστηκαν τα ακόλουθα χαρακτηριστικά στην πεπτιδική αλληλουχία, η σημασία των οποίων αναλύεται στη συνέχεια:

   α. Οι συχνότητες αμινοξικών καταλοίπων

   β. Το μοριακό βάρος

   γ. Η αρωματικότητα

   δ. Η ελαστικότητα

   ε. Το ισοηλεκτρικό σημείο

   στ. Ο δείκτης αστάθειας

   ζ. Το κλάσμα της α-έλικας

   η. Το κλάσμα της στροφής

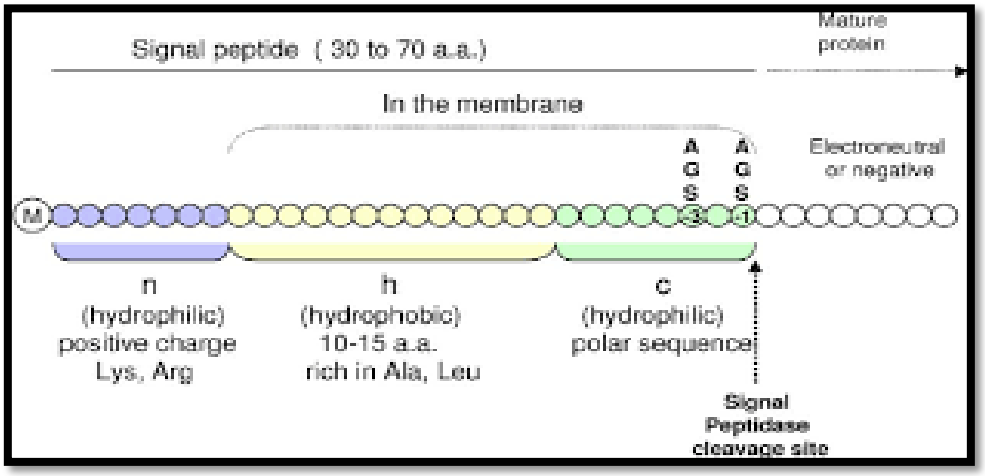
   θ. Το κλάσμα του β-φύλλου

   ι. Η υδροφοβικότητα.

4.2.1 Σηματοδοτικά πεπτίδια (Signal Peptides)

Τα **σηματοδοτικά πεπτίδια** (signal peptides) κατευθύνουν τις πρωτεΐνες στην σωστή τους ενδοκυτταρική και εξωκυτταρική τοποθεσία. Μια από τις διαδικασίες που επιτελούν είναι να ελέγχουν την είσοδο των πρωτεϊνών στο εκκριτικό μονοπάτι, τόσο σε ευκαρυωτικούς όσο και σε προκαρυωτικούς οργανισμούς. Συνήθως αποτελούν το αμινοτελικό N-terminal άκρο της αμινοξικής ακολουθίας το οποίο και αποκόβεται καθώς η πρωτεΐνη διαπερνά τη μεμβράνη από τη σηματοδοτική πεπτιδάση (signal peptidase). Η συνήθης δομή ενός σηματοδοτικού πεπτιδίου περιλαμβάνει ένα θετικά φορτισμένο n-άκρο, ακολουθούμενο από μια υδρόφοβη h-περιοχή και τέλος μια ουδέτερη αλλά πολική c-περιοχή, στην οποία βρίσκεται και το σημείο διάσπασης (H. Nielsen, Engelbrecht, Brunak, & von Heijne, 1997).

Στους ευκαρυτικούς οργανισμούς στην h-περιοχή εμφανίζεται κυρίως το αμινοξύ Λευκίνη (Leu, L) και μερικές φορές μαζί με τα αμινοξέα Βαλίνη (Val, V), Αλανίνη (Ala, A), Φαινυλαλανίνη (Phe, F) και Ισολευκίνη (Ile, I). Αντίθετα, στους ευκαρυωτικούς οργανισμούς στην h-περιοχή συναντώνται σε περίπου ισο ποσοστό τα αμινοξέα Λευκίνη(Leu, L) και Αλανίνη(Ala, A). Σύμφωνα με την έρευνα των von Heijne&Abrahmsen(1989), τα σηματοδοτικά πεπτίδια στα θετικά κατά Gram βακτήρια θεωρούνται μεγαλύτερα σε σχέση με άλλους οργανισμούς και παρουσιάζουν πιο εκτεταμένες h-περιοχές.



Εικόνα 3.1

Τριμερής δομή N-terminal σηματοδοτικού πεπτιδίου πρωτεΐνης στοχευμένης στο ER.

Η ανάγκη εξεύρεσης όσο το δυνατόν περισσότερων αποτελεσματικών μέσων παραγωγής πρωτεϊνών σε περίπλοκα συστήματα οδήγησε στην ανάγκη ανάπτυξης μεθόδων αυτόματης ταυτοποίησης σηματοδοτικών πεπτιδίων καθώς και πρόβλεψης των σημείων διάσπασής τους. Μία από τις μεθόδους που εξυπηρετεί τους παραπάνω σκοπούς είναι και το SignalP AN (Henrik Nielsen, 2017). Η συγκεκριμένη μέθοδος βασίζεται σε αλγόριθμους μηχανικής μάθησης, όπως είναι τα τεχνητά νευρωνικά δίκτυα και τα Hidden Markov Models. Υπάρχει συνδυασμός νευρωνικών δικτύων με το ένα δίκτυο να έχει εκπαιδευτεί ώστε να αναγνωρίζει το σημείο διάσπασης και ένα άλλο δίκτυο να κατηγοριοποιεί πεπτίδια ως σηματοδοτικά και μη-σηματοδοτικά. Η απόδοση δικτύων υπολογίστηκε με τη μέθοδο του cross-validation, χωρίζοντας το dataset σε 5 περίπου ίσα τμήματα με το ένα κομμάτι να αποτελεί κάθε φορά το test dataset και τα υπόλοιπα τέσσερα το training dataset. Τελικά, η συνολική απόδοση προέκυψε από το μέσο των πέντε διαφορετικών διαμερισμών.

Τα δίκτυα εκπαίδευσης παρέχουν τις εξης διαφορετικές βαθμολογίες για κάθε θέση στην ακολουθία:

S-score	Βαθμολογία που δίνει η έξοδος του δικτύου που διαχωρίζει σηματοδοτικά και μη σηματοδοτικά πεπτίδια, εκτιμώντας την πιθανότητα μία θέση να ανήκει σε ένα σηματοδοτικό πεπτίδιο.
C-score	Βαθμολογία που δίνει η έξοδος του δικτύου που αναγνωρίζει τις θέσεις διάσπασης, εκτιμώντας την πιθανότητα μια θέση να αποτελεί την πρώτη σε μια ώριμη πρωτεΐνη.
Y-score	Βαθμολογία που προκύπτει από το γεωμετρικό μέσο του S-score με το C-score. Η τιμή αυτή υπολογίζεται για να καλύψει την περίπτωση όπου εντοπίζονται πολλά σημεία διάσπασης, ανώτερες κορυφές C-scores, με συγκρίσιμη δυναμική. Συγκεκριμένα, το αληθινό σημείο διάσπασης προβλέπεται παρατηρώντας την καμπύλη S-score ώστε να προσδιοριστεί ποια είναι η κορυφή εκείνη που συμπίπτει καλύτερα με τη μετάβαση από τη σηματοδοτική στη μη σηματοδοτική περιοχή.

Mean S	Το μέσο S-score του πιθανού σηματοδοτικού πεπτιδίου, από τη θέση 1 ως τη θέση ακριβώς πριν από τη θέση με το μέγιστο Y-score.
D-score	Ο ζυγισμένος μέσος του μέσου S με τα μέγιστα Y-scores. Αυτή τελικά είναι η βαθμολογία που χρησιμοποιείται για να διαχωρίσει τα σηματοδοτικά από τα μη σηματοδοτικά πεπτίδια.

Ιδιαίτερο ενδιαφέρον παρουσίαζουν οι πρωτεΐνες που συντίθενται στο εκκριτικό μονοπάτι (Lodish et al., 2000). Οι διαλυτές πρωτεΐνες που ανήκουν σε αυτή την κατηγορία μόλις συντεθούν στο ριβόσωμα αρχικά τοποθετούνται στο ενδοπλασματικό δίκτυο και στη συνέχεια οδηγούνται σε άλλα οργανίδια ή εκκρίνονται από το κύτταρο. Παρομοίως, οι ενσωματωμένες μεμβρανικές πρωτεΐνες της ίδιας κατηγορίας αρχικά εισέρχονται στη μεμβράνη του ενδοπλασματικού δικτύου κατά τη σύνθεσή τους και έχουν την δυνατότητα είτε να παραμείνουν εκει, είτε να ενσωματωθούν στην πλασματική μεμβράνη, ή τις μεμβράνες του ενδότερου λείου ενδοπλασματικού δικτύου, είτε στο σύμπλεγμα Golgi, τα λυσοσώματα ή τα ενδοσώματα.

Οι πρωτεΐνες που εισέρχονται στο εκκριτικό μονοπάτι περιέχουν μια σηματοδοτική ακολουθία ενδοπλασματικού δικτύου, η οποία είναι απαραίτητη ώστε να κατευθυνθούν τα υπεύθυνα, για την σύνθεση των συγκεκριμένων πρωτεϊνών, ριβοσώματα στο ακατέργαστο (rough) ενδοπλασματικό δίκτυο. Ένα σηματοδοτικό πεπτίδιο συνήθως αποτελείται από μια ακολουθία των 20 αμινοξικών καταλοίπων.

Κάθε πεπτιδική αλληλουχία από το σύνολο δεδομένων, θετικών και αρνητικών, εξετάστηκε ως προς την παρουσία θέσεων διάσπασης σηματοδοτικών πεπτιδίων με τη χρήση του εργαλείου SignalP. Για το σκοπό αυτό το SignalP εγκαταστάθηκε σε Linux server που χρησιμοποιήθηκε για την εκπόνηση της εργασίας και αναπτύχθηκε κατάλληλο πρόγραμμα σε γλώσσα προγραμματισμού Python για την κλήση των κατάλληλων συναρτήσεων του πακέτου SignalP και την εξαγωγή τον αποτελεσμάτων (Παράρτημα Ι: signalPcalc.py). Τα αποτελέσματα, κατα την εκτέλεση του προγράμματος για το συνόλο δεδομένων των αντιβακτηριακών πεπτιδίων, αποθηκεύονται σε ένα αρχείο και συνολικά παράγονται τρια αρχεία για κάθε μια από τις εφτά κατηγορίες πεπτιδίων. Τα τρια αυτά αρχεία αντιστοιχούν στην πρόβλεψη σε ευκαρυωτικούς, Gram-positive και Gram-negative οργανισμούς.

Τα περιεχόμενα των αρχείων είναι της ακόλουθης μορφής:

```
# SignalP-4.1 euk predictions
# name                    Cmax  pos  Ymax  pos  Smax  pos  Smean   D     ?
Dmaxcut    Networks-used
DRAMP00126|Plantaricin_6   0.110  27  0.113  11  0.123   1  0.113   0.113 N
0.450      SignalP-noTM
DRAMP00127|Plantaricin_7   0.110  26  0.107  26  0.118  12  0.102   0.104 N
0.450      SignalP-noTM
DRAMP00128|Plantaricin_8   0.111  19  0.114  15  0.134   1  0.115   0.114 N
0.450      SignalP-noTM
DRAMP00129|Plantaricin_9   0.108  17  0.138  13  0.242   5  0.181   0.161 N
0.450      SignalP-noTM
DRAMP00136|Enterocin_10    0.140  28  0.216  13  0.559   4  0.397   0.314 N
0.450      SignalP-noTM
DRAMP00171|Lactocyclicin_11  0.244  31  0.223  31  0.344  11  0.219   0.221
N  0.500      SignalP-TM
DRAMP00173|Leucocyclicin_12  0.268  31  0.237  31  0.370  38  0.209   0.226
N  0.500      SignalP-TM
DRAMP00177|Enterocin_13    0.110  43  0.116  15  0.156  12  0.129   0.123 N
0.450      SignalP-noTM
DRAMP00178|Enterocin_14    0.134  20  0.125  20  0.157   2  0.114   0.119 N
0.450      SignalP-noTM
DRAMP00191|Microcin_15     0.107   8  0.108  11  0.120   1  0.104   0.106 N
0.450      SignalP-noTM
```

Αποτελέσματα προβλέψεων SignalP σε ευκαρυωτικούς οργανισμούς

```
# SignalP-4.1 gram- predictions
# name                    Cmax  pos  Ymax  pos  Smax  pos  Smean   D     ?
Dmaxcut    Networks-used
DRAMP00068|Aureocin_3      0.109  21  0.129  11  0.196  17  0.170   0.148 N
0.570      SignalP-noTM
DRAMP00074|Enterocin_4     0.108  24  0.147  11  0.325   3  0.211   0.177 N
0.570      SignalP-noTM
DRAMP00089|Bacteriocin_5   0.153  24  0.294  24  0.712  19  0.456   0.370 N
0.570      SignalP-noTM
DRAMP00126|Plantaricin_6   0.101  22  0.128  11  0.190   4  0.163   0.145 N
0.570      SignalP-noTM
DRAMP00127|Plantaricin_7   0.102  31  0.127  15  0.219   4  0.161   0.143 N
0.570      SignalP-noTM
DRAMP00128|Plantaricin_8   0.100  24  0.105  20  0.137  12  0.099   0.102 N
0.570      SignalP-noTM
DRAMP00129|Plantaricin_9   0.111  22  0.169  11  0.333   6  0.282   0.222 N
0.570      SignalP-noTM
DRAMP00136|Enterocin_10    0.139  22  0.222  11  0.634   4  0.491   0.349 N
0.570      SignalP-noTM
DRAMP00171|Lactocyclicin_11  0.149  39  0.139  39  0.190  38  0.111   0.128
N  0.510      SignalP-TM
DRAMP00173|Leucocyclicin_12  0.148  39  0.183  13  0.382   7  0.324   0.235
N  0.510      SignalP-TM
```

Αποτελέσματα προβλέψεων SignalP σε Gram-negative οργανισμούς

```
# SignalP-4.1 gram+ predictions
# name                    Cmax  pos  Ymax  pos  Smax  pos  Smean   D     ?
Dmaxcut    Networks-used
DRAMP00032|Ruminococcin_1  0.103  12  0.102  11  0.128   1  0.100   0.101 N
0.450      SignalP-TM
DRAMP00063|Lantibiotic_2   0.107  20  0.099  11  0.149   2  0.084   0.093 N
0.450      SignalP-TM
DRAMP00068|Aureocin_3      0.143  21  0.163  21  0.312   5  0.207   0.180 N
0.450      SignalP-TM
DRAMP00074|Enterocin_4     0.136  24  0.136  24  0.231   1  0.129   0.133 N
0.450      SignalP-TM
DRAMP00089|Bacteriocin_5   0.318  24  0.285  24  0.339  20  0.215   0.258 N
0.450      SignalP-TM
DRAMP00126|Plantaricin_6   0.103  21  0.111  11  0.174   1  0.116   0.113 N
0.450      SignalP-TM
DRAMP00127|Plantaricin_7   0.102   5  0.118  11  0.184   1  0.125   0.121 N
0.450      SignalP-TM
DRAMP00128|Plantaricin_8   0.102   2  0.088  25  0.101  25  0.067   0.080 N
0.450      SignalP-TM
DRAMP00129|Plantaricin_9   0.102  30  0.131  11  0.270   1  0.146   0.137 N
0.450      SignalP-TM
DRAMP00136|Enterocin_10    0.165  28  0.211  28  0.433  23  0.225   0.216 N
0.450      SignalP-TM
```

Αποτελέσματα προβλέψεων SignalP σε Gram-positive οργανισμούς

Ανάλογο είναι το περιεχόμενο των αρχείων για όλες τις κατηγορίες πεπτιδίων που εμφανίζουν κάποια βιοδραστηριότητα καθώς και για το αρνητικό σύνολο δεδομένων το οποίο επίσης εξετάστηκε.

Στη συνέχεια της διαδικασίας, προκειμένου να ανακτηθούν και να καταχωρηθούν προς επεξεργασία μόνο τα υπολογισμένα scores, C-score, Y-score, S-score, Mean S και D-score αναπτύχθηκε κατάλληλος αλγόριθμος σε γλώσσα Python (Παράρτημα Ι: signalPprocess.py). Ο αλγόριθμος αυτός λαμβάνει σαν είσοδο ολα τα αρχεία που εξάγονται στο πρηγούμενο βήμα (συνολικα 21 αρχεία) και αποθηκεύει σε ενα μοναδικό αρχείο τις τιμές των παραπάνων scores για όλες τις κατηγορίες του συνόλου εκπαίδευσης. Τελικά, το αρχείο περιέχει ένα μέρος από τα χαρακτηριστικά με βάση τα οποία εκπαιδεύεται ο αλγόριθμος πρόβλεψης των αντιμικροβιακών πεπτιδίων.

4.2.2 Ανάλυση πεπτιδικής αλληλουχίας

Οι πεπτιδικές αλληλουχίες μελετήθηκαν ως προς κάποια χαρακτηριστικά τα οποία έχει παρατηρηθεί οτι επηρεάζουν τη βιοδραστηριότητα τους. Το **ProtParam** είναι ένα module που έχει αναπτυχθεί στη γλώσσα προγραμματισμού Python και συγκεκριμένα αποτελεί τμήμα της βιβλιοθήκης **Biopython**, ανήκει στο πακέτο (package) **SeqUtils** και αποσκοπεί στον υπολογισμό ποικίλων φυσικοχημικών ιδιοτήτων που μπορούν να προκύψουν από μια πρωτεϊνική αλληλουχία(Gasteiger et al., 2005). Ένα σύνολο εργαλείων που είναι διαθέσιμα στην κλάση *ProteinAnalysis* επιτυγχάνουν τον υπολογισμό της συχνότητας των αμινοξικών καταλοίπων (amino acid frequencies), το μοριακό βάρος (molecular weight), την αρωματικότητα (aromaticity), την ευελιξία (flexibility), το ισοηλεκτρικό σημείο (isoelectric point), το δείκτη αστάθειας (instability index), το κλάσμα της έλικας (helix_fraction), το κλάσμα της στροφής (turn_fraction), το κλάσμα της β- πτυχωτής επιφάνειας (sheet_fraction). Επίσης, μια επιπλέον κλάση του πακέτου SeqUtils, η protParamData, επιτρέπει τον υπολογισμό της υδροφοβικότητας (hydrophobicity).

Τα διαθέσιμα εργαλεία είναι τα ακόλουθα:

- **count_amino_acids** : Καταμέτρηση του αριθμού των επαναλήψεων ενός αμινοξικού καταλοιπού την ακολουθία.
- **molecular_weight** : Υπολογισμός του μοριακού βάρους μιας πρωτεΐνης.
- **aromaticity** : Υπολογισμός της τιμής αρωματικότητας μιας πρωτεΐνης σύμφωνα με τους Lobry & Gautier (Lobry & Gautier, 1994). Πρόκειται για τη σχετική συχνότητα εμφάνισης των αρωματικών καταλοίπων Φαινυλαλανίνη και Τρυπτοφάνη και Τυροσίνη(Phe + Trp + Tyr).
- **instability_index** : Μέθοδος που εξετάζει την σταθερότητα μιας πρωτεΐνης σύμφωνα με τους Guruprasad *et al*(1990, Protein Engineering, 4, 155-161). Κάθε τιμή άνω του 40 υποδεικνύει ότι η πρωτεΐνη είναι ασταθής. Τα αποτελέσματα, σύμφωνα με την παράπανω έρευνα, οδηγούν στο συμπέρασμα ότι η σταθερότητα μιας πρωτεΐνης καθορίζεται από τη σειρά συγκεκριμένων αμινοξικών καταλοίπων στην ακολουθία της.
- **Flexibility** : Υλοποίηση της μεθόδου των Vihinen *et al* (Vihinen, Torkkila, & Riikonen, 1994). Η ευελιξία μιας πρωτεΐνης έχει καθοριστικό ρόλο στις βιοχημικές λειτουργίες της κατάλυσης, της δέσμευσης και της αλλοστερίωσης(allostery), της

ικανότητας δηλαδή που διαθέτει ένα μόριο – συνδέτης (ligand), το οποίο προσδένεται σε μια θέση υποδοχής μιας πρωτεΐνης, να μεταβάλλει το σχήμα, τη διάταξη και κατ'επέκταση τη δραστηριότητα της. Τα επιφενειακά κατάλοιπα σε μια ακολουθία δύνανται να αντιπροσωπεύουν πιθανούς επιτόπους, οπότε η ανάλυση της ευελιξίας μιας πρωτεΐνης θα μπορούσε να συνεισφέρει στον εντοπισμό τους. Ειδικά, η πρόβλεψη των αντιγονικών περιοχών σε μια πρωτεΐνη με βάση μόνο την ακολουθία της θα βοηθούσε σημαντικά στην παραγωγή εμβολίων.

