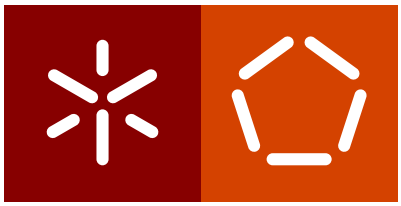
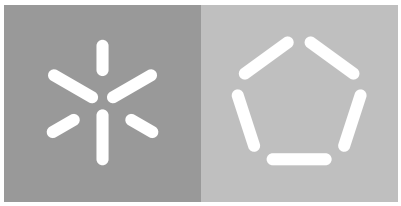


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ana Marta Fernandes Tavares Sequeira

Building an automated platform for the classification of peptides/proteins using machine learning

October 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ana Marta Fernandes Tavares Sequeira

Building an automated platform for the classification of peptides/proteins using machine learning

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Prof. Miguel Francisco Almeida Pereira Rocha
Dr. Diana Andreia Pereira Lousa

October 2019

ACKNOWLEDGEMENTS

Throughout the writing of this dissertation I have received a great deal of support and help from both professional and personal spheres, to which I would like to express my sincere acknowledgment.

First, I would like to thank my supervisors Dr. Miguel Rocha from Minho University and Dr. Diana Lousa from ITQB Nova for all the help and availability during this year. This thesis would not have been possible without their advises and expertise. I would also like to acknowledge all the colleagues from BisBII group, always willing to help me.

In addition, I want to thank to my closest family, who are always there for me, for their love, care and support.

My special thanks are extended to Bomboémia, a percussion group from the university, and my Judo crew for all the teachings, all the experiences and all the amazing people I had the chance to meet this past year. They contributed immensely for my personal growth.

Last, but not least, I wish to thank to my dearest friends that help me get throughout this difficult year and were always there, even from distant cities, giving support but also a happy distraction. Pedro Lourenço, Cláudia Passos, Ricardo Jorge, Armando Nava, Helena Freitas and Rita Ibañez, your support and true friendship was invaluable.

To all of you, here's my sincere (BIG) Thank You!

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

RESUMO

Um dos problemas mais desafiantes em bioinformática é a caracterização de sequências, estruturas e funções de proteínas. Propriedades físico-químicas e estruturais derivadas da sequêcia proteica têm sido utilizadas no desenvolvimento de modelos de *aprendizagem máquina (AM)*. No entanto, ferramentas para calcular estes atributos são escassas e têm limitações em termos de eficiência, facilidade de uso e capacidade de adaptação a diferentes problemas. Aqui, é descrita uma plataforma modular genérica e automatizada para a classificação de proteínas com base nas suas propriedades físico-químicas, que faz uso de diferentes algoritmos de AM. A ferramenta desenvolvida facilita as principais tarefas de AM e inclui módulos para ler e alterar sequências, calcular atributos de proteínas, realizar pré-processamento de dados, fazer redução e seleção de features, executar clustering, criar modelos de AM e fazer previsões. Como é construído de forma modular, o utilizador mantém o poder de alterar o código para atender às suas necessidades específicas. Esta plataforma foi testada com péptidos anticancerígenos e antimicrobianos e foi ainda utilizada para explorar péptidos de fusão virais. Os péptidos de fusão são uma classe de péptidos que interagem com a membrana, encontrados em vírus encapsulados e que são particularmente relevantes para a fusão da membrana do vírus com a membrana do hospedeiro. Determinar quais são as propriedades que os caracterizam é uma questão científica muito relevante, com importantes implicações tecnológicas.

Usando três conjuntos de dados diferentes compostos por sequências bem anotadas, quatro técnicas diferentes de extração de features e cinco métodos diferentes de seleção de features (num total de 24 conjuntos de dados testados), sete modelos de AM, com validação cruzada de 10 vezes e uma abordagem de pesquisa em grelha, foram treinados e testados. Os melhores modelos obtidos, com avaliações MCC entre 0,7 e 0,8 e precisão entre 0,85 e 0,9, foram utilizados para prever a localização de um péptido de fusão conhecido numa sequência da proteína de fusão do vírus do Dengue. Os modelos obtidos para prever a localização do péptido de fusão são úteis em pesquisas futuras, fornecendo também uma visão biológica das características físico-químicas distintivas dos mesmos.

Este trabalho apresenta uma ferramenta disponível gratuitamente para realizar a classificação de proteínas com AM e a primeira análise global de péptidos de fusão virais usando métodos baseados em AM, reforçando a usabilidade e a importância da AM em problemas de classificação de proteínas.

Palavras Chave: Aprendizagem máquina; Classificação de péptidos; Péptidos de fusão viral

ABSTRACT

One of the challenging problems in bioinformatics is to computationally characterize sequences, structures and functions of proteins. Sequence-derived structural and physicochemical properties of proteins have been used in the development of machine learning models in protein related problems. However, tools and platforms to calculate features and perform *Machine learning (ML)* with proteins are scarce and have their limitations in terms of effectiveness, user-friendliness and capacity.

Here, a generic modular automated platform for the classification of proteins based on their physicochemical properties using different ML algorithms is proposed. The tool developed, as a Python package, facilitates the major tasks of ML and includes modules to read and alter sequences, calculate protein features, preprocess datasets, execute feature reduction and selection, perform clustering, train and optimize ML models and make predictions. As it is modular, the user retains the power to alter the code to fit specific needs.

This platform was tested to predict membrane active anticancer and antimicrobial peptides and further used to explore viral fusion peptides. Membrane-interacting peptides play a crucial role in several biological processes. Fusion peptides are a subclass found in enveloped viruses, that are particularly relevant for membrane fusion. Determining what are the properties that characterize fusion peptides and distinguishing them from other proteins is a very relevant scientific question with important technological implications.