- **isoelectric point** : Μέθοδος που χρησιμοποιεί το module IsoelectricPoint για τον υπολογισμό του ισοηλεκτρικού σημείου(pI) μιας πρωτεΐνης. Το σημείο αυτό είναι το PH όπου ένα μόριο δε φέρει καθόλου ηλεκτρικό φορτίο, ενώ πάνω από το σημείο αυτό φέρει αρνητικό και κάτω από το σημείο αυτό φέρει θετικό φορτίο. Το pI είναι σημαντικός παράγοντας καθώς μπορεί να επηρεάσει τη διαλυτότητα της πρωτεΐνης σε συγκεκριμένες τιμές PH.
- **secondary_structure_fraction** : Μέθοδος που επιστρέφει μία λίστα με το κλάσμα των αμινοξέων που τείνουν να βρίσκονται σε μία από τις τρεις γνωστές δευτεροταγείς δομές, έλικα, στροφή ή β-φύλλο.
    - Τα συνήθη αμινοξέα σε έλικα είναι τα Βαλίνη, Ισολευκίνη, Τυροσίνη, Φαινυλαλανίνη, Τρυπτοφάνη, Λευκίνη (V,I,Y,F,W,L)
    - Τα συνήθη αμινοξέα σε στροφή είναι τα Ασπαραγίνη, Προλίνη, Γλουταμίνη, Σερίνη (N,P,G,S).
    - Τα συνήθη αμινοξέα σε β-φύλλο είναι τα Γλουταμικό οξύ, Μεθειονίνη, Αλανίνη, Λευκίνη (E,M,A,L).
- **Hydrophobicity:** Με χρήση της μεθόδου protein_scale που παρέχεται στο ProtParam module υπολογίζεται η υδροφοβικότητα μια αμινοξικής αλληλουχίας. Η κλήση της συγκεκριμένης συνάρτησης επιστρέφει μια λίστα από τιμές για κάθε θέση της αλληλουχίας και τελικά υπολογίζεται ο μέσος όρος των τιμών αυτών ώστε να εκφράζει τη μέση τιμή της υδροφοβικότητας. Τα υδρόφοβα αμινοξέα επηρεάζουν σε μεγάλο βαθμό το πρωτεϊνικό δίπλωμα, τις αλληλεπιδράσεις των πρωτεϊνών με την κυτταρική μεμβράνη καθώς και τις αλληλεπιδράσεις των πρωτεϊνικών υπομονάδων που προσδένονται στους υποδοχείς (Seshadri Sundararajan et al., 2012a)

Για την ανάλυση των ακολουθιών αναπτύχθηκε κατάλληλος αλγόριθμος σε γλώσσα προγραμματισμού Python (Παράρτημα Ι: calculateCharacter.py). Το εκτελέσιμο πρόγραμμα δέχεται ως είσοδο προς επεξεργασία όλα τα αρχεία σε fasta μορφή με τα πεπτίδια σε κάθε μια απο τις 7 κατηγορίες και εξάγει 7 αντίστοιχα αρχεία τα οποία περιέχουν τις υπολογισμένες τιμές που προέκυψαν από την ανάλυση των ακολουθιών με βάση τα παραπάνω εργαλεία του ProtParam module. Παραλληλα, για κάθε βιοδραστηριότητα ανατίθεται ένας μοναδικός αριθμός απο το 0 εως το 6.

Το περιεχόμενο των αρχείων που παράγονται έχει την ακόλουθη μορφή, για την κατηγορία των αντιβακτηριακών πεπτιδίων.

```
GNGVLKTISHECNMNTWQFLFTCC
molecular weight : 2747.1555
amino acid frequency : {'A': 0, 'C': 3, 'E': 1, 'D': 0, 'G': 2, 'F': 2,
'I': 1, 'H': 1, 'K': 1, 'M': 1, 'L': 2, 'N': 3, 'Q': 1, 'P': 0, 'S': 1,
'R': 0, 'T': 3, 'W': 1, 'V': 1, 'Y': 0}
aromaticity : 0.125
instability_index : 1.6125
flexibility : [1.0294523809523812, 0.9783214285714286, 0.9843571428571429,
1.0162619047619048, 0.9821309523809525, 1.026011904761905,
0.9630595238095239, 1.0236666666666667, 0.9789880952380953,
1.0116666666666667, 0.9814999999999999, 0.9623214285714287,
0.9967261904761904, 0.9421904761904761, 0.959047619047619]
sequence average flexibility: 0.989046825397
isoelectric point : 6.72235107422
secondary structure fraction : (0.29166666666666663, 0.24999999999999997,
0.16666666666666666)
sequence hydrophobicity : 0.0025

SSSGWLCTLTIECGTIICACR
molecular weight : 2217.6086
amino acid frequency : {'A': 1, 'C': 4, 'E': 1, 'D': 0, 'G': 2, 'F': 0,
'I': 3, 'H': 0, 'K': 0, 'M': 0, 'L': 2, 'N': 0, 'Q': 0, 'P': 0, 'S': 3,
'R': 1, 'T': 3, 'W': 1, 'V': 0, 'Y': 0}
aromaticity : 0.047619047619
instability_index : 113.871428571
flexibility : [0.9793214285714287, 0.9564880952380952, 0.9707857142857141,
0.9467857142857143, 0.9719880952380952, 0.9582857142857144,
1.0107738095238097, 0.9502976190476191, 1.003297619047619,
0.9701309523809524, 0.9554761904761904, 0.9576666666666666]
sequence average flexibility: 0.969274801587
isoelectric point : 5.71026611328
secondary structure fraction : (0.2857142857142857, 0.23809523809523808,
0.19047619047619047)
sequence hydrophobicity : 1.00476190476
```

Αποτελέσματα εκτέλεσης εργαλείων ProtParam module σε αντιβακτηριακά πεπτίδια

Όμοια είναι η μορφή των αρχείων και για τις υπόλοιπες κατηγορίες.

Παράλληλα με τη δημιουργία των εφτά διαφορετικών αρχείων με το χαρακτηριστικό όνομα «ονομα_κατηγοριας_ PrParamRes.fasta», δημιουργείται και ένα τελικό αρχείο με την ονομασία «CompletesequenceCharacteristics.csv» οι στήλες του οποίου αντιπροσωπεύουν το μοναδικό αριθμό (id) της ακολουθίας, την αμινοξική αλληλουχία, το χαρακτηριστικό αριθμό της βιοδραστηριότητας και τις τιμές που υπολογίστηκαν με τη χρήση των εργαλείων του ProtParam module, με 20 στήλες εξ αυτών να αντιστοιχούν σε ένα αμινοξικό κατάλοιπο.

Παρατηρώντας κάποιες ενδεικτικές τιμές του αρχείου που εξάγεται από τον υπολογισμό των χαρακτηριστικών επιβεβαιώνεται ότι η μέγιστη τιμή εμφάνισης των κατιονικών αμινοξικών καταλοίπων Λυσίνης (K), Αργινίνης (R), και των αμφιπαθητικών καταλοίπων Τυροσίνης (Y), Μεθειονίνης (M) υπολογίστηκε στο αντιμικροβιακό πεπτίδιο που εντοπίστηκε στον οργανισμό *Macadamia integrifolia* και κατέχει αναγνωρισμένη αντιβακτηριακή και αντιμυκητιακή δράση, σύμφωνα με την αντίστοιχη εγγραφή στη Uniprot. Επίσης, ο μέγιστος δείκτης αστάθειας 170.473 υπολογίστηκε σε πεπτίδιο που δε φέρει κάποια βιοδραστηριότητα.

Πίνακας 1: Μέγιστες τιμές Lys, Tyr, Arg, Met

Fasta ID	Max Lys	Max Arg	Max Tyr	Max Met
sp|Q9SPL3|AMP23_MACIN	26	70	24	11

Στη συνέχεια το αρχείο αυτό συνενώθηκε με το αρχείο που εξήχθη στο προηγούμενο βήμα με τα αποτελέσματα του SignalP. Τελικά, στο αρχείο που δόθηκε προς επεξεργασία, peptides_dataset.txt, αφαιρέθηκαν οι στήλες με το id, την αλληλουχία και τη βιοδραστηριότητα. Η στήλη με τις τιμές βιοδραστηριότητας αποθηκεύτηκε σε ένα ξεχωριστό αρχείο, peptides_labels.txt αφού μετατράπηκε η στήλη σε γραμμή. Ομοίως, αντιστράφηκαν οι γραμμές με τις στήλες στο peptides_dataset.txt ώστε οι στήλες να αντιστοιχούν σε κάθε πεπτίδιο και οι αντίστοιχες γραμμές στα υπολογιζόμενα χαρακτηριστικά του. Για την καλύτερη εκπαίδευση του αλγόριθμου τα δεδομένα στο τελικό αρχείο ανακατεύτηκαν, ώστε να μη βρίσκονται συνεχόμενα πεπτίδια που ανήκουν στην ίδια κατηγορία (Παράρτημα Ι: suffle_csv_rows.py).

Ο αλγόριθμος SVR που χρησιμοποιήθηκε για την εκπόνηση της διπλωματικής εργασίας ονομάζεται ***biomarker_discovery_modeller_peptides*** και πρόκειται για μια

υπολογιστική προσέγγιση της εταιρείας InSyBio Ltd. Για την εκτέλεση του απαιτείται η εγκατάσταση Python 3.0 στο Linux server που θα εκτελεστεί και της βιβλιοθήκης LIBSMV που παρέχει τα απαραίτητα εργαλεία λογισμικού για την υποστήριξη των αλγορίθμων ταξινόμησης, SVM και SVR. Ο συγκεκριμένος αλγόριθμος δέχεται τα ακόλουθα δεδομένα ως είσοδο:

1. **Dataset**: Το τελικό αρχείο συνόλου δεδομένων, peptides_dataset.txt
2. **Labels**: Το αρχείο με τις κατηγορίες στην οποία ανήκει κάθε πεπτίδιο, peptides_labels.txt
3. **Population**: Τον πληθυσμό, δηλαδή τον ακέραιο αριθμό των μεμονομένων λύσεων που αναπτύσσονται παράλληλα. Στη συγκεκριμένη εκτέλεση ο πληθυσμός ορίστηκε ως 50.
4. **Generations**: Τις αναπαραγωγές, δηλαδή το μέγιστο ακέραιο αριθμό των αναπαραγωγών που επιτρέπεται για τον πληθυσμό να αναπτυχθεί. Ο αριθμός των αναπαραγωγών ορίστηκε στις 200.
5. **Two-points crossover probability**: Την πιθανότητα χρήσης χειριστή διασταύρωσης 2 σημείων, με τιμή 0.45.
6. **Arithmetic crossover probability**: Την πιθανότητα χρήσης χειριστή αριθμητικής διασταύρωσης, με τιμή 0.45.
7. **Mutation probability**: Την πιθανότητα εφαρμογής gaussian χειριστή μετάλλαξης, με τιμή 0.05.
8. **Gaussian mutation variance proportion**: Το ποσοστό
   ‖max_val – min_val‖ που χρησιμοποιείται ως η διακύμανση του gaussian χειριστή μετάλλαξης, με τιμή 0.1.
9. **Goal significances**: Το αρχείο, το οποίο αποτελείται από μια γραμμή με τα καθορισμένα βάρη για κάθε βέλτιστο στόχο. Τα βάρη που τέθηκαν ήταν 1 100 1.
10. **Number of folds**: Έναν ακέραιο αριθμό που αντιπροσωπεύει τα k folds για k-cross validation. Το σύνολο εκπαίδευσης χωρίζεται σε k υποσύνολα, με το ένα υποσύνολο να αποτελεί το test dataset και τα υπόλοιπα k-1 να αποτελούν κάθε φορά τα training datasets. Ένα μοντέλο εκπαιδεύεται από τα training datasets και κατόπιν αυτό αξιολογείται από το test dataset. Η βαθμολογία αξιόλογησης κρατείται και το μοντέλο απορρίπτεται. Η διαδικασία επαναλαμβάνεται k φορές, ώστε όλα τα υποσύνολα να αποτελέσουν test datasets. Στο τέλος, η συνολική ικανότητα του μοντέλου συνοψίζεται λαμβάνοντας υπόψη τα δείγματα από τις βαθμολογίες αξιολόγησης των μοντέλων. Στη συγκεκριμένη εκτέλεση εφαρμόστηκε 10-fold cross validation.

Στο αρχικό στάδιο της εκτέλεσης του αλγόριθμου τα δεδομένα υφίστανται κανονικοποίση ώστε να μην εμφανίζουν μεγάλες αποκλίσεις στις τιμές των χαρακτηριστικών. Για κάθε χαρακτηριστικό ανακτάται η μέγιστη τιμή και στη συνέχεια οι τιμές όλων των υπόλοιπων χαρακτηριστικών διαιρούνται με αυτή τη μέγιστη τιμή με αποτέλεσμα τελικά το εύρος των τιμών να κυμαίνεται στο διάστημα μεταξύ -1 και 1.

Παράλληλα ο αλγόριθμος διαχειρίζεται το πρόβλημα των ελλιπών τιμών θέτοντας ως τιμή, σε αυτή την περίπτωση το μέσο όρο των υπαρχουσών τιμών που έχουν υπολογιστεί για τα υπόλοιπα χαρακτηριστικά.

Τελικά, η εκτέλεση του αλγόριθμου εκπαίδευσης biomarker_discovery_modeller_peptides πραγματοποιήθηκε με την ακόλουθη εντολή στη γραμμή εντολών του Linux server. Το αρχείο output_log.txt αποτελεί το όνομα του αρχείο στο οποίο καταγράφονται κάποια αποτελέσματα κατά τη διαδικασία εκτέλεσης.

```
python3 biomarker_discovery_modeller_peptides.py peptides_dataset.txt
peptides_labels.txt 50 200 0.45 0.45 0.05 0.1
peptides_goal_significances.txt 10 > output_log.txt
```

4.3 Μετρικές αξιολόγησης αποτελεσμάτων

Ο αλγόριθμος **biomarker_discovery_modeller_peptides** εκτελέστηκε πέντε φορές για όσο το δυνατόν καλύτερη αξιοπιστία και αξιολόγηση των αποτελεσμάτων. Η κάθε εκτέλεση παρήγαγε τα εξής αρχεία ως δεδομένα εξόδου σε μορφή txt, με την ημερομηνία να αναφέρεται στην ημερομηνία δημιουργίας του κάθε αρχείου, ωστε να είναι δυνατή η διάκριση μεταξύ τους:

1. Ένα σύνολο από μοντέλα svm, που συνιστούν τα μοντέλα που παρήχθησαν από τα διαφορετικά fold trainings της βέλτιστης λύσης που αποθηκεύεται στο φάκελο classification_models με περιεχόμενο ένα αρχείο model_2018_06_22
2. Η μέση και η καλύτερη απόδοση average_performance_2018_06_21 και best_performance_2018_06_21,αντίστοιχα
3. Η βέλτιστη λύση ως ένα αρχείο διανυσμάτων (float vector), best_solution_2018_06_21

4. Οι pareto λύσεις ως διανύσματα(float vectors) σε ένα αρχείο, best_solutions_2018_06_21
5. Οι τελικές λύσεις, final_solutions_2018_06_21
6. Το dataset, normalized_2018_06_21
7. Το κανονικοποιημένο dataset στο οποίο καταλογίζονται ελλίπουσες τιμές, normalized_missing_values_completed_2018_06_21
8. selected_normalized_2018_06_22
9. selected_normalized_missing_values_completed_2018_06_22
10. Το αρχείο με τις μέγιστες τιμές για κάθε μελετώμενο χαρακτηριστικό, maximums
11. Το συμπληρωματικό αρχείο με στατιστικές πληροφορίες για τη σύγκλιση της μεθόδου, output_log

Η αξιολόγηση της απόδοσης του αλγόριθμου εκπαίδευσης biomarker_discovery_modeller_peptides πραγματοποιήθηκε με την εφαρμογή ενός νέου αλγόριθμου που αναπτύχθηκε σε γλώσσα προγραμματισμού Python 3.0 και ονομάζεται **apply_svr_model**. Για το σκοπό αυτό δημιουργήθηκε το test dataset, ακολουθώντας την ίδια διαδικασία διεξαγωγής του training dataset, με την εκτέλεση των ίδιων scripts υπολογισμού των χαρακτηριστικών. Παράλληλα, από τα χαρακτηριστικά αφαιρέθηκε η στήλη με τις κατηγορίες της βιοδραστηριότητας στην οποία ανήκει κάθε πεπτίδιο. Η στήλη βιοδραστηριοτήτων κρατήθηκε σε ξεχωριστό αρχείο, test_labels.txt, για την πραγματοποίηση της σύγκρισης μεταξύ των παραγόμενων αποτελεσμάτων της εφαρμογής του apply_svr_model και των πραγματικών κατηγοριών βιοδραστηριότητας των αντίστοιχων πεπτιδίων. Με τη βοήθεια του κώδικα apply_svr_model εφαρμόζεται στο test dataset το καλύτερο μοντέλο εκπαίδευσης (best trained model).

Παρόμοια με το **biomarker_discovery_modeller_peptides,** η εκτέλεση του αλγόριθμου **apply_svr_model** επαναλήφθηκε 5 φορές.

Ο παραπάνω αλγόριθμος δέχεται ως είσοδο 7 ορίσματα:

1. **training_dataset_preprocessed.txt**

   Πρόκειται για το αρχείο normalized_missing_values_completed που παρήχθη κατά την εκτέλεση του αλγόριθμου στη στάδιο της εκπαίδευσης. Σε κάθε μια από τις πέντε εκτελέσεις το αρχείο αυτό έχει τα ίδια δεδομένα.

2. **model.txt**

   Το αρχείο αυτό αναφέρεται στο αντίστοιχο model αρχείο που αποθηκεύεται στο φάκελο classification_models κατά την εκτέλεση του αλγόριθμου εκπαίδευσης και είναι διαφορετικό για κάθε εκτέλεση.

3. **maximums.txt**

   Στο αρχείο αυτό αποθηκεύεται η μέγιστη τιμή την οποία λαμβάνει κάθε χαρακτηριστικό και αντιστοιχεί στο αρχείο **maximums** που παράγεται κατά το προηγούμενο βήμα, με τη διαφορά ότι οι τιμές τοποθετούνται σε γράμμη και χωρίζονται με tabs.

4. **minimums.txt**

   Αντίστοιχα, στο αρχείο αυτό αποθηκεύεται η ελάχιστη τιμή την οποία λαμβάνει κάθε χαρακτηριστικό, όπως υπολογίστηκε από το training dataset.

5. **test_set_outputs.txt**

   Το αρχείο αυτό είναι αρχικά κενό και σε αυτό καταγράφονται τα δεδομένα εξόδου, δηλαδή οι υπολογιζόμενες κατηγορίες βιοδραστηριότητας για το test dataset.

6. **test_dataset.txt**

   Το αρχείο με τα δεδομένα για τη δοκιμαστική διαδικασία.

7. **best_solution.txt**

   Το αρχείο αυτό αναφέρεται στο αντίστοιχο best_solution το οποίο παράγεται κατά την εκτέλεση του αλγόριθμου εκπαίδευσης και επίσης διαφοροποιείται μεταξύ των πέντε εκτελέσεων.

Σε κάθε εκτέλεση παράγεται ένα αρχείο(**test_set_outputs.txt**) στο οποίο καταγράφεται η κατηγορία βιοδραστηριότητας στην οποία προβλέφθηκε ότι ανήκει κάθε ένα από τα δοκιμαστικά πεπτιδία. Το περιεχόμενο του κάθε αρχείου συνοψίζεται ενδεικτικά στον παρακάτω πίνακα για τα πρώτα 30 πεπτίδια.

Πίνακας 2: Πραγματικές και προβλεπόμενες κατηγορίες βιοδραστηριότητας για τα 30 πρώτα πεπτίδια του test dataset, για κάθε μία από τις 5 επαναλήψεις

Κατηγορία Βιοδραστηριότητας	Εκτέλεση 1	Εκτέλεση 2	Εκτέλεση 3	Εκτέλεση 4	Εκτέλεση 5
0	6	1	1	4	6
0	6	6	6	6	6
0	2	0	0	6	0
0	0	0	0	6	0
0	6	0	6	6	6
0	4	6	6	6	6
0	6	1	4	6	4
3	3	1	6	6	1
3	1	1	1	6	1
3	1	1	1	6	1
3	3	1	6	1	3
3	2	1	1	3	3
3	3	6	6	4	1
3	0	1	2	1	1
3	1	1	4	6	6
3	1	1	6	6	1
3	2	3	4	6	2
3	1	1	1	1	1
3	5	3	6	6	3
3	2	3	6	3	6
3	0	1	1	1	1
3	1	1	1	1	1
3	0	0	6	1	1
3	1	1	1	1	3
3	3	3	1	1	4
3	6	3	1	2	2
3	4	1	4	1	1
3	3	2	6	2	1
3	1	1	1	1	1
3	1	1	1	1	1

Στη συνέχεια, υπολογίστηκε το ποσοστό σωστής πρόβλεψης συγκρίνοντας τα πραγματικά με τα προβλεπόμενα αποτελέσματα. Τέλος, υπολογίσθηκε ο τελικός μέσος όρος και η διασπορά για τις πέντε εξαγόμενες ακρίβειες σε κάθε βήμα, προκειμένου να αξιολογηθεί η μέθοδος συνολικά. Η **διασπορά** ή **διακύμανση** (variance) ορίζεται ως ο μέσος όρος των τετραγώνων των αποκλίσεων $\mathbf{t_i}$ από τη μέση τιμή τους **x** και υπολογίζεται μέσω της σχέσης:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(ti - x)^2$$

όπου n το μέγεθος του πληθυσμού.

Στη συγκεκριμένη περίπτωση n = 5, και εκφράζει τον αριθμό των επαναλήψεων. Η σύγκριση και οι μαθηματικοί υπολογισμοί ολοκληρώθηκαν με την ανάπτυξη ενός script σε γλώσσα προγραμματισμού Python (Παράρτημα Ι: precision_all.py). Το εκτελέσιμο δέχεται ως είσοδο το αρχείο test_labels.txt, με τις γνωστές κατηγορίες βιοδραστηριότητας καθώς και τα πέντε παραγόμενα test_set_outputs.txt. Τα αποτελέσματα του προγράμματος, δηλαδή ο μέσος όρος και η διασπορά, καταγράφονται στο αρχείο **precisionResults.txt.**

```
python precision_all.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
```

Εντολή εκτέλεσης precision_all.py

```
Calculated precision for the 1st test output is : 57.57
Calculated precision for the 2nd test output is : 65.08
Calculated precision for the 3rd test output is : 66.33
Calculated precision for the 4th test output is : 71.12
Calculated precision for the 5th test output is : 73.75

Calculated average precision is : 66.77
Calculated variance is : 31.07
Calculated SD is : 5.57
```

Πίνακας 3: Υπολογιζόμενες ακρίβειες για κάθε μια από τις πέντε εκτελέσεις

Εκτέλεση 1	Εκτέλεση 2	Εκτέλεση 3	Εκτέλεση 4	Εκτέλεση 5
57.57	65.08	66.33	71.12	73.75

Πίνακας 4: Υπολογιζόμενος μέσος όρος, διασπορά και τυπική απόκλιση της ακρίβειας των αποτελεσμάτων του αλγόριθμου πρόβλεψης apply_svr_model

Μέσος Όρος	Διασπορά	Τυπική Απόκλιση
66.77	31.07	5.57

Η ποιότητα του αλγόριθμου πρόβλεψης αξιολογήθηκε και με βάση τις μετρικές της Ευαισθησίας (Sensitivity), της Ειδικότητας (Specificity) και του συντελεστή συσχέτισης Matthew's (MCC). Οι προαναφερθείσες μετρικές έχουν καλύτερη εφαρμογή στην αξιολόγηση δυαδικών ταξινομητών και εξετάζουν το κατά πόσον ένα δείγμα προβλέφθηκε οτι ορθώς ανήκει ή δεν ανήκει σε μια κατηγορία. Στη συγκεκριμένη περίπτωση όμως δεν εξετάζεται μόνο αν ένα πεπτίδιο εμφανίζει ή όχι κάποια βιοδραστηριότητα αλλά και σε ποια κατηγορία ανήκει. Οπότε, για να μπορέσουν να υπολογιστούν οι παραπάνω μετρικές, σε μια προσπάθεια απόκτησης καλύτερης εικόνας της απόδοσης του αλγόριθμου, εξετάστηκε κάθε κατηγορία χωριστά, δηλαδή αν ένα πεπτίδιο σωστά προβλέφθηκε οτι παρουσιάζει τη βιοδραστηριότητα η οποία του αποδόθηκε κατά τη διεξαγωγή του αλγόριθμου apply_svr_model.

Αρχικά, για κάθε μία απο τις 5 εκτελέσεις υπολογίσθηκαν οι αριθμοί :

1) True Positive TP
2) True Negative TN
3) False Positive FP και
4) False Negative FN

για κάθε μία από τις πιθανές βιοδραστηριότητες : αντικαρκινική, αντιμικροβιακή, αντιβακτηριακή, αντιμυκητιακή, αντιϊκή, εντομοκτόνα ή καμία-βιοδραστηριότητα. (Παράρτημα Ι: true_positive.py, true_negative.py, false-positive.py, false_negative.py).

Πίνακας 5: TP,FP,FN,TN για πεπτίδια με **αντικαρκινική** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**1** (TP)	**195** (FP)
Αρνητική παρατήρηση	**6** (FN)	**9960** (TN)

Πίνακας 6: TP,FP,FN,TN για πεπτίδια με **αντιμικροβιακή** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**381** (TP)	**999** (FP)
Αρνητική παρατήρηση	**802** (FN)	**7980** (TN)

Πίνακας 7: TP,FP,FN,TN για πεπτίδια με **αντιβακτηριακή** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**17** (TP)	**386** (FP)
Αρνητική παρατήρηση	**182** (FN)	**9577** (TN)

Πίνακας 8: TP,FP,FN,TN για πεπτίδια με **αντιμυκητιακή** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**16** (TP)	**372** (FP)
Αρνητική παρατήρηση	**143** (FN)	**9631** (TN)

Πίνακας 9: TP,FP,FN,TN για πεπτίδια με **αντιϊική** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**2** (TP)	**1270** (FP)
Αρνητική παρατήρηση	**11** (FN)	**8879** (TN)

Πίνακας 10: TP,FP,FN,TN για πεπτίδια με **εντομοκτόνα** βιοδραστηριότητα για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**0** (TP)	**578** (FP)
Αρνητική παρατήρηση	**0** (FN)	**9584** (TN)

Πίνακας 11: TP,FP,FN,TN για πεπτίδια χωρίς **καμία βιοδραστηριότητα** για την 1η εκτέλεση

Συνθήκη / **Παρατήρηση**	**Θετική συνθήκη**	**Αρνητική συνθήκη**
Θετική παρατήρηση	**5433** (TP)	**512** (FP)
Αρνητική παρατήρηση	**3168** (FN)	**1049** (TN)

Ομοίως για τις επόμενες 4 εκτελέσεις οι αντίστοιχες τιμές που προέκυψαν ήταν:

Πίνακας 12: TP,FP,FN,TN για πεπτίδια με **αντικαρκινική** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	3	2	0	2
FP	142	75	49	46
FN	4	5	7	5
TN	10013	10080	10106	10109

Πίνακας 13: TP,FP,FN,TN για πεπτίδια με **αντιμικροβιακή** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	339	425	286	321
FP	590	897	526	399
FN	844	758	897	862
TN	8389	8082	8453	8580

Πίνακας 14: TP,FP,FN,TN για πεπτίδια με **αντιβακτηριακή** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	21	11	18	20
FP	345	243	152	141
FN	178	188	181	179
TN	9618	9720	9811	9822

Πίνακας 15: TP,FP,FN,TN για πεπτίδια με **αντιμυκητιακή** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	16	4	7	7
FP	326	244	149	151
FN	143	155	152	152
TN	9677	9759	9854	9852

Πίνακας 16: TP,FP,FN,TN για πεπτίδια με **αντιϊική** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	3	3	3	1
FP	938	873	918	769
FN	10	10	10	12
TN	9211	9276	9231	9380

Πίνακας 17: TP,FP,FN,TN για πεπτίδια με **εντομοκτόνα** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	0	0	0	0
FP	575	428	321	349
FN	0	0	0	0
TN	9587	9734	9841	9813

Πίνακας 18: TP,FP,FN,TN για πεπτίδια **χωρίς καμία** βιοδραστηριότητα

	Εκτέλεση 2	**Εκτέλεση 3**	**Εκτέλεση 4**	**Εκτέλεση 5**
TP	6231	6295	6913	7143
FP	633	662	820	813
FN	2370	2306	1688	1458
TN	928	899	741	748

Σε κάθε μια από τις παραπάνω περιπτώσεις αθροίζοντας τα TP + FN κάθε κατηγορίας προκύπτει, όπως αναμένονταν, ο συνολικός αριθμός των πεπτιδίων από αυτήν την κατηγορία που συμμετείχαν στο test dataset. Χαρακτηριστικό παράδειγμα είναι η περίπτωση των εντομοκτόνων όπου TP + FN = 0, δεδομένου οτι στο test dataset δεν συμπεριλαμβάνονται πεπτίδια αυτής της βιοδραστηριότητας. Με βάση τους παραπάνω πίνακες εξάγονται τελικά η Ευαισθησία, η Ειδικότητα και ο συντελεστής συσχέτισης Matthew's της μεθόδου σύμφωνα με τους τύπους:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$MCC = \frac{TP * TN - FP * FN}{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))^{1/2}}$$

Τα αποτελέσματα συγκεντρώνονται παρακάτω:

Πίνακας 19: Μετρικές αξιολόγησης για πεπτίδια με **αντικαρκινική** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.14	0.43	0.23	0	0.23
ΕΙΔΙΚΟΤΗΤΑ	0.98	0.99	0.99	0.995	0.995
MCC	0.02	0.09	0.08	-0.002	0.108

Πίνακας 20: Μετρικές αξιολόγησης για πεπτίδια με **αντιμικροβιακή** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.32	0.29	0.36	0.24	0.27
ΕΙΔΙΚΟΤΗΤΑ	0.88	0.93	0.9	0.94	0.96
MCC	0.197	0.246	0.247	0.217	0.284

Πίνακας 21: Μετρικές αξιολόγησης για πεπτίδια με **αντιβακτηριακή** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.086	0.1	0.06	0.09	0.1
ΕΙΔΙΚΟΤΗΤΑ	0.96	0.97	0.98	0.99	0.985
MCC	0.033	0.053	0.027	0.081	0.096

Πίνακας 22: Μετρικές αξιολόγησης για πεπτίδια με **αντιμυκητιακή** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.1	0.1	0.025	0.044	0.044
ΕΙΔΙΚΟΤΗΤΑ	0.96	0.97	0.975	0.985	0.96
MCC	0.041	0.047	0.001	0.029	0.029

Πίνακας 23: Μετρικές αξιολόγησης για πεπτίδια με **αντιϊική** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.15	0.23	0.23	0.23	0.08
ΕΙΔΙΚΟΤΗΤΑ	0.87	0.91	0.914	0.91	0.92
MCC	0.003	0.017	0.018	0.017	0.0

Πίνακας 24: Μετρικές αξιολόγησης για πεπτίδια με **εντομοκτόνα** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0	0	0	0	0
ΕΙΔΙΚΟΤΗΤΑ	0.94	0.97	0.96	0.97	0.97

Πίνακας 25: Μετρικές αξιολόγησης για πεπτίδια **χωρίς καμια** βιοδραστηριότητα

ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ	**ΕΚΤΕΛΕΣΗ 1**	**ΕΚΤΕΛΕΣΗ 2**	**ΕΚΤΕΛΕΣΗ 3**	**ΕΚΤΕΛΕΣΗ 4**	**ΕΚΤΕΛΕΣΗ 5**
ΕΥΑΙΣΘΗΣΙΑ	0.63	0.72	0.73	0.8	0.83
ΕΙΔΙΚΟΤΗΤΑ	0.67	0.6	0.58	0.47	0.48
MCC	0.384	0.246	0.239	0.235	0.271

Ομοίως με τη διαδικασία υπολογισμού της ακρίβειας, λαμβάνεται ο μέσος όρος των μετρικών αξιολόγησης στις 5 επαναλήψεις και τα αποτελέσματα συνοψίζονται στον ακόλουθο πίνακα. Για την κατηγορία «insecticidal» δεν αξιολογούνται οι μετρικές δεδομένης της απουσίας αντιπροσωπευτικών πεπτιδίων στο μελετώμενο test dataset.

Πίνακας 26: Μέσος όρος μετρικών αξιολόγησης για όλες τις κατηγορίες πεπτιδίων που συμμετέχουν στο test dataset

Μετρικές Αξιολόγησης	**Anticancer**	**Antimicrobial**	**Antibacterial**	**Antifungal**	**Antiviral**	**Non-AMP**
Ευαισθησία	0.206	0.3	0.087	0.06	0.184	0.74
Ειδικότητα	0.99	0.92	0.98	0.97	0.9	0.56
MCC	0.06	0.24	0.06	0.03	0.011	0.28

ΚΕΦΑΛΑΙΟ 5

5.1 Συμπεράσματα

Μια καινοτόμος μεθοδολογία παρουσιάστηκε για την πρόβλεψη της βιοδραστηριότητας των πεπτιδίων. Το σημαντικότερο στοιχείο της υπολογιστικής προσέγγισης που εισήχθη είναι η χρήση ενός αλγόριθμου δύο φάσεων, που συνδυάζει πολλαπλούς ταξινομητές nu-SVR και τα παραγόμενα αποτελέσματα καταλήγουν σε μια μοναδική έξοδο, με βάση την αρχή της ensemble μεθοδολογίας. Επιπλέον, η χρήση του εξελικτικού γενετικού αλγόριθμου βελτιστοποιεί τις μελετώμενες παραμέτρους και δημιουργεί συμπαγή υποσύνολα χαρακτηριστικών.

Η ποιότητα της βελτιωμένης μεθοδολογίας Ensemble GASVR αξιολογήθηκε με τη λήψη του μέσου όρου και της διασποράς της ακρίβειας των αποτελεσμάτων που προέκυψαν σε κάθε μια απο τις πέντε εκτελέσεις του αλγόριθμου apply_svr_model, ο οποίος λαμβάνοντας ως είσοδο ενα test dataset παράγει στην έξοδο του ένα αρχείο με τις κατηγορίες 0 (αντικαρκινική), 1 (αντιμικροβιακή), 2 (αντιβακτηριακή), 3 (αντιμυκητιακή), 4 (αντιϊική), 5 (εντομοκτόνα), 6 (καμία βιοδραστηριότητα) στις οποίες προέβλεψε οτι ανήκει κάποιο πεπτίδιο. Ο μέσος όρος διαμορφθώθηκε σε 66.77 και η διασπορά σε 31.07, ωστόσο είναι εμφανές απο τα αποτελέσματα στον Πίνακα 3 η ακρίβεια των προβλέψεων αυξάνεται όσο αυξάνεται ο αριθμός των επαναλήψεων, με την $5^{η}$ εκτέλεση να κατέχει τη μεγαλύτερη ακρίβεια με 73.75. Παράλληλα, έγινε μια προσπάθεια υπολογισμού των μετρικών αξιολόγησης Ευαισθησίας, Ειδικότητας και Συντελεστή Συσχέτισης Matthew's για κάθε μια κατηγορία ξεχωριστά και της λήψης, τελικά, του μέσου όρου των αντίστοιχων τιμών για την εξαγωγή μιας συνολικής εικόνας αξιοπιστίας της μεθόδου.

Οι τιμές Ευαισθησίας και Ειδικότητας, σε περιπτώσεις ταξινομητών, έχει αποδειχθεί οτι εξαρτώνται σε μεγάλο βαθμό από τα συγκεκριμένα κατώφλια ταξινόμησης (classifier thresholds) τα οποία έχουν τεθεί μεταξύ των κατηγοριών. Επομένως, μεταβάλλοντας τα κατώφλια ταξινόμησης μπορεί να επιτευχθούν διαφορετικά ποσοστά ως προς τις τιμές των παραπάνω μετρικών. Στην πλειοψηφία, παρατηρήθηκαν καλύτερες τιμές Ειδικότητας έναντι της Ευαισθησίας.

Ο βαθμός συσχέτισης MCC αντιπροσωπεύει μια τέλεια πρόβλεψη όταν είναι 1. Όταν ο βαθμός αυτός ισούται με 0 δεν αντιπροσωπεύει τίποτα περισσότερο από μια τυχαία πρόβλεψη, ενώ όταν ισούται με -1 υποδεικνύει απόλυτη διαφωνία μεταξύ πρόβλεψης και

παρατήρησης. Από τα παραπάνω αποτελέσματα είναι εμφανές ότι ο αλγόριθμος εμφανίζει ικανοποιητική απόδοση, ωστόσο είναι απαραίτητη περαιτέρω μελέτη και βελτίωση ώστε να γίνει πιο αξιόπιστος. Δεν παρατηρήθηκαν αρνητικές τιμές, ωστόσο, ο συντελεστής συσχέτισης κυμαίνεται κοντά σε μηδενικές τιμές, αρκετά μακριά από τη μονάδα. Επομένως, είναι πιθανό να υπάρχει αρκετή τυχαιότητα στην πρόβλεψη.

Σε μια προσπάθεια να εξεταστεί αν οι παράμετροι εισόδου του αλγόριθμου επηρεάζουν την απόδοσή του, ο αλγόριθμος εκτελέστηκε μεταβάλλοντας κάποιες από τις παραμέτρους προκειμένου να διαπιστωθεί αν βελτιώνονται τα αποτελέσματα και κατά πόσον αυτή η βελτίωση είναι αξιόλογη. Ορίζοντας τον αριθμό των επαναλήψεων σε 100, τον πληθυσμό σε 30 και κατόπιν πέντε επαναλήψεων, στην 5η εκτέλεση ανακτήθηκε η μεγαλύτερη ακρίβεια με τιμή 82.37, ενώ συνολικά παρατηρήθηκε μια αύξηση, μικρής τάξης, με το μέσο όρο της ακρίβειας να λαμβάνει την τιμή 71.28. Η εφαρμογή εναλλακτικών συνδυασμών αναφορικά και με τις πιθανότητες διασταύρωσης και μετάλλαξης, ίσως, να οδηγήσει σε καλύτερα αποτελέσματα πρόβλεψης ως απόρροια νέων πιο αξιόπιστων μοντέλων.

Ένας σημαντικός παράγοντας που θα μπορούσε να επηρεάσει την αξιοπιστία της μεθοδολογίας είναι το γεγονός ότι αρκετά πεπτίδια ανήκουν σε παραπάνω από μια κατηγορίες βιοδραστηριότητας. Κατά την συγκέντρωση των πεπτιδίων από τις βάσεις δεδομένων, υπήρχαν, για παράδειγμα, περιπτώσεις πεπτιδίων με αναγωρισμένη αντιμικροβιακή και αντικαρκινική δράση. Επομένως, ο αλγόριθμος apply_svr_model, που προβλέπει μόνο μια πιθανή κατηγορία για κάθε πεπτίδιο είναι πιθανό να το κατέταξε σε μια από τις δύο αναγνωρισμένες κατηγορίες και κατά αυτόν τον τρόπο δεν υπήρξε πλήρης ταύτιση της πραγματικής κατηγορίας βιοδραστηριότητας με την προβλεπόμενη. Αυτό που θα είχε ίσως περισσότερη σημασία είναι να εξεταστεί το κατά πόσον προβλέφθηκε σωστά ότι πεπτίδια που έχουν κάποια βιοδραστηριότητα, βρέθηκε να ανήκουν σε μία από τις έξι κατηγορίες και, ομοίως, πεπτίδια που δεν εμφανίζουν κάποια βιοδραστηριότητα κατά πόσον προβλέφθηκε σωστά ότι ανήκουν στην καταγορία «καμια βιοδραστηριότητα».