Using three different datasets composed by well annotated sequences, different feature extraction techniques and feature selection methods (resulting in a total of over 20 datasets), seven ML models were trained and tested, using cross validation for error estimation and grid search for model selection. The different models, feature sets and feature selection techniques were compared. The best models obtained for distinct metric were then used to predict the location of a known fusion peptide in a protein sequence from the Dengue virus. Feature importances were also analysed. The models obtained will be useful in future research, also providing a biological insight of the distinctive physicochemical characteristics of fusion peptides.

This work presents a freely available tool to perform ML-based protein classification and the first global analysis and prediction of viral fusion peptides using ML, reinforcing the usability and importance of ML in protein classification problems.

Keywords: Machine Learning; Peptide Classification; Viral Fusion Peptides

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A

AA aminoacid.

AAC Aminoacid Composition.

ACACS average chemical shifts.

ACP anticancer Peptides.

ALA alanine.

AM aprendizagem máquina.

AMP Antimicrobial Peptides.

ANN Artificial neural networks.

ANOVA analysis of variance.

APAAC Amphiphilic pseudo aminoacid composition.

ARG arginine.

ASN asparagine.

ASP aspartic acid.

ATC atomic composition.

AUC Area under curve.

B

BLOSUM Blocks of Amino Acid Substitution Matrix.

C

C Composition.

CL cardiolipin.

CTD Composition, Transisition, Distribution.

CTRIAD Conjoint triad.

D

D Distribution.

DA discriminant analysis.

DI Informatics Department.

DPC Dipeptide Composition.

E

EBOV Ebola virus.

F

FDR False discovery rate.

FI features importances.

FN false negative.

FP false positive.

FPEP fusion peptide.

G

GB Gradient Boosting classifier.

GLN glutamine.

GLU glutamic acid.

GLY glycine.

GNB gaussian naive bayes.

GRAVY grand average of hydropathy.

GRNN Generalized regression neural network.

H

HC hierarchical clustering.

HIS histidine.

HIV human immunodeficiency virus.

HMM Hidden markov models.

I

IFV Influenza virus.

ILE isoleucine.

K

KNN k-nearest neighbour algorithms.

L

LDA Linear discriminant analysis.

LEU leucine.

LPS lipopolysaccharide.

LYS lysine.

M

MCC Matthews correlation coefficient.

MET methionine.

ML Machine learning.

N

NGC negative gaussian curvature.

P

PAAC Pseudo aminoacid composition.

PC Principal component.

PCA Principal component analysis.

PECC Percentage of Examples Correctly Classified.

PG phosphatidylglycerol.

PHE phenylalanine.

PNN Probabilistic Neutral network.

PPI Protein-Protein Interactions.

PRO proline.

PS phosphatidylserine.

PSSM position-specific weight matrix.

Q

QSO Quasi Sequence order descriptors.

R

RAAC Reduced amino acid composition.

RBF radial basis function.

RF Random Forest.

RFE recursive feature elimination.

RMSE Root mean squared error.

ROC receiver operating characteristic.

S

SAAC Split amino acid composition.

SER serine.

SGD stochastic gradient descendent.

SSE sum of squared errors.

SVC Support Vector Classification.

SVM support vector machines.

T

T Transition.

TMD Transmembrane domain.

TN true negative.

TP true positive.

TPC Tripeptide composition.

U

UM University of Minho.

V

VAL valine.

W

WHO World Health Organization.

1

INTRODUCTION

This dissertation describes the Master's work developed in the context of the Bioinformatics master held at *Informatics Department (DI)*, *University of Minho (UM)*.

1.1 MOTIVATION

Membrane-interacting peptides play a crucial role in several biological processes, such as viral fusion, attaching proteins to membrane bilayers, inactivating bacteria (antimicrobial peptides) or disrupting cancer cells (anticancer peptides) [1]. These peptides have some common characteristics, such as a high hydrophobic content, nevertheless their specific properties can vary according to their biological role. Determining what are the properties that characterize different classes of membrane-interacting peptides and distinguishing them is a very relevant scientific question with important technological implications. Classifying these peptides according to their specific function can have very important applications, such as the identification of novel fusion peptides or novel antimicrobial/anticancer peptides.

Fusion peptides are an important class of membrane-interacting peptides, found in enveloped viruses, such as influenza, Dengue and HIV. These viruses have proteins whose function is to fuse their membrane with the host membrane – fusion proteins. A segment of these proteins known as the *fusion peptide (FPep)*, which inserts into the host membrane, is particularly relevant for membrane fusion. The FPeps from different virus families are quite diverse at the sequence and structural level, although they have some common features [2]. The host team is currently trying to find the specific patterns that characterize viral FPeps and enabling the identification of novel ones. The team has previously collected a large amount of data on viral fusion peptides and proteins and performed a preliminary machine learning-based classification of these peptides, with promising results.

Sequence-derived structural and physiochemical properties of peptides/proteins (e.g. length, charge hydrophobicity, aminoacid composition, dipeptide composition) have been used in the development of machine learning models to predict, among others, protein structural and functional classes, protein-protein or protein-ligand interactions and peptides with specific properties [3], as it is the case with the problems mentioned above. Indeed,

this type of characterization can be useful in a variety of areas such as chemogenomics, predicting drug target interaction [4] or antimicrobial activity [5]. This type of approach assumes that the function/activity of a peptide (e.g. antimicrobial activity) may be predicted from its physicochemical properties that can be computed from the knowledge of only the peptide sequence [6]. After obtaining the subset of descriptors that is most relevant for the prediction of the property of interest, it is then possible to use supervised machine learning algorithms to assess and predict the property of interest in different protein sequences.

Machine learning algorithms have been used in the characterization and identification of different types of peptides. Regarding the mentioned case studies, in what concerns anticancer and antimicrobial peptides there are already tools for their identification [7, 8, 9] and design [10, 11, 12, 13]. For fusion peptides, as far as we know, there is no systematic and global analysis of FPep sequences from different viral families using machine learning algorithm.