Οι βάσεις δεδομένων αντιμικροβιακών πεπτιδίων συνεχώς εμπλουτίζονται με νέα πεπτίδια που ανακαλύπτονται. Μια πρόταση ώστε να εκπαιδευτεί ακόμα καλύτερα ο αλγόριθμος ταξινόμησης στο μέλλον είναι η συλλογή όσο τον δυνατόν μεγαλύτερου αριθμού αντιπροσωπευτικών δειγμάτων από κάθε κατηγορία ως το θετικό σύνολο εκπαίδευσης. Χαρακτηριστικό είναι το γεγονός ότι, συγκριτικά με τις υπόλοιπες

κατηγορίες, τα non-AMPs είχαν τις καλύτερες τιμές Ευαισθησίας/Ειδικότητας, το οποίο μπορεί να οφείλεται στη μεγάλη συμμετοχή τους στο test dataset.

Αξιοσημείωτο είναι και το γεγονός ότι εκτός από τον αριθμό των πεπτιδίων, η εκπαίδευση του αλγόριθμου βασίστηκε στη μελέτη κάποιων συγκεκριμένων χαρακτηριστικών των AMPs και non-AMPs πεπτιδίων τα οποία χρησιμοποιούν οι περισσότερες μέθοδοι πρόβλεψης πεπτιδικής βιοδραστηριότητας που έχουν αναπτυχθεί ως σήμερα, όπως η υδροφοβικότητα, η αμινοξική σύνθεση, το ισοηλεκτρικό σημείο, η δευτεροταγής δομή, η αρωματικότητα, κοκ. Τα χαρακτηριστικά αυτά εμφανίζουν άμεση σύνδεση μεταξύ τους και η μεταβολή ακόμα και μιας απο τις παραμέτρους μπορεί να συντελέσει στη μείωση ή ακόμα και την απενεργοποίηση της οποιασδήποτε βιοδραστηριότητας. Μια εξαιρετική πρόκληση για την εξέλιξη της έρευνας στον τομέα της μηχανικής μάθησης είναι η μελέτη και εξαγωγή όλο και περισσότερων χαρακτηριστικών που εμφανίζουν τα πεπτίδια αυτά. Όσο αυξάνονται οι παράμετροι εκπαίδευσης ενός μοντέλου και όσο πιο αντιπροσωπευτικά δείγματα παραχωρούνται ως είσοδος, τόσο καλύτερα θα μπορούν να εκπαιδευτούν και να προκύψουν αξιόπιστοι αλγόριθμοι ταξινόμησης, δεδομένου οτι ακόμα και οι πιο απλές μέθοδοι δύνανται να παρουσιάζουν μέγιστη απόδοση όταν τροφοδοτούνται με τα κατάλληλα χαρακτηριστικά.

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

AMP	AntiMicrobial Peptide
APD	Antimicrobial Peptide Database
AUC	Area Under the Curve
CAMP	Collection of Anti-Microbial Peptides
DAMPD	Dragon Antimicrobial Peptide Database
DRAMP	Data Repository of AntiMicrobial Peptides
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
MCC	Matthews Correlation Coefficient
RF	Random Forest
ROC	Receiver Operating Characteristic Curve
SVM	Support Vector Machine
SVR	Support Vector Regression
TN	True Negative
TP	True Positive

ΠΑΡΑΡΤΗΜΑ Ι

Οι κώδικες προγραμματισμού, σε γλώσσα προγραμματισμού Python 2.7, που αναπτύχθηκαν κατάλληλα για την επεξεργασία των δεδομένων εισόδου του μοντέλου ταξινόμησης και τον υπολογισμό των χαρακτηριστικών για την εκπαίδευση του αλγόριθμου ταξινόμησης είναι διαθέσιμοι παρακάτω.

1. selectNegative.py

Αλγόριθμος για την επιβεβαίωση των αρνητικών πεπτιδίων εκπαίδευσης. Το αρχείο fasta που προέκυψε από τα αποτελέσματα της Pfam για την εξαγωγή της οικογένειας των μη αντιμικροβιακών πεπτιδίων δίνεται ως είσοδος. Με χρήση του εργαλείου hmmer3 ελέγχεται αν περιέχουν το χαρακτηριστικό «Antimicrobial» οι οικογένεις που βρέθηκε να ανήκουν τα πεπτίδια. Απαραίτητη προϋπόθεση η εγκάτασταση της βιβλιοθήκης hmmer από τον σύνδεσμο http://hmmer.org/

```
# give the file with the peptides pfam results as input to be scanned
# towards hmmer
# in order to verify that they don't belong to any antimicrobial family
# selectNegative.py hmmer-3.1b2-linux-intel-x86_64/PfamResults.txt
# note that hmmer-3 library should be installed for the script to run

if __name__ == "__main__":
    pfam_results = sys.argv[1]
    input_file = open(pfam_results,"r")
    outputFile = open("hmmerResults.txt","w")
    negativeFile = input_file.read()
    negativeFile1 = negativeFile.split("//")
    count = 0
    for line in negativeFile1:
        if "Antimicrobial" in line:
            outputFile.write("Antimicrobial family found\n")
            count+=1

    if count == 0:
        outputFile.write("No Antimicrobial family found\n")
```

2. replaceX.py

Βοηθητικός αλγόριθμος για την αντικατάσταση των «X» που βρέθηκαν στις πεπτιδικές αλληλουχίες. Δέχεται 2 ορίσματα, το όνομα του αρχείου fasta των πεπτιδίων και το

επιθυμητό όνομα του νέου αρχείου που θα παραχθεί, στο οποίο κάθε «X» έχει αντικατασταθεί με κάθε ένα από τα 20 αμινοξέα.

```
# replaceX.py takes as an input a fasta file and a desired output fasta
# filename
# searches for X appearances in the peptide sequences in the input file
# if found replaces them with one of the 20 aminoacids
# all the sequences even replaced or not are written to the new output
# file.

import sys
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

if __name__ == "__main__":
    antiAMPS = open(sys.argv[1],"r")
    replacedXAMPS = open(sys.argv[2],"w")
    count = 0
    aminoAcids =
['G','A','L','M','F','W','K','Q','E','S','P','V','I','C','Y','H','R','N','D
','T']
    position_X = list()
    headerNumber = 0

    for record in SeqIO.parse(antiAMPS,"fasta"):
        header = str(record.id)
        sequence = str(record.seq)
        if("X" not in sequence):
            replacedXAMPS.write(">"+header + "\n" + sequence + "\n" )
        else:
            count+=1
            for i in range(len(sequence)):
                if sequence.startswith('X', i):
                    position_X.append(i)
            print position_X
            while (position_X):
                pos = position_X.pop(0)
                for amino in aminoAcids:
                    headerNumber+=1
                    replacedXAMPS.write(">"+header +"_"+str(headerNumber)+
"\n"+sequence.replace(sequence[pos],amino)+"\n")

    print count
```

3. replaceXBUZ.py

Ομοίως, βοηθητικός αλγόριθμος για την αντικατάσταση των «X», «B», «U», «Z» που βρέθηκαν στις πεπτιδικές αλληλουχίες των αρνητικών δεδομένων εκπαίδευσης. Δέχεται 2 ορίσματα, το όνομα του αρχείου fasta των πεπτιδίων και το επιθυμητό όνομα του νέου αρχείου που θα παραχθεί, στο οποίο κάθε «X», «B», «U», «Z» έχει αντικατασταθεί με κάθε ένα από τα 20 αμινοξέα.

```
# replaceXBUZ.py replaces XBUZ with the valid 20 aminoacids for non-AMPs #
# only
# takes as an input the fasta file
# and produces the output fasta file with the sequences modified
# accordingly

import sys
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

if __name__ == "__main__":
    antiAMPS = (sys.argv[1],"r")
    replacedXAMPS =_open(sys.argv[2],"w")

    count = 0
    countAminos = 0
    deletedAminos = ["X","B","Z","U"]
    for record in SeqIO.parse(antiAMPS,"fasta"):
        header = str(record.id)
        sequence = str(record.seq)
        for amino in deletedAminos:
            if(amino in sequence):
                countAminos+=1
        if(countAminos == 0):
            replacedXAMPS.write(">"+header + "\n" + sequence + "\n" )
            count +=1
        countAminos = 0
    print count
```

4. countBalanced.py

Αλγόριθμος για τον υπολογισμο του συνολικού αριθμού των πεπτιδίων κατόπιν αντικατάστασης των αμινοξικών καταλοίπων που είχαν επισημανθεί ως «X» στις πεπτιδικές αλληλουχίες.

```
# countBalanced.py counts the peptides in each category after X replacement

from Bio import SeqIO
from Bio.Alphabet import IUPAC

cnt_antimicrobial = 0
cnt_antibacterial = 0
cnt_anticancer = 0
cnt_antifungal = 0
cnt_antiviral = 0
cnt_insecticidal = 0
cnt_nonAmps = 0
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/antibacteria
l.fasta","fasta"):
    cnt_antibacterial+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/anticancer.f
asta","fasta"):
    cnt_anticancer+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/antifungal.f
asta","fasta"):
    cnt_antifungal+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/antimicrobia
l.fasta","fasta"):
    cnt_antimicrobial+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/antiviral.fa
sta","fasta"):
    cnt_antiviral+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/insecticidal
.fasta","fasta"):
    cnt_insecticidal+=1
for record in
SeqIO.parse("/home/ekolotourou/diploma/dataset/datasetBalanced/nonAMPS.fast
a","fasta"):
    cnt_nonAmps+=1
print("antibacterial " +str(cnt_antibacterial))
print("anticancer " + str(cnt_anticancer))
print("antifungal " + str(cnt_antifungal))
print("antimicrobial " + str(cnt_antimicrobial))
print("antiviral " + str(cnt_antiviral))
print("insecticidal " + str(cnt_insecticidal))
print ("sum of AMPs " +
str(cnt_antibacterial+cnt_anticancer+cnt_antifungal+cnt_antimicrobial+cnt_a
ntiviral+cnt_insecticidal))
print("nonAmps "+ str(cnt_nonAmps))
```

5. modify_header.py

Ο παρακάτω αλγόριθμος αναπτύχθηκε για την αντικαταστάση της επικεφαλίδας header στην fasta μορφή της αλληλουχίας, προκειμένου να διαφοροποιηθούν μεταξύ τους τα πεπτίδια όπου λόγω της αντικατάστασης των «X» κατέληξαν να παραχθούν επιπλέον αλληλουχίες με το ίδιο header. Σκοπός ήταν να γίνει εφικτή η σωστή εφαρμογή του προγράμματος SIGNALP, το οποίο επεργάζεται και διαφοροποιεί τις πεπτιδικές αλληλουχίες με βάση το header.

```
# modify_header.py modify the fasta sequence headers by adding a number at
the end
# so as to de distinct for SIGNALP to work properly
# it takes two arguments one for the fasta file and one filename for the
desired output.

import sys
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

if __name__ == "__main__":
    antiAMPS = open(sys.argv[1],"r")
    replacedHeaderAMPS = open(sys.argv[2],"w")
    headerNumber = 0

    for record in SeqIO.parse(antiAMPS,"fasta"):
        header = str(record.id)
        sequence = str(record.seq)
        headerNumber+=1
        replacedHeaderAMPS.write(">"+header +"_"+str(headerNumber)+
"\n"+sequence+"\n")
```

6. calculateCharacter.py

Ο βασικός αλγόριθμος υπολογισμού χαρακτηριστικών. Λαμβάνει ως είσοδο τις πεπτιδικές αλληλουχίες σε fasta μορφή και υπολογίζει το μοριακό βάρος, την αρωματικότητα, το δείκτη αστάθειας, την ευελιξία, το ισοηλεκτρικό σημείο, τη δευτεροταγή δομή, την υδροφοβικότητα και την αμινοξική συχνότητα. Στο τέλος παράγει ένα αρχείο csv με στήλες τις τιμές που έλαβε το κάθε χαρακτηριστικό καθώς και μια επιπλέον στήλη που αναφέρει την κατηγορία της βιοδραστηριότητας.

```
# calculateCharacter.py  takes all the balanced fasta files from all
antimicrobial categories and
# calculates the features: molecular weight/aromaticity/instability
index/flexibility/isoelectric point
# /helix secondary strucrure/b-sheet secondary structure/turn secondary
structure/hydrophobicity
# /bioactivity/aminoacid frequency
# Finally produces a csv file named CompletesequenceCharacteristics

import sys
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
from Bio.SeqUtils.ProtParam import ProteinAnalysis
from Bio.SeqUtils import ProtParamData
import pandas as pd



if __name__ == "__main__":
    AMPsDataset =
{"/home/ekolotourou/diploma/dataset/datasetBalanced/antibacterial.fasta":"/
home/ekolotourou/diploma/Results/characteristics/antibacterialPrParamRes.fa
sta",

"/home/ekolotourou/diploma/dataset/datasetBalanced/anticancer.fasta":"/home
/ekolotourou/diploma/Results/characteristics/anticancerPrParamRes.fasta",

"/home/ekolotourou/diploma/dataset/datasetBalanced/antifungal.fasta":"/home
/ekolotourou/diploma/Results/characteristics/antifungalPrParamRes.fasta",

"/home/ekolotourou/diploma/dataset/datasetBalanced/antimicrobial.fasta":"/h
ome/ekolotourou/diploma/Results/characteristics/antimicrobialPrParamRes.fas
ta",

"/home/ekolotourou/diploma/dataset/datasetBalanced/antiviral.fasta":"/home/
ekolotourou/diploma/Results/characteristics/antiviralPrParamRes.fasta",

"/home/ekolotourou/diploma/dataset/datasetBalanced/insecticidal.fasta":"/ho
me/ekolotourou/diploma/Results/characteristics/insecticidalPrParamRes.fasta
",

"/home/ekolotourou/diploma/dataset/datasetBalanced/nonAMPS.fasta":"/home/ek
olotourou/diploma/Results/characteristics/nonAMPSPrParamRes.fasta"}

    sequence = list()
    mw = list()
    aroma = list()
    instab = list()
    isoelec = list()
    ssf_helix = list()
    ssf_sheet = list()
    ssf_turn = list()

    average_hydro = list()
    average_flex = list()
    bioact = list()
    frAminoAcidsList = list()
    frames = list()
    ids = list()
    count antic = 0
```

```
    for data,fastaResults in AMPsDataset.items():
        anti_dataset = open(data)
        anti_results  = open(fastaResults,"w")
        print anti_dataset
        print anti_results

        if "cancer" in data:
            bioactivity = 0
        elif "microbial" in data:
            bioactivity = 1
        elif "bacterial" in data:
            bioactivity = 2
        elif "fungal" in data:
            bioactivity = 3
        elif "viral" in data:
            bioactivity = 4
        elif "insecticidal" in data:
            bioactivity = 5
        else:
            bioactivity = 6 #for negative dataset
        print bioactivity

        for record in SeqIO.parse(anti_dataset,"fasta"):
            analysed_seq = ProteinAnalysis(str(record.seq))
            seq_length = len(str(record.seq))
            #values initialization
            total_hydrophobocity = 0.0
            total_flexibility = 0.0
            average_hydrophobocity = 0.0
            average_flexibility = 0.0

            flex = list()
            hydrophobicity = list()

            #sequence
            sequence.append(record.seq)
            ids.append(record.id)
            #molecular weight
            mw.append(analysed_seq.molecular_weight())
            anti_results.write(str(record.seq) +"\n" + "molecular weight :
" +str(analysed_seq.molecular_weight())+ "\n" )
            #amino acid frequence
            anti_results.write("amino acid frequency : " +
str(analysed_seq.count_amino_acids())+ "\n" )

            frequency_amino = analysed_seq.count_amino_acids()
            frAminoAcidsList.append(frequency_amino.values())

            fr_ac = pd.DataFrame(frequency_amino,index=[count_antic])
            frames.append(fr_ac)

            #aromaticity
            aroma.append(analysed_seq.aromaticity())
            anti_results.write("aromaticity : "+
str(analysed_seq.aromaticity())+"\n")
```

```
            #instability
            instab.append(analysed_seq.instability_index())
            anti_results.write("instability_index : "+
str(analysed_seq.instability_index())+"\n")

            #flexibilty
            flex = analysed_seq.flexibility()
            anti_results.write("flexibility : "+
str(analysed_seq.flexibility())+"\n")
            for value in range(len(flex)):
                total_flexibility += flex[value]
            if len(flex)!=0:
                average_flexibility = total_flexibility/len(flex)
                average_flex.append(average_flexibility)
            else:
                average_flexibility = total_flexibility
                average_flex.append(total_flexibility)
            anti_results.write("sequence average flexibility: " +
str(average_flexibility) + "\n" )

            #isoelectric point
            isoelec.append(analysed_seq.isoelectric_point())
            anti_results.write("isoelectric point : "+
str(analysed_seq.isoelectric_point())+"\n")

            #secondary structure fraction
            struct =
str(analysed_seq.secondary_structure_fraction()).split(',')
            ssf_helix.append(struct[0][1:])
            ssf_sheet.append(struct[1])
            ssf_turn.append(struct[2][0:-1])
            anti_results.write("secondary structure fraction : "+
str(analysed_seq.secondary_structure_fraction())+"\n")

            #hydrophobicity
            hydrophobicity = analysed_seq.protein_scale( ProtParamData.kd,
5 ,1.0)
            for value in range(len(hydrophobicity)):
                total_hydrophobocity += hydrophobicity[value]
            average_hydrophobocity = total_hydrophobocity/seq_length
            average_hydro.append(average_hydrophobocity)
            anti_results.write("sequence hydrophobicity : " +
str(average_hydrophobocity) + "\n\n" )

            bioact.append(bioactivity)
            count_antic+=1

        characteristics = {'ids' : ids,
                    'sequence' : sequence,
                        'molecular_weight' : mw,
                    'aromaticity' : aroma,
                    'instability_index' : instab,
                    'flexibility' : average_flex,
                    'isoelectric_point' : isoelec,
                    'helix secondary structure' : ssf_helix,
                    'b-sheet secondary structure' : ssf_sheet,
                    'turn secondary structure' : ssf_turn,
                    'hydrophobicity' : average_hydro,
                    'bioactivity' : bioact}
```

```
        charact = pd.DataFrame(characteristics)
        charact.to_csv("sequenceCharacteristics.csv",columns =
['ids','sequence','molecular_weight','aromaticity','instability_index','fle
xibility','isoelectric_point','helix secondary structure','b-sheet
secondary structure','turn secondary
structure','hydrophobicity','bioactivity'])
        result = pd.concat(frames)
        result.to_csv("aminoAcidFreq.csv")

        #merge csvs sequenceCharacteristics.csv and aminoAcidFreq.csv
        joinedCsv = charact.join(result)
        joinedCsv.to_csv("CompletesequenceCharacteristics.csv",columns =
['ids','sequence','molecular_weight','aromaticity','instability_index','fle
xibility','isoelectric_point','helix secondary structure','b-sheet
secondary structure','turn secondary
structure','hydrophobicity','bioactivity',

'A','C','E','D','G','F','I','H','K','M','L','N','Q','P','S','R','T','W','V'
,'Y'])
```

7. signalPcalc.py

Αλγόριθμος εφαρμογής του προγράμματος SignalP. Δίδονται ως είσοδοι τα πεπτίδια σε κάθε κατηγορία σε fasta μορφή και αφού εφαρμοστεί το SignalP παράγονται οι αντίστοιχες έξοδοι για κάθε μία απο τις 3 περιπτώσεις, ευκαρυότες, βακτήρια θετικά κατά Gram και βακτήρια αρνητικά κατά Gram.

```
# signalPcalc.py SignalP predicts the presence and location of signal
peptide cleavage sites
# in amino acid sequences  from  different organisms:  Gram-positive
bacteria,
# Gram-negative bacteria, and eukaryotes.  The method incorporates a
prediction
# of cleavage sites and a signal peptide/non-signal peptide prediction
based on
# a combination of several artificial neural networks.

import sys
import os

if __name__ == "__main__":

    os.system('./signalp -t euk -f short data/antibacterial.fasta >
signal_results/antibacterialSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/antibacterial.fasta >
signal_results/antibacterialSgpos.short_out')
    os.system('./signalp -t gram- -f short data/antibacterial.fasta >
signal_results/antibacterialSgneg.short_out')
    os.system('./signalp -t euk -f short data/anticancer.fasta >
signal_results/anticancerSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/anticancer.fasta >
signal_results/anticancerSgpos.short_out')
    os.system('./signalp -t gram- -f short data/anticancer.fasta >
signal_results/anticancerSgneg.short_out')
    os.system('./signalp -t euk -f short data/antifungal.fasta >
signal_results/antifungalSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/antifungal.fasta >
signal_results/antifungalSgpos.short_out')
    os.system('./signalp -t gram- -f short data/antifungal.fasta >
signal_results/antifungalSgneg.short_out')
    os.system('./signalp -t euk -f short data/antimicrobial.fasta >
signal_results/antimicrobialSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/antimicrobial.fasta >
signal_results/antimicrobialSgpos.short_out')
    os.system('./signalp -t gram- -f short data/antimicrobial.fasta >
signal_results/antimicrobialSgneg.short_out')
    os.system('./signalp -t euk -f short data/antiviral.fasta >
signal_results/antiviralSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/antiviral.fasta >
signal_results/antiviralSgpos.short_out')
    os.system('./signalp -t gram- -f short data/antiviral.fasta >
signal_results/antiviralSgneg.short_out')
    os.system('./signalp -t euk -f short data/insecticidal.fasta >
signal_results/insecticidalSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/insecticidal.fasta >
signal_results/insecticidalSgpos.short_out')
    os.system('./signalp -t gram- -f short data/insecticidal.fasta >
signal_results/insecticidalSgneg.short_out')
    os.system('./signalp -t euk -f short data/nonAMPS.fasta >
signal_results/nonAmpsSeuk.short_out')
    os.system('./signalp -t gram+ -f short data/nonAMPS.fasta >
signal_results/nonAmpsSgpos.short_out')
os.system('./signalp -t gram- -f short data/nonAMPS.fasta >
signal_results/nonAmpsSgneg.short_out')
```

8. signalPprocess.py

Τα αρχεία που παράγονται μέσω του αλγόριθμου signalPcalc.py, επεξεργάζονται περαιτέρω προκειμένου να κρατηθούν μόνο οι βαθμολογίες (scores) που μας ενδιαφέρουν για κάθε πεπτίδιο. Τα scores αποθηκεύονται σε ένα αρχείο csv, το οποίο στη συνέχεια συνενώνεται με το csv αρχείο των χαρακτηριστικών και δημιουργείται κατ' αυτόν τον τρόπο το τελικό training dataset.

```
# signalProcess.py the results from signalPcalc are processed
# keeps only scores : Cmax  Ymax Smax  Smean  D for each of Gram-positive
bacteria,Gram-negative bacteria,and eukaryotes
# the final results are added to AllSignalPFeatures.csv

import sys
import os
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC
import pandas as pd

if __name__ == "__main__":
    signalPDataResults =
[("signal_results/antibacterialSeuk.short_out","signal_results/antibacteria
lSgpos.short_out","signal_results/antibacterialSgneg.short_out"),

("signal_results/anticancerSeuk.short_out","signal_results/anticancerSgpos.
short_out","signal_results/anticancerSgneg.short_out"),

("signal_results/antifungalSeuk.short_out","signal_results/antifungalSgpos.
short_out","signal_results/antifungalSgneg.short_out"),

("signal_results/antimicrobialSeuk.short_out","signal_results/antimicrobial
Sgpos.short_out","signal_results/antimicrobialSgneg.short_out"),

("signal_results/antiviralSeuk.short_out","signal_results/antiviralSgpos.sh
ort_out","signal_results/antiviralSgneg.short_out"),

("signal_results/insecticidalSeuk.short_out","signal_results/insecticidalSg
pos.short_out","signal_results/insecticidalSgneg.short_out"),

("signal_results/nonAmpsSeuk.short_out","signal_results/nonAmpsSgpos.short_
out","signal_results/nonAmpsSgneg.short_out")]
    data = list()
    ids = list()
    #eukaryotes
    CmaxE = list()
    YmaxE = list()
    SmaxE = list()
    SmeanE = list()
    DE = list()
    #gram positive
    CmaxGP = list()
    YmaxGP = list()
    SmaxGP = list()
    SmeanGP = list()
    DGP =list()
    #gram negative
    CmaxGN = list()
    YmaxGN = list()
    SmaxGN = list()
    SmeanGN = list()
    DGN = list()

    final_ = list()
    for dataFile in signalPDataResults:
        res_Euk = open(dataFile[0])
        res_GramP = open(dataFile[1])
        res_GramN = open(dataFile[2])
        print res_Euk
```

```
print res_GramP
        print res_GramP

        #create csv for Eukariotes
        for line in res_Euk:
            if not line.startswith("#"):
                data = line.split()
                ids.append(data[0])
                CmaxE.append(data[1])
                YmaxE.append(data[3])
                SmaxE.append(data[5])
                SmeanE.append(data[7])
                DE.append(data[8])


        #create csv for Gram+
        for line in res_GramP:
            if not line.startswith("#"):
                data = line.split()

                CmaxGP.append(data[1])
                YmaxGP.append(data[3])
                SmaxGP.append(data[5])
                SmeanGP.append(data[7])
                DGP.append(data[8])

        #create csv for Gram -
        for line in res_GramN:
            if not line.startswith("#"):
                data = line.split()

                CmaxGN.append(data[1])
                YmaxGN.append(data[3])
                SmaxGN.append(data[5])
                SmeanGN.append(data[7])
                DGN.append(data[8])

    SignalFeatures = {'ids' : ids,
                      'Cmax_Eukariotes' : CmaxE,
                      'Ymax_Eukariotes' : YmaxE,
                      'Smax_Eukariotes' : SmaxE,
                      'Smean_Eukariotes' : SmeanE,
                      'D_Eukariotes' : DE,
                      'Cmax_Gram+' : CmaxGP,
                      'Ymax_Gram+' : YmaxGP,
                      'Smax_Gram+' : YmaxGP,
                      'Smean_Gram+' : SmeanGP,
                      'D_Gram+' : DGP,
                      'Cmax_Gram-' : CmaxGN,
                      'Ymax_Gram-' : YmaxGN,
                      'Smax_Gram-' : SmaxGN,
                      'Smean_Gram-' : SmeanGN,
                      'D_Gram-' : DGN
                      }
    final = pd.DataFrame(SignalFeatures)
    final.to_csv("signal_results/AllSignalPFeatures.csv",columns =
['ids','Cmax_Eukariotes','Ymax_Eukariotes','Smax_Eukariotes','Smean_Eukario
tes','D_Eukariotes',

'Cmax_Gram+','Ymax_Gram+','Smax_Gram+','Smean_Gram+','D_Gram+','Cmax_Gram-
','Ymax_Gram-','Smax_Gram-','Smean_Gram-','D_Gram-'])
```

9. suffle_csv_rows.py

Βοηθητικός αλγόριθμος απο «ανακατεύει» τα δεδομένα στο training dataset προκειμένου να μη βρίσκονται ομαδοποιημένα και συνεχόμενα τα πεπτίδια ανάλογα με την κατηγορία βιοδραστηριότητας τους.

```
# suffle_csv_rows.py: suffles the final csv file with all the features
calculated and produces AllCalculatedFeatures_Suffled.csv

import pandas as pd
import sys
import os

df =
pd.read_csv('/home/ekolotourou/diploma/python_codes/AllCalculatedFeatures.c
sv', header=None)
ds = df.sample(frac=1)
ds.to_csv('AllCalculatedFeatures_Suffled.csv')
```

10. precision_all.py

Ο αλγόριθμος παραγωγής των τελικών αποτελεσμάτων για την αξιολόγηση της απόδοσης του αλγόριθμου biomarker_discovery_modeller. Λαμβάνει ως είσοδο τα αρχεία με τις πραγματικές και τις προβλεπόμενες κατηγορίες βιοδραστηριοτητας, από τον αλγόριθμο apply_svr_model, και παράγει ως έξοδο το μέσο όρο της ακρίβειας των 5 εκτελέσεων και τη διασπορά.

```
# calculates precision and variation for the predicted bioactivity
classification
# input files:
# 1. test_labels.txt : the file with the test dataset known bioactivity
# 2,3,4,5,6 test_set_outputs: the output files from apply_svr_model for
each of the 5 runs
# Results are written to precisionResults.txt

#python precision_all.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
import sys
import math

if __name__ == "__main__":
    test_labels       = sys.argv[1]
    test_set_outputs1 = sys.argv[2]
    test_set_outputs2 = sys.argv[3]
    test_set_outputs3 = sys.argv[4]
    test_set_outputs4 = sys.argv[5]
    test_set_outputs5 = sys.argv[6]

    count_match1 = 0.0
    count_match2 = 0.0
    count_match3 = 0.0
    count_match4 = 0.0
    count_match5 = 0.0
    countlines1 = 0
    countlines2 = 0
    countlines3 = 0
    countlines4 = 0
    countlines5 = 0

    precision1 = 0.0
    precision2 = 0.0
    precision3 = 0.0
    precision4 = 0.0
    precision5 = 0.0
    average    = 0.0
    variation  = 0.0
    sd         = 0.0

    out_fname=open("precisionResults.txt",'w')

    #precision for the 1st test output
    with open(test_labels) as f0, open(test_set_outputs1) as f1:
        for line0, line1 in zip(f0, f1):
            countlines1 +=1
            if float(line0.rstrip()) == float(line1.rstrip()):
                count_match1 += 1
        print (count_match1)
        precision1 = (count_match1*100) / countlines1
        precision1 = float("{0:.2f}".format(precision1))
        print precision1
        out_fname.write("Calculated precision for the 1st test output is :
" +str(precision1) +"\n")

    #precision for the 2nd test output
```

```
with open(test_labels) as f0, open(test_set_outputs2) as f2:
        for line0, line2 in zip(f0, f2):
            countlines2 +=1
            if float(line0.rstrip()) == float(line2.rstrip()):
                count_match2 += 1
        print (count_match2)
        precision2 = (count_match2*100) / countlines2
        precision2 = float("{0:.2f}".format(precision2))
        print precision2
        out_fname.write("Calculated precision for the 2nd test output is :
" +str(precision2) +"\n")

    #precision for the 3rd test output
    with open(test_labels) as f0, open(test_set_outputs3) as f3:
        for line0, line3 in zip(f0, f3):
            countlines3 +=1
            if float(line0.rstrip()) == float(line3.rstrip()):
                count_match3 += 1
        print (count_match3)
        precision3 = (count_match3*100) / countlines3
        precision3 = float("{0:.2f}".format(precision3))
        print precision3
        out_fname.write("Calculated precision for the 3rd test output is :
" +str(precision3) +"\n")

    #precision for the 4th test output
    with open(test_labels) as f0, open(test_set_outputs4) as f4:
        for line0, line4 in zip(f0, f4):
            countlines4 +=1
            if float(line0.rstrip()) == float(line4.rstrip()):
                count_match4 += 1
        print (count_match4)
        precision4 = (count_match4*100) / countlines4
        precision4 = float("{0:.2f}".format(precision4))
        print precision4
        out_fname.write("Calculated precision for the 4th test output is :
" +str(precision4) +"\n")

    #precision for the 5th test output
    with open(test_labels) as f0, open(test_set_outputs5) as f5:
        for line0, line5 in zip(f0, f5):
            countlines5 +=1
            if float(line0.rstrip()) == float(line5.rstrip()):
                count_match5 += 1
        print (count_match5)
        precision5 = (count_match5*100) / countlines5
        precision5 = float("{0:.2f}".format(precision5))
        print precision5
        out_fname.write("Calculated precision for the 5th test output is :
" +str(precision5) +"\n\n")

    #average precision for the 5 values calculated above
    average = (precision1 + precision2 + precision3 + precision4 +
precision5) / 5
    average = float("{0:.2f}".format(average))
    print average
    out_fname.write ("Calculated average precision is : " + str(average) +
"\n")
```

```
    #variance calculation
    sum_diff_from_average = pow((precision1 - average),2) + pow((precision2
- average),2) + pow((precision3 - average),2) + pow((precision4 -
average),2) + pow((precision5 - average),2)
    variation = sum_diff_from_average / 5
    sd = math.sqrt(variation)

    variation = float("{0:.2f}".format(variation))
    sd = float("{0:.2f}".format(sd))
    print variation
    print sd
    out_fname.write("Calculated variance is : " +str(variation) +"\n")
    out_fname.write("Calculated SD is : " +str(sd))
```

11. true_positive.py

Αλγόριθμος για τον υπολογισμό του αριθμού True Positive για καθε μια απο τις κατηγορίες βιοδραστηριότητας.

```
# calculates true positive

#python true_positive.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
import sys
import math

if __name__ == "__main__":
    test_labels       = sys.argv[1]
    test_set_outputs1 = sys.argv[2]
    test_set_outputs2 = sys.argv[3]
    test_set_outputs3 = sys.argv[4]
    test_set_outputs4 = sys.argv[5]
    test_set_outputs5 = sys.argv[6]

    tp0 = 0
    tp1 = 0
    tp2 = 0
    tp3 = 0
    tp4 = 0
    tp5 = 0
    tp6 = 0

    out_fname=open("TPResults.txt",'w')

    #TP for the 1st test output
    with open(test_labels) as f0, open(test_set_outputs1) as f1:
        for line0, line1 in zip(f0, f1):
            if float(line0.rstrip()) == 0 and  float(line1.rstrip()) == 0:
                tp0 += 1
            elif float(line0.rstrip()) == 1 and  float(line1.rstrip()) ==
1:
                tp1 += 1
            elif float(line0.rstrip()) == 2 and  float(line1.rstrip()) ==
2:
                tp2 += 1
            elif float(line0.rstrip()) == 3 and  float(line1.rstrip()) ==
3:
                tp3+= 1
            elif float(line0.rstrip()) == 4 and  float(line1.rstrip()) ==
4:
                tp4 += 1
            elif float(line0.rstrip()) == 5 and  float(line1.rstrip()) ==
5:
                tp5 += 1
            elif float(line0.rstrip()) == 6 and  float(line1.rstrip()) ==
6:
                tp6 += 1

        out_fname.write("First output True Positive" + "\n")
        out_fname.write("TP_0: " +str(tp0) + "\n")
        out_fname.write("TP_1: " +str(tp1) + "\n")
        out_fname.write("TP_2: " +str(tp2) + "\n")
        out_fname.write("TP_3: " +str(tp3) + "\n")
        out_fname.write("TP_4: " +str(tp4) + "\n")
        out_fname.write("TP_5: " +str(tp5) + "\n")
        out_fname.write("TP_6: " +str(tp6) + "\n")

    tp0 = 0
    tp1 = 0
```

```
    tp2 = 0
    tp3 = 0
    tp4 = 0
    tp5 = 0
    tp6 = 0

    #TP for the 2nd test output
    with open(test_labels) as f0, open(test_set_outputs2) as f2:
        for line0, line2 in zip(f0, f2):
            if float(line0.rstrip()) == 0 and  float(line2.rstrip()) == 0:
                tp0 += 1
            elif float(line0.rstrip()) == 1 and  float(line2.rstrip()) ==
1:
                tp1 += 1
            elif float(line0.rstrip()) == 2 and  float(line2.rstrip()) ==
2:
                tp2 += 1
            elif float(line0.rstrip()) == 3 and  float(line2.rstrip()) ==
3:
                tp3+= 1
            elif float(line0.rstrip()) == 4 and  float(line2.rstrip()) ==
4:
                tp4 += 1
            elif float(line0.rstrip()) == 5 and  float(line2.rstrip()) ==
5:
                tp5 += 1
            elif float(line0.rstrip()) == 6 and  float(line2.rstrip()) ==
6:
                tp6 += 1

        out_fname.write("Second output True Positive" + "\n")
        out_fname.write("TP_0: " +str(tp0) + "\n")
        out_fname.write("TP_1: " +str(tp1) + "\n")
        out_fname.write("TP_2: " +str(tp2) + "\n")
        out_fname.write("TP_3: " +str(tp3) + "\n")
        out_fname.write("TP_4: " +str(tp4) + "\n")
        out_fname.write("TP_5: " +str(tp5) + "\n")
        out_fname.write("TP_6: " +str(tp6) + "\n")

    tp0 = 0
    tp1 = 0
    tp2 = 0
    tp3 = 0
    tp4 = 0
    tp5 = 0
    tp6 = 0

    #TP for the 3rd test output
    with open(test_labels) as f0, open(test_set_outputs3) as f3:
        for line0, line3 in zip(f0, f3):
            if float(line0.rstrip()) == 0 and  float(line3.rstrip()) == 0:
                tp0 += 1
            elif float(line0.rstrip()) == 1 and  float(line3.rstrip()) ==
1:
                tp1 += 1
            elif float(line0.rstrip()) == 2 and  float(line3.rstrip()) ==
2:
                tp2 += 1
            elif float(line0.rstrip()) == 3 and  float(line3.rstrip()) ==
3:
                tp3+= 1
```

```
elif float(line0.rstrip()) == 4 and  float(line3.rstrip()) == 4:
                tp4 += 1
            elif float(line0.rstrip()) == 5 and  float(line3.rstrip()) ==
5:
                tp5 += 1
            elif float(line0.rstrip()) == 6 and  float(line3.rstrip()) ==
6:
                tp6 += 1

        out_fname.write("Third output True Positive" + "\n")
        out_fname.write("TP_0: " +str(tp0) + "\n")
        out_fname.write("TP_1: " +str(tp1) + "\n")
        out_fname.write("TP_2: " +str(tp2) + "\n")
        out_fname.write("TP_3: " +str(tp3) + "\n")
        out_fname.write("TP_4: " +str(tp4) + "\n")
        out_fname.write("TP_5: " +str(tp5) + "\n")
        out_fname.write("TP_6: " +str(tp6) + "\n")

    tp0 = 0
    tp1 = 0
    tp2 = 0
    tp3 = 0
    tp4 = 0
    tp5 = 0
    tp6 = 0

    #TP for the 4th test output
    with open(test_labels) as f0, open(test_set_outputs4) as f4:
        for line0, line4 in zip(f0, f4):
            if float(line0.rstrip()) == 0 and  float(line4.rstrip()) == 0:
                tp0 += 1
            elif float(line0.rstrip()) == 1 and  float(line4.rstrip()) ==
1:
                tp1 += 1
            elif float(line0.rstrip()) == 2 and  float(line4.rstrip()) ==
2:
                tp2 += 1
            elif float(line0.rstrip()) == 3 and  float(line4.rstrip()) ==
3:
                tp3+= 1
            elif float(line0.rstrip()) == 4 and  float(line4.rstrip()) ==
4:
                tp4 += 1
            elif float(line0.rstrip()) == 5 and  float(line4.rstrip()) ==
5:
                tp5 += 1
            elif float(line0.rstrip()) == 6 and  float(line4.rstrip()) ==
6:
                tp6 += 1

        out_fname.write("Fourth output True Positive" + "\n")
        out_fname.write("TP_0: " +str(tp0) + "\n")
        out_fname.write("TP_1: " +str(tp1) + "\n" )
        out_fname.write("TP_2: " +str(tp2) + "\n")
        out_fname.write("TP_3: " +str(tp3) + "\n")
        out_fname.write("TP_4: " +str(tp4) + "\n")
        out_fname.write("TP_5: " +str(tp5) + "\n")
        out_fname.write("TP_6: " +str(tp6) + "\n")

    tp0 = 0
    tp1 = 0
```

```
tp2 = 0
    tp3 = 0
    tp4 = 0
    tp5 = 0
    tp6 = 0
    #TP for the 5th test output
    with open(test_labels) as f0, open(test_set_outputs5) as f5:
        for line0, line5 in zip(f0, f5):
            if float(line0.rstrip()) == 0 and  float(line5.rstrip()) == 0:
                tp0 += 1
            elif float(line0.rstrip()) == 1 and  float(line5.rstrip()) ==
1:
                tp1 += 1
            elif float(line0.rstrip()) == 2 and  float(line5.rstrip()) ==
2:
                tp2 += 1
            elif float(line0.rstrip()) == 3 and  float(line5.rstrip()) ==
3:
                tp3+= 1
            elif float(line0.rstrip()) == 4 and  float(line5.rstrip()) ==
4:
                tp4 += 1
            elif float(line0.rstrip()) == 5 and  float(line5.rstrip()) ==
5:
                tp5 += 1
            elif float(line0.rstrip()) == 6 and  float(line5.rstrip()) ==
6:
                tp6 += 1

        out_fname.write("Fifth output True Positive" + "\n")
        out_fname.write("TP_0: " +str(tp0) + "\n")
        out_fname.write("TP_1: " +str(tp1) + "\n")
        out_fname.write("TP_2: " +str(tp2) + "\n")
        out_fname.write("TP_3: " +str(tp3) + "\n")
        out_fname.write("TP_4: " +str(tp4) + "\n")
        out_fname.write("TP_5: " +str(tp5) + "\n")
        out_fname.write("TP_6: " +str(tp6))
```