In the context of machine learning methods to analyse and classify proteins based on sequence-derived properties, general tools have been developed. An example is the web server Bio-seq analysis, a general platform that uses machine learning to analyse DNA, RNA and protein sequences [14]. However, there is no tool specifically focused in analysing and/or distinguishing membrane interacting peptides (such as anticancer peptides and viral fusion peptides). Moreover, in the existing platforms the user does not have full control over all the steps of the protocol and cannot adapt the code to its specific problem.

1.2 OBJETIVES

The aim of this project is to build a generic automated platform for the classification of peptides/proteins based on their physicochemical properties that can make use of different machine learning algorithms. This platform will be built in a modular way, allowing users to have control over the different steps and adapting/extending the code to fit their specific needs. During this thesis, we will focus on membrane-interacting peptides as test cases to evaluate our platform.

The pipeline will receive as inputs a set of protein/peptide sequences and calculate a large number of physicochemical descriptors. The pipeline will then evaluate different machine learning algorithms, accounting for feature selection, hyperparameter optimization and error estimation for the different models, aiming to select models with high performance and to calculate variable importance. This platform will be applied to two different test cases: classification of viral fusion peptides and classification of anticancer and antimicrobial peptides.

In detail, the work will address the following scientific/technological objectives:

- Review relevant literature for peptide/protein classification, machine learning methods and applications in related scenarios;
- Collect and curate train and test datasets used to construct and evaluate the model (definition of positive and negative datasets for training and testing);
- Develop a ML-based general-purpose pipeline for peptide classification, including choice and calculation of physicochemical descriptors; selection and filtering of relevant physicochemical descriptors; evaluation and selection of different machine learning algorithms;
- Validate the platform with anticancer and antimicrobial test studies;
- Apply the developed pipeline to identify novel fusion peptides from the sequence of viral fusion proteins with unknown fusion peptides;
- Analyse the distinctive features of FPep and their biological significance.

1.3 STRUCTURE OF THE TEXT

In chapter 2, an introduction to membrane active peptides will be presented. The definition, structure and mechanism of action will be explained for antimicrobial peptides, anticancer peptides and viral fusion peptides, which are the test-case peptides in this scientific work.

Chapter 3 will deliver an explanation of the concepts on ML, namely on unsupervised and supervised machine learning algorithms, supervised ML workflow and key concepts, such as cross validation, feature selection, and over and underfitting. Relevant ML packages in python will also be addressed.

Chapter 4 will focus on the application of ML on peptides and description of the relevant previous work existing in peptide classification using ML approaches.

In chapter 5 the description of the code developed will be presented. The structure of the package, outcomes and possible improvements will be presented.

The developed package is tested against anticancer and antimicrobial case studies. The results and discussion of this validation will be presented on chapter 6.

Chapter 7 will present the viral fusion peptide case study. The methods and results will be reported followed by a discussion of the main results.

Finally, chapter 8 will highlight the main conclusions of the thesis followed by a description of possible improvements and future work.

2

MEMBRANE-ACTIVE PEPTIDES

Membrane-interacting peptides are characterized for having a high membrane affinity and in most cases an effect on the membranes they interact with. They are ubiquitous in nature and participate in several biological processes such as viral fusion, attachment of proteins to membrane bilayers, inactivation of bacteria or disruption of cancer cells [1].

Membrane-active peptides constitute a specific type of membrane-interacting peptides that have an active role in the membrane. They can disrupt the membrane leading to cell lysis, translocate through it to deliver cargos into the cell or promote membrane fusion [15]. These peptides have common characteristics like high hydrophobicity and,, in some cases, the ability to generate a *negative gaussian curvature (NGC)* in model membranes (membrane curvature required for common membrane permeation such as pore formation, blobbing and budding) [6, 15]. They have, nonetheless, specific properties according to their specific biological role.

Nowadays, two of the major global health challenges, signalled by the *World Health Organization (WHO)*, are the antibiotic resistant bacteria and cancer. Both microbial infection and cancer are leading causes of morbidity and mortality worldwide and urgent solutions and targeted approaches are needed [16]. Another major health concern are viral infections caused by viruses such as *human immunodeficiency virus (HIV)*, ZIKA, *Ebola virus (EBOV)*, *Influenza virus (IFV)*, or Hepatitis virus [17].

Peptides have been in the front-line of the development and design of effective and cell specific therapeutics [16]. This work will focus on the study of membrane-active peptides, specifically peptides with antimicrobial activity (antimicrobial peptides), peptides with anticancer activity (anticancer peptides) and peptides essential for the infection of viruses (viral fusion peptides). A small summary of some of some of the specific properties and characteristics of antimicrobial peptides, anticancer peptides and viral fusion peptides is described in the following sub sections.

2.1 ANTIMICROBIAL PEPTIDES

Antimicrobial Peptides (AMP) are compounds widely distributed in nature that can kill microbial pathogens directly or indirectly by modulating host defence systems [18]. AMPs are produced by all life forms of organisms, from bacteria, fungi and plants to animals, existing also synthetic variants. Bacteria produce AMPs to kill other bacteria competing for the same ecological niche, while in higher organisms they are components of the innate immunity. These small and generally amphipathic molecules, which usually contain cationic and hydrophobic residues in elevated proportion [1], show a broad spectrum of antimicrobial activities and the majority can kill both gram positive and gram negative bacteria. Besides that, a significant number have been shown to have anticancer and antiviral activities, some are known to have *lipopolysaccharide (LPS)* neutralizing ability and others have the capacity to modulate the immune system [18, 19]. Examples of AMPs include defensin, dermcidin, LL-37 (human origin), cecropin (insect origin) and magainin (amphibian origin) [15].