12. true_negative.py

Αλγόριθμος για τον υπολογισμό του αριθμού True Negative για καθε μια απο τις κατηγορίες βιοδραστηριότητας.

```
# calculates true negative

#python true_negative.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
import sys
import math

if __name__ == "__main__":
    test_labels       = sys.argv[1]
    test_set_outputs1 = sys.argv[2]
    test_set_outputs2 = sys.argv[3]
    test_set_outputs3 = sys.argv[4]
    test_set_outputs4 = sys.argv[5]
    test_set_outputs5 = sys.argv[6]

    tn0 = 0
    tn1 = 0
    tn2 = 0
    tn3 = 0
    tn4 = 0
    tn5 = 0
    tn6 = 0


    out_fname=open("TNResults.txt",'w')

    #TP for the 1st test output
    with open(test_labels) as f0, open(test_set_outputs1) as f1:
        for line0, line1 in zip(f0, f1):
            if float(line0.rstrip()) != 0 and  float(line1.rstrip()) != 0:
                tn0 += 1
            if float(line0.rstrip()) !=  1 and  float(line1.rstrip()) != 1:
                tn1 += 1
            if float(line0.rstrip()) !=  2 and  float(line1.rstrip()) != 2:
                tn2 += 1
            if float(line0.rstrip()) !=  3 and  float(line1.rstrip()) != 3:
                tn3+= 1
            if float(line0.rstrip()) !=  4 and  float(line1.rstrip()) != 4:
                tn4 += 1
            if float(line0.rstrip()) !=  5 and  float(line1.rstrip()) != 5:
                tn5 += 1
            if float(line0.rstrip()) !=  6 and  float(line1.rstrip()) != 6:
                tn6 += 1

        out_fname.write("First output True Negative" + "\n")
        out_fname.write("TN_0: " +str(tn0) + "\n")
        out_fname.write("TN_1: " +str(tn1) + "\n")
        out_fname.write("TN_2: " +str(tn2) + "\n")
        out_fname.write("TN_3: " +str(tn3) + "\n")
        out_fname.write("TN_4: " +str(tn4) + "\n")
        out_fname.write("TN_5: " +str(tn5) + "\n")
        out_fname.write("TN_6: " +str(tn6) + "\n")

    tn0 = 0
    tn1 = 0
    tn2 = 0
    tn3 = 0
    tn4 = 0
    tn5 = 0
    tn6 = 0
```

```
#TP for the 2nd test output
    with open(test_labels) as f0, open(test_set_outputs2) as f2:
        for line0, line2 in zip(f0, f2):
            if float(line0.rstrip()) != 0 and  float(line2.rstrip()) != 0:
                tn0 += 1
            if float(line0.rstrip()) !=  1 and  float(line2.rstrip()) != 1:
                tn1 += 1
            if float(line0.rstrip()) !=  2 and  float(line2.rstrip()) != 2:
                tn2 += 1
            if float(line0.rstrip()) !=  3 and  float(line2.rstrip()) != 3:
                tn3+= 1
            if float(line0.rstrip()) !=  4 and  float(line2.rstrip()) != 4:
                tn4 += 1
            if float(line0.rstrip()) != 5 and  float(line2.rstrip()) != 5:
                tn5 += 1
            if float(line0.rstrip()) !=  6 and  float(line2.rstrip()) != 6:
                tn6 += 1
        out_fname.write("Second output True Negative" + "\n")
        out_fname.write("TN_0: " +str(tn0) + "\n")
        out_fname.write("TN_1: " +str(tn1) + "\n")
        out_fname.write("TN_2: " +str(tn2) + "\n")
        out_fname.write("TN_3: " +str(tn3) + "\n")
        out_fname.write("TN_4: " +str(tn4) + "\n")
        out_fname.write("TN_5: " +str(tn5) + "\n")
        out_fname.write("TN_6: " +str(tn6) + "\n")
    tn0 = 0
    tn1 = 0
    tn2 = 0
    tn3 = 0
    tn4 = 0
    tn5 = 0
    tn6 = 0
    #TP for the 3rd test output
    with open(test_labels) as f0, open(test_set_outputs3) as f3:
        for line0, line3 in zip(f0, f3):
            if float(line0.rstrip()) !=  0 and  float(line3.rstrip()) != 0:
                tn0 += 1
            if float(line0.rstrip()) !=  1 and  float(line3.rstrip()) != 1:
                tn1 += 1
            if float(line0.rstrip()) !=  2 and  float(line3.rstrip()) != 2:
                tn2 += 1
            if float(line0.rstrip()) !=  3 and  float(line3.rstrip()) != 3:
                tn3+= 1
            if float(line0.rstrip()) !=  4 and  float(line3.rstrip()) != 4:
                tn4 += 1
            if float(line0.rstrip()) !=  5 and  float(line3.rstrip()) != 5:
                tn5 += 1
            if float(line0.rstrip()) !=  6 and  float(line3.rstrip()) != 6:
                tn6 += 1
        out_fname.write("Third output True Negative" + "\n")
        out_fname.write("TN_0: " +str(tn0) + "\n")
        out_fname.write("TN_1: " +str(tn1) + "\n")
        out_fname.write("TN_2: " +str(tn2) + "\n")
        out_fname.write("TN_3: " +str(tn3) + "\n")
        out_fname.write("TN_4: " +str(tn4) + "\n")
        out_fname.write("TN_5: " +str(tn5) + "\n")
        out_fname.write("TN_6: " +str(tn6) + "\n")

    tn0 = 0
    tn1 = 0
    tn2 = 0
```

```
    tn3 = 0
    tn4 = 0
    tn5 = 0
    tn6 = 0
    #TP for the 4th test output
    with open(test_labels) as f0, open(test_set_outputs4) as f4:
        for line0, line4 in zip(f0, f4):
            if float(line0.rstrip()) !=  0 and  float(line4.rstrip()) != 0:
                tn0 += 1
            if float(line0.rstrip()) !=  1 and  float(line4.rstrip()) != 1:
                tn1 += 1
            if float(line0.rstrip()) !=  2 and  float(line4.rstrip()) != 2:
                tn2 += 1
            if float(line0.rstrip()) !=  3 and  float(line4.rstrip()) != 3:
                tn3+= 1
            if float(line0.rstrip()) !=  4 and  float(line4.rstrip()) != 4:
                tn4 += 1
            if float(line0.rstrip()) !=  5 and  float(line4.rstrip()) != 5:
                tn5 += 1
            if float(line0.rstrip()) !=  6 and  float(line4.rstrip()) != 6:
                tn6 += 1
        out_fname.write("Fourth output True Negative" + "\n")
        out_fname.write("TN_0: " +str(tn0) + "\n")
        out_fname.write("TN_1: " +str(tn1) + "\n")
        out_fname.write("TN_2: " +str(tn2) + "\n")
        out_fname.write("TN_3: " +str(tn3) + "\n")
        out_fname.write("TN_4: " +str(tn4) + "\n")
        out_fname.write("TN_5: " +str(tn5) + "\n")
        out_fname.write("TN_6: " +str(tn6) + "\n")
    tn0 = 0
    tn1 = 0
    tn2 = 0
    tn3 = 0
    tn4 = 0
    tn5 = 0
    tn6 = 0
    #TP for the 5th test output
    with open(test_labels) as f0, open(test_set_outputs5) as f5:
        for line0, line5 in zip(f0, f5):
            if float(line0.rstrip()) !=  0 and  float(line5.rstrip()) != 0:
                tn0 += 1
            if float(line0.rstrip()) !=  1 and  float(line5.rstrip()) != 1:
                tn1 += 1
            if float(line0.rstrip()) !=  2 and  float(line5.rstrip()) != 2:
                tn2 += 1
            if float(line0.rstrip()) !=  3 and  float(line5.rstrip()) != 3:
                tn3+= 1
            if float(line0.rstrip()) !=  4 and  float(line5.rstrip()) != 4:
                tn4 += 1
            if float(line0.rstrip()) !=  5 and  float(line5.rstrip()) != 5:
                tn5 += 1
            if float(line0.rstrip()) !=  6 and  float(line5.rstrip()) != 6:
                tn6 += 1
        out_fname.write("Fifth output True Negative" + "\n")
        out_fname.write("TN_0: " +str(tn0) + "\n")
        out_fname.write("TN_1: " +str(tn1) + "\n")
        out_fname.write("TN_2: " +str(tn2) + "\n")
        out_fname.write("TN_3: " +str(tn3) + "\n")
        out_fname.write("TN_4: " +str(tn4) + "\n")
        out_fname.write("TN_5: " +str(tn5) + "\n")
        out_fname.write("TN_6: " +str(tn6))
```

13. false_positive_py

Αλγόριθμος για τον υπολογισμό του αριθμού False Positive για καθε μια απο τις κατηγορίες βιοδραστηριότητας.

```
# calculates false positive

#python false_positive.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
import sys
import math

if __name__ == "__main__":
    test_labels       = sys.argv[1]
    test_set_outputs1 = sys.argv[2]
    test_set_outputs2 = sys.argv[3]
    test_set_outputs3 = sys.argv[4]
    test_set_outputs4 = sys.argv[5]
    test_set_outputs5 = sys.argv[6]
    fp0 = 0
    fp1 = 0
    fp2 = 0
    fp3 = 0
    fp4 = 0
    fp5 = 0
    fp6 = 0
    out_fname=open("FPResults.txt",'w')
    #TP for the 1st test output
    with open(test_labels) as f0, open(test_set_outputs1) as f1:
        for line0, line1 in zip(f0, f1):
            if float(line0.rstrip()) != 0 and  float(line1.rstrip()) == 0:
                fp0 += 1
            elif float(line0.rstrip()) != 1 and  float(line1.rstrip()) ==
1:
                fp1 += 1
            elif float(line0.rstrip()) != 2 and  float(line1.rstrip()) ==
2:
                fp2 += 1
            elif float(line0.rstrip()) != 3 and  float(line1.rstrip()) ==
3:
                fp3+= 1
            elif float(line0.rstrip()) != 4 and  float(line1.rstrip()) ==
4:
                fp4 += 1
            elif float(line0.rstrip()) != 5 and  float(line1.rstrip()) ==
5:
                fp5 += 1
            elif float(line0.rstrip()) != 6 and  float(line1.rstrip()) ==
6:
                fp6 += 1
        out_fname.write("First output False Positive" + "\n")
        out_fname.write("FP_0: " +str(fp0) + "\n")
        out_fname.write("FP_1: " +str(fp1) + "\n")
        out_fname.write("FP_2: " +str(fp2) + "\n")
        out_fname.write("FP_3: " +str(fp3) + "\n")
        out_fname.write("FP_4: " +str(fp4) + "\n" )
        out_fname.write("FP_5: " +str(fp5) + "\n")
        out_fname.write("FP_6: " +str(fp6) + "\n")
    fp0 = 0
    fp1 = 0
    fp2 = 0
    fp3 = 0
    fp4 = 0
    fp5 = 0
    fp6 = 0
```

```
    #TP for the 2nd test output
    with open(test_labels) as f0, open(test_set_outputs2) as f2:
        for line0, line2 in zip(f0, f2):
            if float(line0.rstrip()) != 0 and  float(line2.rstrip()) == 0:
                fp0 += 1
            elif float(line0.rstrip()) != 1 and  float(line2.rstrip()) ==
1:
                fp1 += 1
            elif float(line0.rstrip()) != 2 and  float(line2.rstrip()) ==
2:
                fp2 += 1
            elif float(line0.rstrip()) != 3 and  float(line2.rstrip()) ==
3:
                fp3+= 1
            elif float(line0.rstrip()) != 4 and  float(line2.rstrip()) ==
4:
                fp4 += 1
            elif float(line0.rstrip()) != 5 and  float(line2.rstrip()) ==
5:
                fp5 += 1
            elif float(line0.rstrip()) != 6 and  float(line2.rstrip()) ==
6:
                fp6 += 1
        out_fname.write("Second output False Positive" + "\n")
        out_fname.write("FP_0: " +str(fp0) + "\n")
        out_fname.write("FP_1: " +str(fp1) + "\n")
        out_fname.write("FP_2: " +str(fp2) + "\n")
        out_fname.write("FP_3: " +str(fp3) + "\n")
        out_fname.write("FP_4: " +str(fp4) + "\n" )
        out_fname.write("FP_5: " +str(fp5) + "\n")
        out_fname.write("FP_6: " +str(fp6) + "\n")
    fp0 = 0
    fp1 = 0
    fp2 = 0
    fp3 = 0
    fp4 = 0
    fp5 = 0
    fp6 = 0
    #TP for the 3rd test output
    with open(test_labels) as f0, open(test_set_outputs3) as f3:
        for line0, line3 in zip(f0, f3):
            if float(line0.rstrip()) != 0 and  float(line3.rstrip()) == 0:
                fp0 += 1
            elif float(line0.rstrip())!= 1 and  float(line3.rstrip()) == 1:
                fp1 += 1
            elif float(line0.rstrip()) != 2 and  float(line3.rstrip()) ==
2:
                fp2 += 1
            elif float(line0.rstrip()) != 3 and  float(line3.rstrip()) ==
3:
                fp3+= 1
            elif float(line0.rstrip()) != 4 and  float(line3.rstrip()) ==
4:
                fp4 += 1
            elif float(line0.rstrip()) != 5 and  float(line3.rstrip()) ==
5:
                fp5 += 1
            elif float(line0.rstrip()) != 6 and  float(line3.rstrip()) ==
6:
                fp6 += 1
  out_fname.write("Third output False Positive" + "\n")
```

```
out_fname.write("FP_0: " +str(fp0) + "\n")
        out_fname.write("FP_1: " +str(fp1) + "\n")
        out_fname.write("FP_2: " +str(fp2) + "\n")
        out_fname.write("FP_3: " +str(fp3) + "\n")
        out_fname.write("FP_4: " +str(fp4) + "\n" )
        out_fname.write("FP_5: " +str(fp5) + "\n")
        out_fname.write("FP_6: " +str(fp6) + "\n")

    fp0 = 0
    fp1 = 0
    fp2 = 0
    fp3 = 0
    fp4 = 0
    fp5 = 0
    fp6 = 0

    #TP for the 4th test output
    with open(test_labels) as f0, open(test_set_outputs4) as f4:
        for line0, line4 in zip(f0, f4):
            if float(line0.rstrip()) != 0 and  float(line4.rstrip()) == 0:
                fp0 += 1
            elif float(line0.rstrip()) != 1 and  float(line4.rstrip()) ==
1:
                fp1 += 1
            elif float(line0.rstrip()) != 2 and  float(line4.rstrip()) ==
2:
                fp2 += 1
            elif float(line0.rstrip()) != 3 and  float(line4.rstrip()) ==
3:
                fp3+= 1
            elif float(line0.rstrip()) != 4 and  float(line4.rstrip()) ==
4:
                fp4 += 1
            elif float(line0.rstrip()) != 5 and  float(line4.rstrip()) ==
5:
                fp5 += 1
            elif float(line0.rstrip()) != 6 and  float(line4.rstrip()) ==
6:
                fp6 += 1

        out_fname.write("Fourth output False Positive" + "\n")
        out_fname.write("FP_0: " +str(fp0) + "\n")
        out_fname.write("FP_1: " +str(fp1) + "\n")
        out_fname.write("FP_2: " +str(fp2) + "\n")
        out_fname.write("FP_3: " +str(fp3) + "\n")
        out_fname.write("FP_4: " +str(fp4) + "\n" )
        out_fname.write("FP_5: " +str(fp5) + "\n")
        out_fname.write("FP_6: " +str(fp6) + "\n")

    fp0 = 0
    fp1 = 0
    fp2 = 0
    fp3 = 0
    fp4 = 0
    fp5 = 0
    fp6 = 0

    #TP for the 5th test output
    with open(test_labels) as f0, open(test_set_outputs5) as f5:
        for line0, line5 in zip(f0, f5):
            if float(line0.rstrip()) != 0 and  float(line5.rstrip()) == 0:
```

```
    fp0 += 1
            elif float(line0.rstrip()) != 1 and  float(line5.rstrip()) ==
1:
                fp1 += 1
            elif float(line0.rstrip()) != 2 and  float(line5.rstrip()) ==
2:
                fp2 += 1
            elif float(line0.rstrip()) != 3 and  float(line5.rstrip()) ==
3:
                fp3+= 1
            elif float(line0.rstrip()) != 4 and  float(line5.rstrip()) ==
4:
                fp4 += 1
            elif float(line0.rstrip()) != 5 and  float(line5.rstrip()) ==
5:
                fp5 += 1
            elif float(line0.rstrip()) != 6 and  float(line5.rstrip()) ==
6:
                fp6 += 1

        out_fname.write("Fifth output False Positive" + "\n")
        out_fname.write("FP_0: " +str(fp0) + "\n")
        out_fname.write("FP_1: " +str(fp1) + "\n")
        out_fname.write("FP_2: " +str(fp2) + "\n")
        out_fname.write("FP_3: " +str(fp3) + "\n")
        out_fname.write("FP_4: " +str(fp4) + "\n" )
        out_fname.write("FP_5: " +str(fp5) + "\n")
        out_fname.write("FP_6: " +str(fp6))
```

14. false_negative.py

Αλγόριθμος για τον υπολογισμό του αριθμού False Negative για καθε μια απο τις κατηγορίες βιοδραστηριότητας.

```
# calculates false negative

#python false_negative.py test_labels.txt test_set_outputs1.txt
test_set_outputs2.txt test_set_outputs3.txt test_set_outputs4.txt
test_set_outputs5.txt
import sys
import math

if __name__ == "__main__":
    test_labels       = sys.argv[1]
    test_set_outputs1 = sys.argv[2]
    test_set_outputs2 = sys.argv[3]
    test_set_outputs3 = sys.argv[4]
    test_set_outputs4 = sys.argv[5]
    test_set_outputs5 = sys.argv[6]
    fn0 = 0
    fn1 = 0
    fn2 = 0
    fn3 = 0
    fn4 = 0
    fn5 = 0
    fn6 = 0
    out_fname=open("FNResults.txt",'w')
    #TP for the 1st test output
    with open(test_labels) as f0, open(test_set_outputs1) as f1:
        for line0, line1 in zip(f0, f1):
            if float(line0.rstrip()) == 0 and  float(line1.rstrip()) != 0:
                fn0 += 1
            elif float(line0.rstrip()) == 1 and  float(line1.rstrip()) !=
1:
                fn1 += 1
            elif float(line0.rstrip()) == 2 and  float(line1.rstrip()) !=
2:
                fn2 += 1
            elif float(line0.rstrip()) == 3 and  float(line1.rstrip()) !=
3:
                fn3+= 1
            elif float(line0.rstrip()) == 4 and  float(line1.rstrip()) !=
4:
                fn4 += 1
            elif float(line0.rstrip()) == 5 and  float(line1.rstrip()) !=
5:
                fn5 += 1
            elif float(line0.rstrip()) == 6 and  float(line1.rstrip()) !=
6:
                fn6 += 1
        out_fname.write("First output False Negative" + "\n")
        out_fname.write("FN_0: " +str(fn0) + "\n")
        out_fname.write("FN_1: " +str(fn1) + "\n")
        out_fname.write("FN_2: " +str(fn2) + "\n")
        out_fname.write("FN_3: " +str(fn3) + "\n")
        out_fname.write("FN_4: " +str(fn4) + "\n")
        out_fname.write("FN_5: " +str(fn5) + "\n")
        out_fname.write("FN_6: " +str(fn6) + "\n")
    fn0 = 0
    fn1 = 0
    fn2 = 0
    fn3 = 0
    fn4 = 0
    fn5 = 0
    fn6 = 0
```