The rapidly increasing resistance toward conventional antibiotics is a growing problem all over the world [16]. Most AMPs interact with the pathogen membrane in a reduced time frame and act through nonspecific interactions with membrane lipids, decreasing the probability of developing resistance [1]. Consequently, AMPs have captured attention as novel drug candidates and for the development of novel therapies, with several being already evaluated in clinical trials [18]. With the grown interest and focus on AMPs, AMPs databases, have increased. An example is the antimicrobial peptide database (APD3), which contained records for 3051 antimicrobial peptides on January 2019, in categories such as antibacterial, antiviral, antifungal or anticancer [20], with some peptides overlapping in some categories as they can exhibit dual properties [21].

2.1.1 *Structure of antimicrobial peptides*

AMPs display remarkable structural and functional diversity and a great variation in their secondary structures has been reported [19]. They are generally short, commonly consisting of 10-50 *aminoacid (aa)* residues (optimal length of 21-30) [21], net positively charged (cationic), amphipathic and with a substantial fraction of hydrophobic residues (30% or more). Lysine and arginine residues are frequently present [18, 19, 22] and are responsible for the positive net charge interacting with the negatively charged phosphate groups on membranes of bacteria [15]. Furthermore, glycine (neutral) and leucine (hydrophobic) are predominant aa residues as well [21].

AMPs are commonly classified based on their secondary structure into α-helical, β-sheet, or peptides with extended/random-coil structure (cysteine rich), with most AMPs belonging to the first two categories [1, 18, 19]. The α-helical peptides, like melittin, dermcidin and

LL-37 [15] are often unstructured in aqueous solution, but adopt an amphipathic helical structure in contact with a biological membrane [18]. β-sheet peptides, like defensins [18], are stabilized by disulphide bonds and, due to their rigid structure, are more ordered in aqueous solution and do not undergo a drastic conformational change as helical peptides upon membrane interaction. Furthermore, a peptide can adopt different conformations depending on its concentration an the specific membrane composition [15].

Decreasing of the peptide length is expected to decrease binding to membranes and the tendency to form ordered secondary structures resulting in a decreased adsorption driving force. Adsorption and destabilization of membranes increase with peptide positive charge, hydrophobicity and at pH<pKa (where the peptide is fully positively charged). Additionally, amphiphilic conformations like α-helices,with an hydrophilic and an hydrophobic face, contribute to peptide binding, membrane disruption, and bacterial killing. Moreover, AMP anti-inflammatory effects are linked to LPS and lipid-A binding and activation of the inflammatory cascade through the nuclear transcription factor NF-KB [22].

2.1.2 *Mechanism of action of antimicrobial peptides*

Unlike the majority of available antibiotics that interact with a specific target protein, AMPs primary mode of action is the disruption of cell membrane of the microorganisms leading to cell lysis and death [18, 19, 22]. The cationic charge of AMPs leads to their accumulation next to negatively charged surfaces like the gram negative outer membranes (wich have lipids such as *phosphatidylglycerol (PG)*, *cardiolipin (CL)*, *phosphatidylserine (PS)* and LPS) or gram positive cell wall [19, 23].

In addition, by charge-exchange mechanisms, they can translocate across the membrane into the cytoplasm and across pores in the cell wall of gram positive bacteria, acting on intracellular targets including DNA, RNA, and interfering with protein synthesis [18, 19, 22, 23]. Besides that, as the peptides undergo changes in conformation and aggregation state in the presence of membranes, a single peptide can act through several mechanisms depending on the peptide's structure, the peptide:lipid ratio, and the properties of the lipid membrane [23].

Mammalian cell membranes consist largely of neutral charge phospholipids and are therefore less attractive to cationic AMPs. In addition, cholesterol present in mammalian membranes makes it harder for AMPs to disrupt lipid bilayer structures. This way, AMPs are selectively toxic to bacteria [19].

The interaction of AMP with membrane depends on the physicochemical properties such as the aminoacid sequence, peptide length, net charge, amphipathicity, hydrophobicity, structural folding in membranes (secondary structure, dynamics and orientation), oligomerization, peptide concentration and membrane composition [1, 19, 22]. There are several

mechanisms of membrane-disruption proposed to explain the activityAMPs. The models that have receive most attention in the field includes pore-forming models as the barrel-stave model and toroidal-pore wormhole and nonpore-forming models such as the carpet model that describes a mechanism of membrane dissolution and detergent type membrane lytic mechanism [1, 15, 19].

Some AMPs have shown antiviral activities, interacting directly with the envelope of the virus, leading to a pore formation and consequent lysis of the viral particle [19], capacity of modulation of inflammation and neutralization of pathogenic toxins [18, 22]. In addition, some AMPs (cationic AMPs), have anticancer properties which may be due to both bacteria and cancer cell membranes having negative charge [19]. The properties and mode of action of anticancer peptides are described in the section below.

The development of AMPs as therapeutic agents has grown interest due to the high and rapid bactericidal activity and broad range of action (important in polymicrobial infections). AMPS are a class of therapeutic agents, which are complementary to existing antibiotics, to which bacteria develop less resistance, being on the frontline as an alternative to solve the growing problem of resistance to conventional antibiotics [18, 19, 23]. Furthermore, altering specific residues can significantly impact on their biological activity [16, 22].Few AMPs are already approved for clinical use, numerous are nowadays under clinical development [18, 19]. For all the reasons above, computational methods that can quickly and accurately identify candidate peptides as AMPs for subsequent experimental assays are necessary to shorten the drug discovery process [24].

2.2 ANTICANCER PEPTIDES

Cancer is one of the major causes of death and affects millions of people. This condition is caused by the uncontrolled growth and spreading of abnormal cells that have the ability to rapidly spread or invade other parts of the body [1]. It is not a single disease but rather a heterogeneous group of several complex diseases, which makes the development of anticancer therapies difficult [12]. Despite the advances in cancer therapy, techniques used today, such as chemotherapy and surgery, do not have high success rates in some cancer types and cancer cells can develop mechanisms of resistance. Consequently, there is interest in developing anticancer agents with a new mode of action, selective and more efficient [1, 19]. In this context, a growing number of studies has indicated cationic antimicrobial peptides as exhibiting a broad spectrum of cytotoxic activity against cancer cells. These *anticancer Peptides (ACP)*s are considered a resourceful therapeutic strategy, that can be either used isolated or in combination with other conventional drugs [1, 19].

ACPs display a broad spectrum of modes of action, not yet fully understood, including the ability to kill cancer cells, interfere with cancer cells by causing apoptosis mediated

via mitochondrial disruption, trigger necrosis via cell lysis, stimulate the immune system of the host, prevent tumour angiogenesis and metastasis [21] or destroy primary tumours [1, 12]. They are expected to be selective towards cancer cells without damaging normal cells or vital organs and display a variety of modes of action, which are different across cancer types [1, 12]. Despite some drawbacks, as low in vivo stability and high costs for production, the fact that they do not impair the normal body physiological functions, makes them promising therapeutics for cancer treatment [9]. Besides that, since ACPs exhibit short time-frame of interaction, they are less probable to cause resistance than other drugs. They also present good solubility as well as good tumour penetration [21]. As it is quite expensive and time-consuming to identify anticancer peptides using experimental methods, it is necessary to develop sequence based computational methods capable of determining ACP candidates, accelerating the process of discovery and design of ACP [7, 12].

The mechanism underlying each membranolytic peptide activity is dependent on the ACP characteristics and the target membrane. Cancer and normal mammalian cells have several differences that are accounted responsible for the selectivity of some of the ACPs. Normal cells usually have an overall neutral charge, whereas cancer cell membranes typically carry a net negative charge due to a higher than normal expression of anionic molecules such as PS (<9% of the total phospholipids of membranes) and O-glycosylated mucins. As it occurs with AMPs, the electrostatic attraction between the negatively charged components of cancer cells and the positively charged ACPs is believed to play a major role in the strong binding and selective disruption of cancer cell membranes. Besides having a more positive charge, cancer cell membranes present other differences, such as higher fluidity and lower amounts of cholesterol, which makes the membrane more fluid and may allow cancer cells to bind increased numbers of ACP molecules. These alterations could possibly affect receptor accessibility, cell adhesion, and other types of interaction between the cancer cell and its environment, and could also play a role in the selective binding of ACPs to the cancer cells [1, 19].

ACPs include Aurein 1.2 (isolated from a frog species) which has been described to have both antimicrobial activity and anticancer activity without significant cytotoxicity and defensins like the human neutrophil peptide-1 , 2 and 3 (HNP-1, HNP-2, HNP-3) [21]. Being a subset of antimicrobial peptides, the characteristics of ACPs are very similar to those of other AMPs. However, the physicochemical properties that drive some AMPs to possess anticancer activity, while others do not, is still unclear and more research is needed to understand these differences and help drive specific designs of ACPs [21]. Although AMPs and ACPs share similar molecular properties, not all AMPs are ACPs and therefore it is important to understand the factors that allow ACPs to recognize and lyse neoplastic cells, unravelling the specific targets that are expressed and presented within a certain tumour type [1].

2.2.1 *Properties of anticancer peptides*

In terms of structure, ACPs are mainly categorized as adopting either an α-helix or β-sheet conformation [1]. ACPs are short peptides containing between 5-30 aa residues and exhibit cationic amphipathic structures [7, 12, 21]. ACPs most often consist of positively charged, aromatic, and hydrophobic residues. Furthermore, a significant difference in residue preference between ACPs and non-ACPs has been observed [12]. In a study performed in 2013, Tyagi et al used computational methods to differentiate anticancer peptides from general sequences and from AMPs that do not present anticancer activity. They concluded that anticancer peptides, when compared to random sequences, had overall, more cysteine, glycine, isoleucine, lysine and tryptophan residues. However, considering a comparison between ACPs and AMPs who are not described to displayed anticancer activities, AMPs have more alanine, glycine, lysine and leucine residues than ACPs and less phenylalanine, proline and tryptophan residues. Furthermore, taking in consideration only the C and N terminal residues (terminal residues play crucial roles in biological functions), ACPs have shown to have a higher proportion of cysteine (and less glycine) in N-terminal and higher proportion of tyrosin and tryptophan in C-terminal when comparing to other AMPs without anticancer properties [25].

2.3 VIRAL FUSION PEPTIDES

Membrane fusion can be defined as the merging of two separate and apposed lipid bilayers into a single bilayer resulting in the mix of two initially distinct aqueous compartments. It is an important and ubiquitous process to the cell life [26, 27]. There are two types of biological membrane fusion with different mechanisms associated: endoplasmatic fusion, i.e., fusion inside the cells, and exoplasmatic membrane fusion, fusion that starts with merging monolayers topologically corresponding to the outer monolayers of plasma membranes. Here, the focus is in the exoplasmic fusion that can be observed in virus-host cell membrane fusion [28].

One of the key aspects of the infection by virus having lipid bilayer envelopes is the fusion of the viral membrane with a membrane of the target cell [29]. Lipid enveloped pathogenic viruses such as IFV, HIV or EBOV, use membrane glycoproteins to induce membrane fusion and enter selected host cell, gaining access to internal components of the cell in a process called 'viral entry'. Some fuse with the plasma membrane at cell surface and others fuse with the endosome after endocytosis by host cells [26, 27].

Present in the viral envelope, fusion proteins are a class of proteins indispensable to the process of membrane fusion of the virus with host cell [27]. Although they can vary in

structure, its mechanism of action is common: triggered by the binding of ligand, they suffer a conformational change that is coupled to apposition and merging of the two bilayers [29].