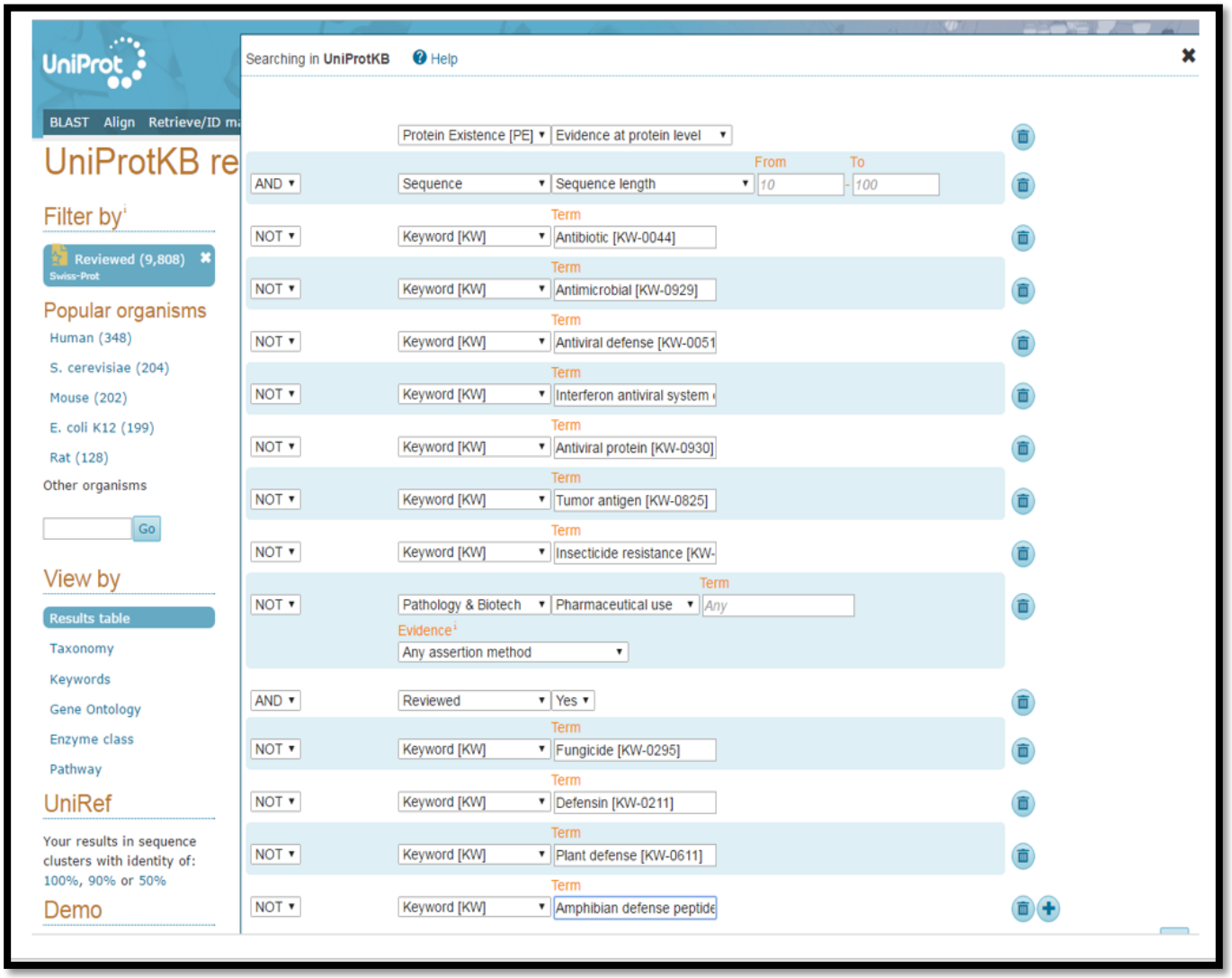
```
#TP for the 2nd test output
    with open(test_labels) as f0, open(test_set_outputs2) as f2:
        for line0, line2 in zip(f0, f2):
            if float(line0.rstrip()) == 0 and  float(line2.rstrip()) != 0:
                fn0 += 1
            elif float(line0.rstrip()) == 1 and  float(line2.rstrip()) !=
1:
                fn1 += 1
            elif float(line0.rstrip()) == 2 and  float(line2.rstrip()) !=
2:
                fn2 += 1
            elif float(line0.rstrip()) == 3 and  float(line2.rstrip()) !=
3:
                fn3+= 1
            elif float(line0.rstrip()) == 4 and  float(line2.rstrip()) !=
4:
                fn4 += 1
            elif float(line0.rstrip()) == 5 and  float(line2.rstrip()) !=
5:
                fn5 += 1
            elif float(line0.rstrip()) == 6 and  float(line2.rstrip()) !=
6:
                fn6 += 1
        out_fname.write("Second output False Negative" + "\n")
        out_fname.write("FN_0: " +str(fn0) + "\n")
        out_fname.write("FN_1: " +str(fn1) + "\n")
        out_fname.write("FN_2: " +str(fn2) + "\n")
        out_fname.write("FN_3: " +str(fn3) + "\n")
        out_fname.write("FN_4: " +str(fn4) + "\n")
        out_fname.write("FN_5: " +str(fn5) + "\n")
        out_fname.write("FN_6: " +str(fn6) + "\n")
    fn0 = 0
    fn1 = 0
    fn2 = 0
    fn3 = 0
    fn4 = 0
    fn5 = 0
    fn6 = 0
    #TP for the 3rd test output
    with open(test_labels) as f0, open(test_set_outputs3) as f3:
        for line0, line3 in zip(f0, f3):
            if float(line0.rstrip()) == 0 and  float(line3.rstrip()) != 0:
                fn0 += 1
            elif float(line0.rstrip()) == 1 and  float(line3.rstrip()) !=
1:
                fn1 += 1
            elif float(line0.rstrip()) == 2 and  float(line3.rstrip()) !=
2:
                fn2 += 1
            elif float(line0.rstrip()) == 3 and  float(line3.rstrip()) !=
3:
                fn3+= 1
            elif float(line0.rstrip()) == 4 and  float(line3.rstrip()) !=
4:
                fn4 += 1
            elif float(line0.rstrip()) == 5 and  float(line3.rstrip()) !=
5:
                fn5 += 1
elif float(line0.rstrip()) == 6 and  float(line3.rstrip()) != 6:
                fn6 += 1
```

```
out_fname.write("Third output False Negative" + "\n")
        out_fname.write("FN_0: " +str(fn0) + "\n")
        out_fname.write("FN_1: " +str(fn1) + "\n")
        out_fname.write("FN_2: " +str(fn2) + "\n")
        out_fname.write("FN_3: " +str(fn3) + "\n")
        out_fname.write("FN_4: " +str(fn4) + "\n")
        out_fname.write("FN_5: " +str(fn5) + "\n")
        out_fname.write("FN_6: " +str(fn6) + "\n")
    fn0 = 0
    fn1 = 0
    fn2 = 0
    fn3 = 0
    fn4 = 0
    fn5 = 0
    fn6 = 0
    #TP for the 4th test output
    with open(test_labels) as f0, open(test_set_outputs4) as f4:
        for line0, line4 in zip(f0, f4):
            if float(line0.rstrip()) == 0 and  float(line4.rstrip()) != 0:
                fn0 += 1
            elif float(line0.rstrip()) == 1 and  float(line4.rstrip()) !=
1:
                fn1 += 1
            elif float(line0.rstrip()) == 2 and  float(line4.rstrip()) !=
2:
                fn2 += 1
            elif float(line0.rstrip()) == 3 and  float(line4.rstrip()) !=
3:
                fn3+= 1
            elif float(line0.rstrip()) == 4 and  float(line4.rstrip()) !=
4:
                fn4 += 1
            elif float(line0.rstrip()) == 5 and  float(line4.rstrip()) !=
5:
                fn5 += 1
            elif float(line0.rstrip()) == 6 and  float(line4.rstrip()) !=
6:
                fn6 += 1
        out_fname.write("Fourth output False Negative" + "\n")
        out_fname.write("FN_0: " +str(fn0) + "\n")
        out_fname.write("FN_1: " +str(fn1) + "\n")
        out_fname.write("FN_2: " +str(fn2) + "\n")
        out_fname.write("FN_3: " +str(fn3) + "\n")
        out_fname.write("FN_4: " +str(fn4) + "\n")
        out_fname.write("FN_5: " +str(fn5) + "\n")
        out_fname.write("FN_6: " +str(fn6) + "\n")
    fn0 = 0
    fn1 = 0
    fn2 = 0
    fn3 = 0
    fn4 = 0
    fn5 = 0
    fn6 = 0
    #TP for the 5th test output
    with open(test_labels) as f0, open(test_set_outputs5) as f5:
        for line0, line5 in zip(f0, f5):
            if float(line0.rstrip()) == 0 and  float(line5.rstrip()) != 0:
                fn0 += 1
            elif float(line0.rstrip()) == 1 and  float(line5.rstrip()) !=
1:
                fn1 += 1
```

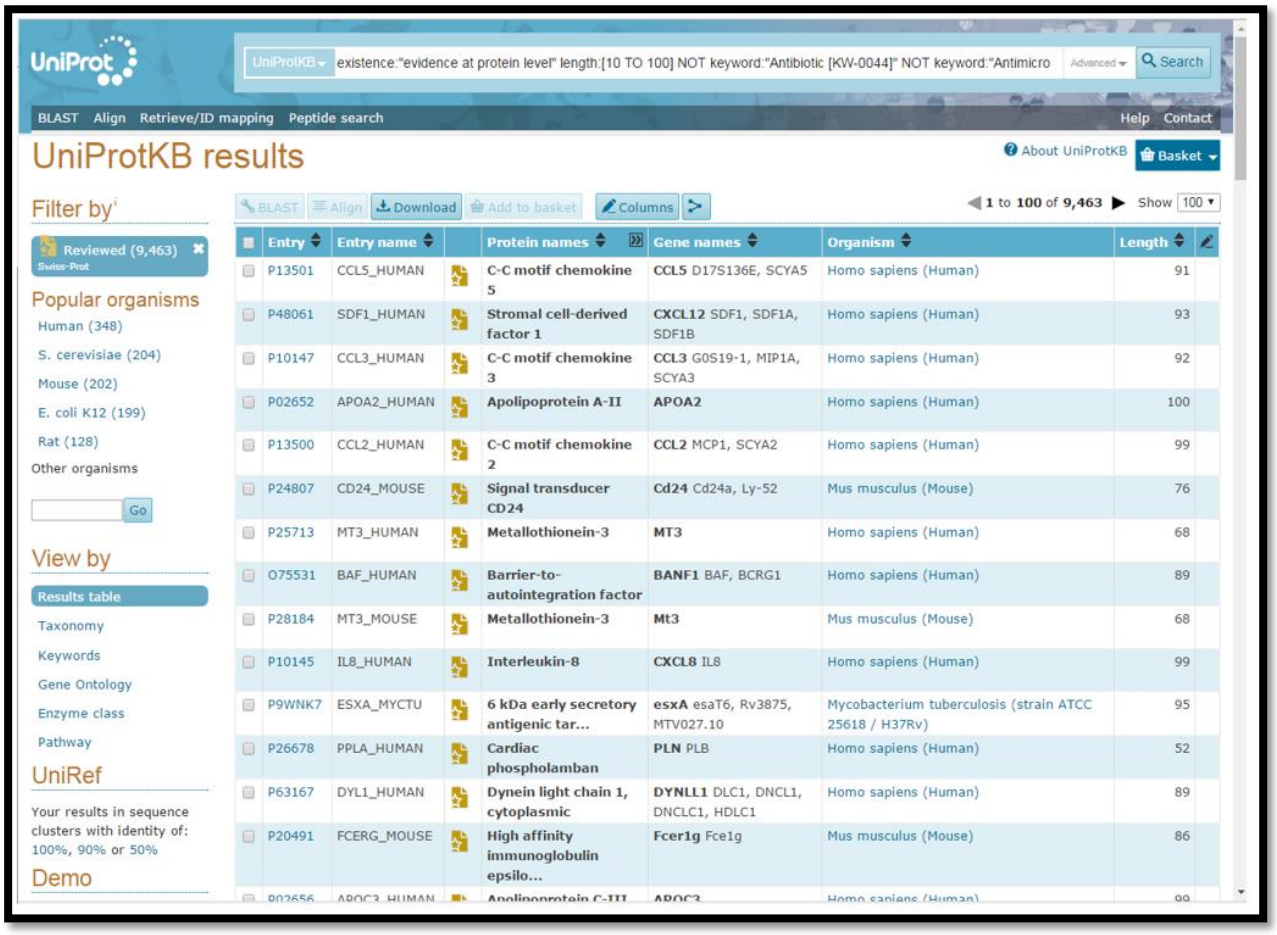
```
elif float(line0.rstrip()) == 2 and  float(line5.rstrip()) != 2:
                fn2 += 1
            elif float(line0.rstrip()) == 3 and  float(line5.rstrip()) !=
3:
                fn3+= 1
            elif float(line0.rstrip()) == 4 and  float(line5.rstrip()) !=
4:
                fn4 += 1
            elif float(line0.rstrip()) == 5 and  float(line5.rstrip()) !=
5:
                fn5 += 1
            elif float(line0.rstrip()) == 6 and  float(line5.rstrip()) !=
6:
                fn6 += 1

        out_fname.write("Fifth output False Negative" + "\n")
        out_fname.write("FN_0: " +str(fn0) + "\n")
        out_fname.write("FN_1: " +str(fn1) + "\n")
        out_fname.write("FN_2: " +str(fn2) + "\n")
        out_fname.write("FN_3: " +str(fn3) + "\n")
        out_fname.write("FN_4: " +str(fn4) + "\n")
        out_fname.write("FN_5: " +str(fn5) + "\n")
        out_fname.write("FN_6: " +str(fn6))
```

ΠΑΡΑΡΤΗΜΑ ΙΙ

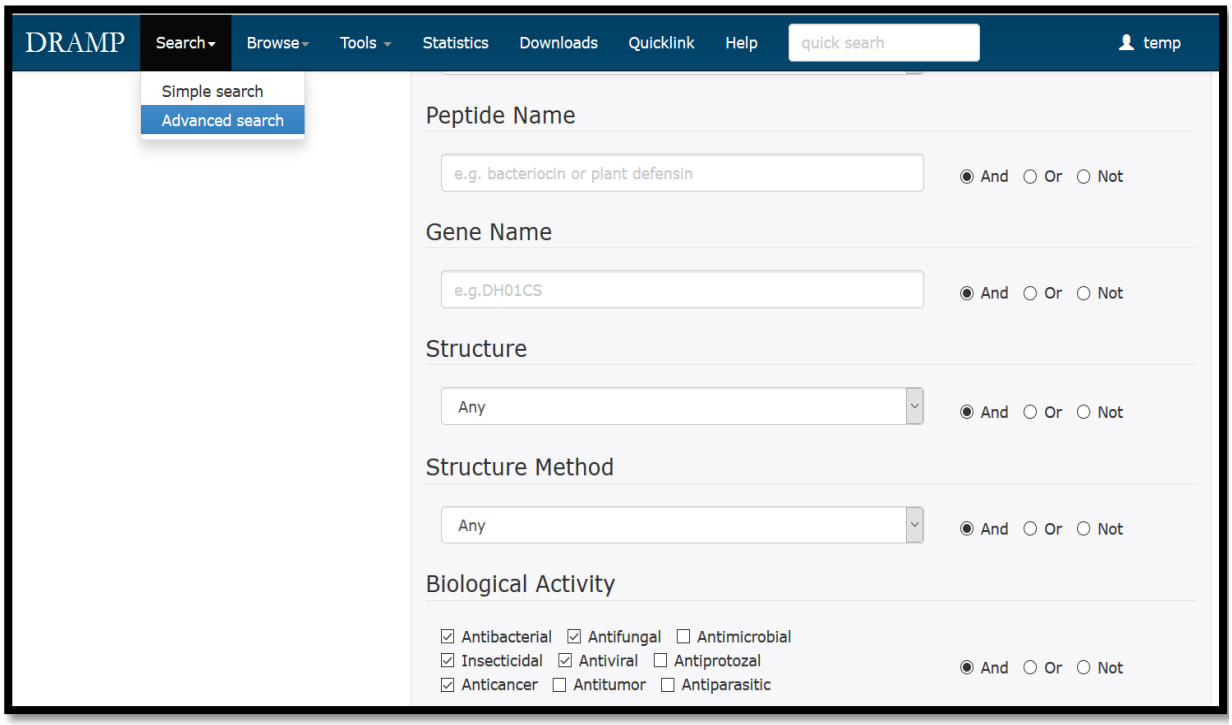
Η διαδικασία συλλογής αρνητικών και θετικών δειγμάτων εκπαίδευσης απεικονίζεται παρακάτω.