Fusion peptides are segments of viral fusion proteins, with approximately 20-25 aa residues frequently found at the N (amino) terminus [2, 27]. They are moderately hydrophobic segments of membrane fusion proteins that enable these proteins to disrupt and connect two apposed biological membranes [26], being of viral (viral FPeps) or non-viral origin [30]. FPeps have been proposed to reduce binding energy, fill the void space between bilayers (the space is energetically unfavourable), alter curvature of the membrane (promote negative curvature) and induce depth dependent membrane ordering [2].

Viruses have very dynamic genomes and undergo many replication cycles during infection and because of this, viruses and theirs viral entry machineries, are under an intense evolutionary pressure, having high mutation rates [26]. However, the sequences of fusion peptides are highly conserved within different groups of fusion proteins within the same virus family, but not between different viral families [30], which hinders the discovery of the unique features that characterize viral FPeps. Understanding the common properties present in the FPeps of all enveloped virus is crucial to understand the general mechanism of protein mediated fusion. The conserved elements of the viral entry mechanism provide clinical targets for the development of inhibitors (antivirals) and immunogens (vaccines) [26].

2.3.1 *Mechanisms of viral fusion proteins*

For fusion to occur, the fusing bilayers must come into close apposition. To overcome the kinetic barrier necessary for lipid bilayer fusion, viruses expose on their surface copies of viral fusion proteins [29].

Viral fusion proteins are indispensable to this process as they have two membrane – interacting elements: A C-terminal transmembrane anchor to hold the protein in the viral membrane and the fusion peptide that interacts with the target membrane. They are held in a 'prefusion conformation' until the fusogenic conformational transition occurs [29].

The fusion protein is stimulated by a signal associated with arrival at the cell to be infected (for e.g., binding of a ligand like protons or receptor) and undergoes a conformational change (liberating free energy and lowering the kinetic barrier to draw membranes together) that favours the contact between membrane and fusion peptide and initiates the process of membrane fusion. Then, the two proximal leaflets of the lipid bilayers, but not the two distal ones, merge in a stalk structure, allowing the mixing of the lipids of the monolayers in contact (hemifusion intermediate). Increased tension leads to the formation of a hydrophilic pore and pore enlargement. This fusion pore connects the internal aqueous volumes enclosed by the originally separated membranes [26, 28, 29].

In Figure 1 a schematic representation of the events in membrane fusion promoted by a viral fusion protein describe as in [29] is represented.

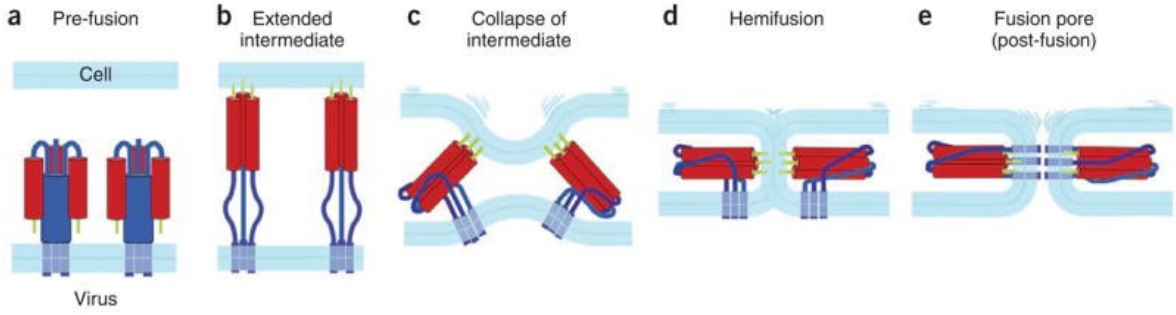


Figure 1: Schematic representation of the sequence of events in membrane fusion promoted by a viral fusion peptide. a) Protein in pre-fusion conformation, with fusion peptide (green) sequestered b) extended intermediate, the protein opens up, extending the fusion peptide to interact with cell lipid bilayer. The part of protein bearing the fusion peptide forms a trimer cluster. c) Collapse of the extended intermediate, a C-terminal segment of the protein folds back along the outside of the trimer core. The segments from the three subunits fold back independently. d) Hemifusion occurs when collapse of the intermediate has proceeded far enough to bring the two bilayers into contact, the apposed, proximal leaflets merge into a hemifusion stalk. e) Fusion pore formation, as the hemifused open into a pore, the final zipping up of the C terminal ectodomain segment snaps the refolder trimer into its fully symmetric, post-fusion conformation, preventing the pore from resealing. From [29]

Viral fusion proteins can be divided in three glycoprotein classes, class I, II and III [26].

Class I glycoproteins are present in viruses such as IFV, HIV, EBOV or Influenza and is probably the best studied class. In the pre-fusion state, class I virus-cell fusion proteins are α-helix rich trimers of non-covalently associated heterodimers. They insert fusion peptides that are located at the N-terminal of the fusion protein [26].

Class II virus-cell fusion proteins, such as Dengue E glycoprotein, are β-sheet-rich pre-fusion homo- or heterodimers, employing barrel domains to assemble membrane trimeric hairpins [29]. They insert hydrophobic fusion loops (internal fusion peptides connected to fusion protein at both terminals) into membranes [26, 28].

Class III from viruses like vesicular stomatitis virus [29], combine structural signatures found in classes I and II, i.e., they are trimers, with both α-helices and β-sheets that dissociate into monomers. They insert hydrophobic fusion loops into membranes, and oligomerize into post-fusion trimers [26, 28].

In the process of membrane fusion, fusion peptides insert into the host membrane, and are thought to promote lipid disorder and lipid tail protrusion, increase membrane curvature and promote polar head intrusion (sinking of nearby lipid head-groups and dimpling of the membranes). The result is the destabilization of the membrane and the promotion of the fusion between the cell and the virus membrane [28, 31, 32]. During membrane fusion,

orientation, insertion depth and structure of the FPep in the membrane are factors that influence the process [31].

Despite the large number of studies focusing on these peptides, the mechanism of action of fusion peptides remains unclear and several mechanisms have been suggested [31].

2.3.2 *Biophysical properties of viral fusion peptides*

FPeps comprise conserved hydrophobic domains absolutely required for the fusogenic activity of glycoproteins from all enveloped virus families [26].

Some fusion peptides are located at the extreme N-termini of the transmembrane subunits of the fusion proteins (e.g. IFV and HIV), whereas others are internal fusion loops (e.g. Dengue). They are relatively conserved in terms of amino acid composition within the same species. In many cases, even relatively conservative single amino acid changes in the fusion peptide completely abolish the ability of fusion proteins to fuse membranes, while other structural and functional properties of these proteins may remain intact [30].

Concerning the primary structure, FPeps have an intraspecies sequence homology as high as 90% having high residue conservation and high alanine/glycine content which is uncommon for hydrophobic protein domains and suggests an intrinsic conformational flexibility required for membrane insertion. In particular when located internally, they contain invariable proline residues. Aromatic residues are also frequently present in fusion protein sequences. These residues may help overcome the energy cost of peptide partitioning into membranes, enable membrane insertion of internal FPeps or contribute to stabilize the insertion of FPep in one bilayer leaflet [26]. Furthermore, the fusion peptide length influences its structure which has a strong correlation with function [32].

Viral fusion peptides sequences are structurally constrained to access the three states of pre fusion, active form and post fusion [26].

In the first stage, peptides are constraint to fold after synthesis as a constituent of globular ectodomains, i.e., the sequence must remain stable in folding within the globular ectodomains of glycoprotein complexes (inactive state) [26]. As they get transferred to target cell, they adopt specific conformations therein (active state). The conformation adopted by fusion peptides in membranes and both the insertion depth and angle vary on factors such as the membrane lipid composition (particularly the presence of cholesterol or anionic phospholipids) or peptide length [26]. Finally, in some cases, they should co-assemble with transmembrane anchors coupled to fusion pore extension [26].

3

MACHINE LEARNING

Machine Learning can be defined as a branch of artificial intelligence that systematically applies algorithms to induce the underlying relationships within data [33]. ML encompasses computational methods that are able to discover hidden non-obvious patterns in a training dataset and, by building a mathematical model, predict future events or scenarios in new data [34, 35]. In 1959, A. Samuel described ML as the "field of study that gives computers the ability to learn without being explicitly programmed" [36]. ML has become essential on the field of information technology and with the increasing amounts of data it is reasonable to assume that it will become even more prominent in the future [37]. It has been applied to multiple computational biology problems, where the generation of biological and clinical data has increased exponentially, particularly after the advent of the next generation sequencing technologies [34, 38].

In machine learning, an instance or example, represents one row of the dataset that contains features as columns, including possibly a target/output variable. The training set is used to train the model and the test set is used to calculate the performance metrics of the model. The features that describe the instance are the input variables and can be categorical or continuous. The target variable is used to train supervised models and is often called label [39, 33].

ML algorithms fall into two broad classes: unsupervised and supervised algorithms suited to address distinct questions that are based on the desired output and the type of input available for training the model. Whereas unsupervised machine learning intends to extract knowledge from inputs alone, supervised machine learning goal is to infer the relationship between the observed data (input data) and a target variable (output variable). Additional classes include semi supervised algorithms (combine labelled and unlabelled data) [38].

3.1 UNSUPERVISED MACHINE LEARNING

In unsupervised learning, the objective is to discover hidden and new patterns within data and understand the underlying organizational structure of data [33]. The dataset is usually composed of only inputs, from which the algorithms are expected to extract

knowledge [39]. One difficulty in unsupervised learning is evaluating whether the algorithm learned something useful. As a consequence, unsupervised algorithms are used often in an exploratory setting, to understand the data better and as a pre-processing step for supervised algorithms [39]. Their goal is to hypothesize representations of the input data in order to filter information, clustering and for an efficient decision making. These algorithms have been used in areas as data compression (discover dominant features in data), outlier detection, classification and others. They are best applied within a single homogeneous dataset [33, 38]. Dimensionality reduction and clustering are examples of unsupervised learning.

3.1.1 *Dimensionality reduction - principal component analysis*

Dimensionality reduction allows to summarize the essential characteristics (represented by a small number of features) of a high dimensional data representation (consisting of many features). The most common motivations are visualization, compressing the data, and finding a representation that is more informative for further processing [39]. One of the algorithms used for this purpose is the *Principal component analysis (PCA)*. PCA generates novel latent features in the data that provide the most signal (the first PCA is the latent feature that explains most of the variability in the data) [38]. PCA achieves its results by rotating the dataset in a way such that the rotated features are orthogonal [39]. Besides the PCA algorithm, there are the non-negative matrix factorization (NMF) algorithm, which is commonly used for feature extraction, and t-SNE algorithm (commonly used for visualization using two-dimensional scatter plots) [39].

3.1.2 *Clustering*

Clustering algorithms allow partitioning the dataset into clusters, i.e., groups of similar items. The data is split in a way that a cluster contains similar points and points in different clusters are different. Algorithms for clustering include the K-means (most commonly used), agglomerative clustering or hierarchical clustering [39]. After unsupervised learning detects clusters, supervised algorithms can classify new samples by assigning them to the closest cluster [38].

3.2 SUPERVISED MACHINE LEARNING ALGORITHMS

Supervised algorithms attempt to infer the relationship between the observed data (input data) and a target variable (dependent/output variable) that is subject to prediction. The algorithm uses the training data to synthesize the model function that generalizes the relation

between the input of feature vectors and the desirable output value [33]. The overall goal is to make predictions for unseen data [39].

A well-trained model based on a supervised learning algorithm can accurately predict the class labels for hidden examples embedded in unfamiliar or unobserved data instances [33]. These algorithms can be better at ignoring dataset specific noise and are particularly suitable for integrative analysis of broad data collections (collections that span multiple datasets) which is relevant in the application of these models in biological contexts [38].