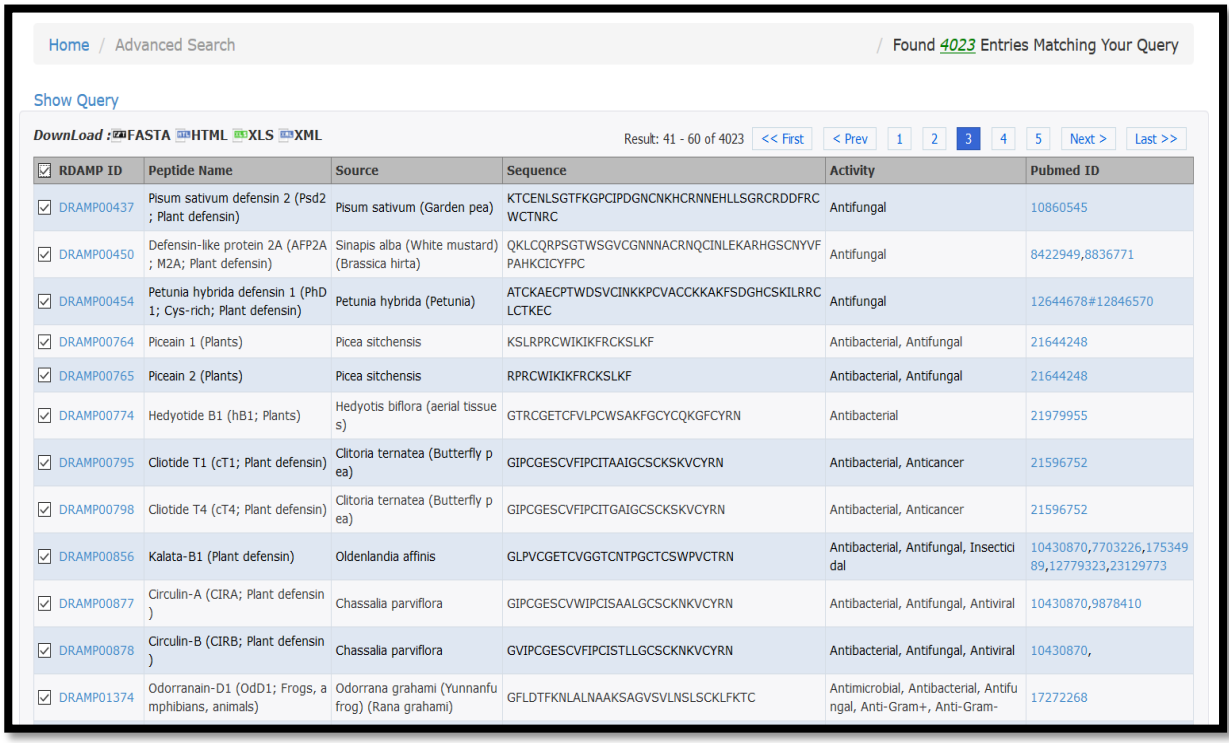
Κριτήρια επιλογής non-AMPs από Uniprot



Ενδεικτικά αποτελέσματα αναζήτησης στη Uniprot

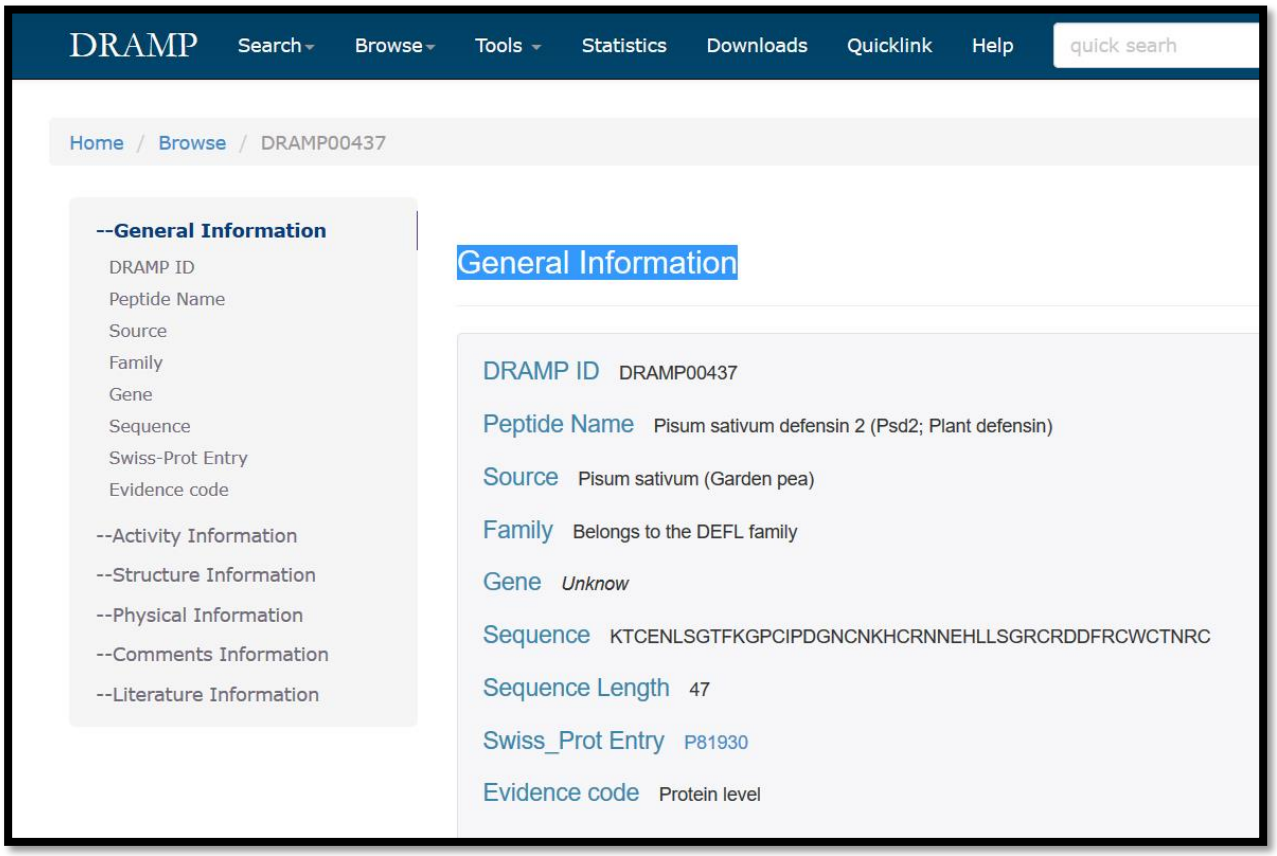


Κριτήρια επιλογής AMPs από DRAMP



Ενδεικτικά αποτελέσματα αναζήτησης στη DRAMP.

Επιλέγοντας το εικονίδιο FASTA γίνεται λήψη όλων των πεπτιδίων που εχουν επιλεγεί στην εμφανιζόμενη λίστα



Επιπλέον πληροφορία διαθέσιμη για κάθε πεπτίδιο που επιλέγεται. Το πεδίο Swiss_Prot Entry ανακατευθύνει στην αντίστοιχη εγγραφή στη Uniprot.

ΑΝΑΦΟΡΕΣ

[1]. Bahar, A. A., & Ren, D. (2013a). Antimicrobial Peptides. *Pharmaceuticals*, *6*(12), 1543–1575. https://doi.org/10.3390/ph6121543

[2]. Bahar, A. A., & Ren, D. (2013b). Antimicrobial Peptides. *Pharmaceuticals*, *6*(12), 1543–1575. https://doi.org/10.3390/ph6121543

[3]. Barbeta, B. L., Marshall, A. T., Gillon, A. D., Craik, D. J., & Anderson, M. A. (2008). Plant cyclotides disrupt epithelial cells in the midgut of lepidopteran larvae. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(4), 1221–1225. https://doi.org/10.1073/pnas.0710338104

[4]. Bastian, A., & Schäfer, H. (2001). Human alpha-defensin 1 (HNP-1) inhibits adenoviral infection in vitro. *Regulatory Peptides*, *101*(1–3), 157–161.

[5]. Ben Lagha, A., Haas, B., Gottschalk, M., & Grenier, D. (2017). Antimicrobial potential of bacteriocins in poultry and swine production. *Veterinary Research*, *48*. https://doi.org/10.1186/s13567-017-0425-6

[6]. Bevers, E. M., Comfurius, P., & Zwaal, R. F. (1996). Regulatory mechanisms in maintenance and modulation of transmembrane lipid asymmetry: pathophysiological implications. *Lupus*, *5*(5), 480–487. https://doi.org/10.1177/096120339600500531

[7]. Bhadra, P., Yan, J., Li, J., Fong, S., & Siu, S. W. I. (2018). AmPEP: Sequence-based prediction of antimicrobial peptides using distribution patterns of amino acid properties and random forest. *Scientific Reports*, *8*(1), 1697. https://doi.org/10.1038/s41598-018-19752-w

[8]. Boohaker, R. J., Lee, M. W., Vishnubhotla, P., Perez, J. M., & Khaled, A. R. (2012). The Use of Therapeutic Peptides to Target and to Kill Cancer Cells. *Current Medicinal Chemistry*, *19*(22), 3794–3804.

[9]. Brahmachary, M., Krishnan, S. P. T., Koh, J. L. Y., Khan, A. M., Seah, S. H., Tan, T. W., … Bajic, V. B. (2004). ANTIMIC: a database of antimicrobial sequences. *Nucleic Acids Research*, *32*(Database issue), D586–D589. https://doi.org/10.1093/nar/gkh032

[10]. Brogden, K. A. (2005). Antimicrobial peptides: pore formers or metabolic inhibitors in bacteria? *Nature Reviews. Microbiology*, *3*(3), 238–250. https://doi.org/10.1038/nrmicro1098

[11]. Castel, G., Chtéoui, M., Heyd, B., & Tordo, N. (2011). Phage display of combinatorial peptide libraries: application to antiviral research. *Molecules (Basel, Switzerland)*, *16*(5), 3499–3518. https://doi.org/10.3390/molecules16053499

[12]. Cavera, V. L., Arthur, T. D., Kashtanov, D., & Chikindas, M. L. (2015). Bacteriocins and their position in the next wave of conventional antibiotics. *International Journal of Antimicrobial Agents*, *46*(5), 494–501. https://doi.org/10.1016/j.ijantimicag.2015.07.011

[13]. Chang, K. Y., & Yang, J.-R. (2013). Analysis and Prediction of Highly Effective Antiviral Peptides Based on Random Forests. *PLOS ONE*, *8*(8), e70166. https://doi.org/10.1371/journal.pone.0070166

[14]. Corthésy, J., Theofilatos, K., Mavroudi, S., Macron, C., Cominetti, O., Remlawi, M., … Dayon, L. (2018). An Adaptive Pipeline To Maximize Isobaric Tagging Data in Large-Scale MS-Based Proteomics. *Journal of Proteome Research*, *17*(6), 2165–2173. https://doi.org/10.1021/acs.jproteome.8b00110

[15]. De Lima ME1, Figueiredo SG, Pimenta AM, Santos DM, Borges MH, Cordeiro MN, Richardson M, Oliveira LC, Stankiewicz M, Pelhate M. (n.d.). Peptides of arachnid venoms with insecticidal activity targeting sodium channels.

[16]. De Lucca, A. J., Bland, J. M., Jacks, T. J., Grimm, C., & Walsh, T. J. (1998). Fungicidal and binding properties of the natural peptides cecropin B and dermaseptin. *Medical Mycology*, *36*(5), 291–298.

[17]. De Lucca, Anthony J., & Walsh, T. J. (1999). Antifungal Peptides: Novel Therapeutic Compounds against Emerging Pathogens. *Antimicrobial Agents and Chemotherapy*, *43*(1), 1–11.

[18]. DURGESH K. SRIVASTAVA, & LEKHA BHAMBHU. (n.d.). DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE.

[19]. Fan, L., Sun, J., Zhou, M., Zhou, J., Lao, X., Zheng, H., & Xu, H. (2016). DRAMP: a comprehensive data repository of antimicrobial peptides. *Scientific Reports*, *6*. https://doi.org/10.1038/srep24482

[20]. Gaspar, D., Veiga, A. S., & Castanho, M. A. R. B. (2013). From antimicrobial to anticancer peptides. A review. *Frontiers in Microbiology*, *4*. https://doi.org/10.3389/fmicb.2013.00294

[21]. Gasteiger, E., Hoogland, C., Gattiker, A., Duvaud, S. 'everine, Wilkins, M. R., Appel, R. D., & Bairoch, A. (2005). Protein Identification and Analysis Tools on the ExPASy Server. In J. M. Walker (Ed.), *The Proteomics Protocols Handbook* (pp. 571–607). Totowa, NJ: Humana Press. https://doi.org/10.1385/1-59259-890-0:571

[22]. Gordon, Y. J., Romanowski, E. G., & McDermott, A. M. (2005). A Review of Antimicrobial Peptides and Their Therapeutic Potential as Anti-Infective Drugs. *Current Eye Research*, *30*(7), 505–515. https://doi.org/10.1080/02713680590968637

[23]. Gould, A., Ji, Y., Aboye, T. L., & Camarero, J. A. (2011). Cyclotides, a novel ultrastable polypeptide scaffold for drug discovery. *Current Pharmaceutical Design*, *17*(38), 4294–4307.

[24]. Hammami, R., Ben Hamida, J., Vergoten, G., & Fliss, I. (2009). PhytAMP: a database dedicated to antimicrobial plant peptides. *Nucleic Acids Research*, *37*(Database issue), D963–D968. https://doi.org/10.1093/nar/gkn655

[25]. Hammami, R., Zouhir, A., Le Lay, C., Ben Hamida, J., & Fliss, I. (2010). BACTIBASE second release: a database and tool platform for bacteriocin characterization. *BMC Microbiology*, *10*, 22. https://doi.org/10.1186/1471-2180-10-22

[26]. Hancock, R. E. W., & Chapple, D. S. (1999). Peptide Antibiotics. *Antimicrobial Agents and Chemotherapy*, *43*(6), 1317–1323.

[27]. Hancock, R. E. W., & Scott, M. G. (2000). The role of antimicrobial peptides in animal defenses. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(16), 8856–8861.

[28]. Horne, W. S., Wiethoff, C. M., Cui, C., Wilcoxen, K. M., Amorin, M., Ghadiri, M. R., & Nemerow, G. R. (2005). Antiviral Cyclic D,L-α-Peptides: Targeting a General Biochemical

Pathway in Virus Infections. *Bioorganic & Medicinal Chemistry*, *13*(17), 5145–5153. https://doi.org/10.1016/j.bmc.2005.05.051

[29]. Jenssen, H., Hamill, P., & Hancock, R. E. W. (2006). Peptide Antimicrobial Agents. *Clinical Microbiology Reviews*, *19*(3), 491–511. https://doi.org/10.1128/CMR.00056-05

[30]. Kanchiswamy, C. N., Takahashi, H., Quadro, S., Maffei, M. E., Bossi, S., Bertea, C., … Arimura, G. (2010). Regulation of Arabidopsis defense responses against Spodoptera littoralis by CPK-mediated calcium signaling. *BMC Plant Biology*, *10*, 97. https://doi.org/10.1186/1471-2229-10-97

[31]. Khamis, A. M., Essack, M., Gao, X., & Bajic, V. B. (2015). Distinct profiling of antimicrobial peptide families. *Bioinformatics*, *31*(6), 849–856. https://doi.org/10.1093/bioinformatics/btu738

[32]. Kohavi R. (1997). Artificial Intelligence.

[33]. Kościuczuk, E. M., Lisowski, P., Jarczak, J., Strzałkowska, N., Jóźwik, A., Horbańczuk, J., … Bagnicka, E. (2012). Cathelicidins: family of antimicrobial peptides. A review. *Molecular Biology Reports*, *39*(12), 10957–10970. https://doi.org/10.1007/s11033-012-1997-x

[34]. Langhammer, J., Česák, J., Langhammer, J., & Česák, J. (2016). Applicability of a Nu-Support Vector Regression Model for the Completion of Missing Data in Hydrological Time Series. *Water*, *8*(12), 560. https://doi.org/10.3390/w8120560

[35]. Lee, Y. T., Kim, D. H., Suh, J. Y., Chung, J. H., Lee, B. L., Lee, Y., & Choi, B. S. (1999). Structural characteristics of tenecin 3, an insect antifungal protein. *Biochemistry and Molecular Biology International*, *47*(3), 369–376.

[36]. Lehrer, R. I., Szklarek, D., Ganz, T., & Selsted, M. E. (1985). Correlation of binding of rabbit granulocyte peptides to Candida albicans with candidacidal activity. *Infection and Immunity*, *49*(1), 207–211.

[37]. Lobry, J. R., & Gautier, C. (1994). Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 Escherichia coli chromosome-encoded genes. *Nucleic Acids Research*, *22*(15), 3174–3180. https://doi.org/10.1093/nar/22.15.3174

[38]. Lodish, H., Berk, A., Zipursky, S. L., Matsudaira, P., Baltimore, D., & Darnell, J. (2000). Overview of the Secretory Pathway. *Molecular Cell Biology. 4th Edition*. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK21471/

[39]. Moerman, L., Bosteels, S., Noppe, W., Willems, J., Clynen, E., Schoofs, L., … Verdonck, F. (2002). Antibacterial and antifungal properties of alpha-helical, cationic peptides in the venom of scorpions from southern Africa. *European Journal of Biochemistry*, *269*(19), 4799–4810.

[40]. Ng, X. Y., Rosdi, B. A., & Shahrudin, S. (2015). Prediction of Antimicrobial Peptides Based on Sequence Alignment and Support Vector Machine-Pairwise Algorithm Utilizing LZ-Complexity [Research article]. https://doi.org/10.1155/2015/212715

[41]. Nielsen, H., Engelbrecht, J., Brunak, S., & von Heijne, G. (1997). Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering, Design and Selection*, *10*(1), 1–6. https://doi.org/10.1093/protein/10.1.1

[42]. Nielsen, Henrik. (2017). Predicting Secretory Proteins with SignalP. In D. Kihara (Ed.), *Protein Function Prediction: Methods and Protocols* (pp. 59–73). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4939-7015-5_6

[43]. Park, Y., Jang, S.-H., Lee, D. G., & Hahm, K.-S. (2004). Antinematodal effect of antimicrobial peptide, PMAP-23, isolated from porcine myeloid against Caenorhabditis elegans. *Journal of Peptide Science: An Official Publication of the European Peptide Society*, *10*(5), 304–311. https://doi.org/10.1002/psc.518

[44]. Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238. https://doi.org/10.1109/TPAMI.2005.159

[45]. Poth, A. G., Colgrave, M. L., Lyons, R. E., Daly, N. L., & Craik, D. J. (2011). Discovery of an unusual biosynthetic origin for circular proteins in legumes. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(25), 10127–10132. https://doi.org/10.1073/pnas.1103660108

[46]. Rapakoulia, T., Theofilatos, K., Kleftogiannis, D., Likothanasis, S., Tsakalidis, A., & Mavroudi, S. (2014). EnsembleGASVR: a novel ensemble method for classifying missense single nucleotide polymorphisms. *Bioinformatics (Oxford, England)*, *30*(16), 2324–2333. https://doi.org/10.1093/bioinformatics/btu297

[47]. Seshadri Sundararajan, V., Gabere, M. N., Pretorius, A., Adam, S., Christoffels, A., Lehväslaiho, M., … Bajic, V. B. (2012a). DAMPD: a manually curated antimicrobial peptide database. *Nucleic Acids Research*, *40*(Database issue), D1108–D1112. https://doi.org/10.1093/nar/gkr1063

[48]. Seshadri Sundararajan, V., Gabere, M. N., Pretorius, A., Adam, S., Christoffels, A., Lehväslaiho, M., … Bajic, V. B. (2012b). DAMPD: a manually curated antimicrobial peptide database. *Nucleic Acids Research*, *40*(Database issue), D1108–D1112. https://doi.org/10.1093/nar/gkr1063

[49]. Shai, Y. (2002). Mode of action of membrane active antimicrobial peptides. *Biopolymers*, *66*(4), 236–248. https://doi.org/10.1002/bip.10260

[50]. Terras, F. R., Schoofs, H. M., De Bolle, M. F., Van Leuven, F., Rees, S. B., Vanderleyden, J., … Broekaert, W. F. (1992). Analysis of two novel classes of plant antifungal proteins from radish (Raphanus sativus L.) seeds. *The Journal of Biological Chemistry*, *267*(22), 15301–15309.

[51]. Thomas, S., Karnik, S., Barai, R. S., Jayaraman, V. K., & Idicula-Thomas, S. (2010). CAMP: a useful resource for research on antimicrobial peptides. *Nucleic Acids Research*, *38*(Database issue), D774–D780. https://doi.org/10.1093/nar/gkp1021

[52]. Tom M. Mitchell. (1997). *Machine Learning*. McGraw-Hill Science/Engineer ing/Math; (March 1, 1997).

[53]. Torrent, M., Di Tommaso, P., Pulido, D., Nogués, M. V., Notredame, C., Boix, E., & Andreu, D. (2012). AMPA: an automated web server for prediction of protein antimicrobial regions. *Bioinformatics*, *28*(1), 130–131. https://doi.org/10.1093/bioinformatics/btr604

[54]. van der Weerden, N. L., Hancock, R. E. W., & Anderson, M. A. (2010). Permeabilization of Fungal Hyphae by the Plant Defensin NaD1 Occurs through a Cell Wall-dependent

Process. *The Journal of Biological Chemistry*, *285*(48), 37513–37520. https://doi.org/10.1074/jbc.M110.134882

[55]. Vihinen, M., Torkkila, E., & Riikonen, P. (1994). Accuracy of protein flexibility predictions. *Proteins: Structure, Function, and Bioinformatics*, *19*(2), 141–149. https://doi.org/10.1002/prot.340190207

[56]. Wang, G., Li, X., & Wang, Z. (2009). APD2: the updated antimicrobial peptide database and its application in peptide design. *Nucleic Acids Research*, *37*(Database issue), D933–D937. https://doi.org/10.1093/nar/gkn823

[57]. Wang, G., Li, X., & Wang, Z. (2016). APD3: the antimicrobial peptide database as a tool for research and education. *Nucleic Acids Research*, *44*(Database issue), D1087–D1093. https://doi.org/10.1093/nar/gkv1278

[58]. Wang, P., Hu, L., Liu, G., Jiang, N., Chen, X., Xu, J., … Chou, K.-C. (2011). Prediction of Antimicrobial Peptides Based on Sequence Alignment and Feature Selection Methods. *PLOS ONE*, *6*(4), e18476. https://doi.org/10.1371/journal.pone.0018476

[59]. Wang, Z., & Wang, G. (2004). APD: the Antimicrobial Peptide Database. *Nucleic Acids Research*, *32*(Database issue), D590–D592. https://doi.org/10.1093/nar/gkh025

[60]. Wolska, K. I., Grześ, K., & Kurek, A. (2012). Synergy between novel antimicrobials and conventional antibiotics or bacteriocins. *Polish Journal of Microbiology*, *61*(2), 95–104.

[61]. Zasloff, M. (1987). Magainins, a class of antimicrobial peptides from Xenopus skin: isolation, characterization of two active forms, and partial cDNA sequence of a precursor. *Proceedings of the National Academy of Sciences of the United States of America*, *84*(15), 5449–5453.

[62]. Zhang, L., Rozek, A., & Hancock, R. E. (2001). Interaction of cationic antimicrobial peptides with model membranes. *The Journal of Biological Chemistry*, *276*(38), 35714–35722. https://doi.org/10.1074/jbc.M104925200

[63]. Zhang, Ling-juan, & Gallo, R. L. (2016). Antimicrobial peptides. *Current Biology*, *26*(1), R14–R19. https://doi.org/10.1016/j.cub.2015.11.017