In supervised learning, we may distinguish models for classification, where output values belong in a discrete set, and regression models, where output values are within a continuous set [40]. In classification problems, a classifier trains a model representing the relationship between features and the class label in the training set, i. e. it takes the values of various features of an instance to classify and predicts to which class the instance belongs. Classification can be separated into binary or multiclass classification if it involves two or more than two classes, respectively [39]. In regression problems, the goal is to predict a continuous number, i.e., there is continuity between possible outcomes [39].

Below, a brief description of the most popular machine learning algorithms is addressed.

3.2.1 *Hidden markov models*

Hidden markov models (HMM) are stochastic models related to Bayesian networks [35]. They consist of a finite set of nodes representing 'hidden states' interconnected by links describing the probabilities of a transition between the individual states. They are defined by a finite set of states, a discrete alphabet of symbols, a probability transition, and a probability emission matrix. When the system is in a given state i, it has a probability tji of moving to state j and a probability eiX of emitting symbol X. The emissions and transitions depend only on the current state and only the emitted symbols can be visible not the underlying walk between states ('hidden'). It is not possible to express dependencies between non-neighboured states. Furthermore, model parameters increase with the size of alphabet employed, which is a downside when considering for example protein models [35, 41].

3.2.2 *K nearest neighbour*

In *k-nearest neighbour algorithms (KNN)* algorithms, used both for regression and classification, the distances between samples and training data are calculated in a descriptor hyperspace, i.e., to make a prediction for a new point, the algorithm finds the closest point in the training dataset, its nearest neighbour [39]. In classification, the prediction is determined by the class of the majority of the k nearest points and in regression by the average of the k nearest points [40]. The number of neighbours considered and the distance measure (e.g. Euclidean

distance) are important parameters in this algorithm. The method requires pre-process previously the data, not working well with datasets with many (hundreds or more) features [39].

3.2.3 *Linear methods*

Linear models make a prediction using a linear function of the input features. There are differences if the model is set for regression or classification. In regression the output is a linear function like a plane or hyperplane, of the features. The simplest method used is the linear regression, or ordinary least squares, which has no parameters and does not allow to control model complexity. Other linear models are available such as ridge regression where coefficients, that should be close to zero, are chosen not only to predict well on the training data but to control model complexity [39]. In classification, the formula is similar but instead of returning the weighted sum of features, the returned value represents probability of belonging to a class.

The coefficients W and the intercept b can be calculated through the decision boundary function, which separates two classes using a plane or a hyperplane. These models are fast to train and predict, they work well with large datasets and sparse data and are intuitive to understand. With highly correlated features, however, the coefficients can be hard to interpret. Also, they do not model nonlinear relations between the data [39].

3.2.4 *Naïve bayes*

The Bayes theorem describes the conditional probability of an event, based on prior knowledge about the problem. Naïve Bayes algorithms are based on Bayes theorem. These algorithms assume that, regardless of the existence of correlations between features, the values of individual features are independent from the values of any other feature, given the predicted value (class variable). These ML algorithms work by calculating a likelihood function between the relative frequencies of each attribute and the frequency of the class variable. The class prediction will correspond to the highest likelihood probability. Despite of simple and naive assumptions, they have shown to perform well on classification tasks [42]. They are quite similar to the linear models but tend to be even faster in training and predicting, work well with large, high dimensional and sparse data and are relatively robust to parameter choice [39].

3.2.5 *Decision trees*

Decision trees, used for both classification and regression, are flowchart-like structures used to determine a course of action or outcome. Each node represents a 'test' (if/else questions) in an attribute, each branch of the tree represents a possible decision (based on the values of input features) and each leaf represents a class label (decision after computing all attributes). They learn a hierarchy of if/else questions, leading to a decision [39, 40, 43]. The tree is structured to show how and why one choice may lead to the next, with branches indicating that each option is mutually exclusive. The root node is the starting point and leaf nodes contain questions or criteria to be addressed while the branches connects the nodes [40].

Decision tree learning uses a decision tree as a predictive model. Trees can be classification trees when the target variable is a discrete set of values and leaves represent class labels or regression trees where the target value can take continuous values [44].

A general framework to grow a tree is as follows: The algorithm first determines the best attribute (the attribute that has more information gain) in the dataset. This attribute is assigned as the decision attribute for the node and creates a new descendent of the node. This way, on each iteration, new decision trees are recursively generated by using the subset of data in each node. This cycle happens until a stage where it is not possible to further classify date is reached and the class is represented as a leaf node. To make a prediction, each attribute of the example is tested on the root node and goes on through next nodes until reach a leaf node, obtaining the final prediction [44].

Decision trees are prone to overfitting. To prevent this, pre and post-pruning can be applied. The pre-pruning of the tree is achieved by stopping the development of the tree before it is fully developed, limiting the maximum depth of the tree, limiting the maximum number of leaves or requiring a minimum number of points in a node to keep splitting. Post-pruning of the tree is achieved by building the tree removing or collapsing nodes that contain little information [39].

Decision trees algorithms do not vary with scaling of data, not needing pre-processing like normalization or standardization of features and can be easily visualized and understood. However, even with pruning they tend to have a poor generalization performance and therefore, are often used in ensemble methods, which combine multiple trees into one predictive model to improve performance [39, 40].

3.2.6 *Kernel support vector machines methods*

Kernelized *support vector machines (SVM)* are models for both regression and classification. In these models, feature points are projected in a high dimensional space where groups are separated using a hyperplane [40, 45]. The goal of SVM is to find a hyperplane that can

maximize the distance between the samples of different classes [46]. The distance to the hyperplane in a higher dimensional space can be computed using a polynomial kernel that computes all possible polynomials up to a certain degree of the original features or through the *radial basis function (RBF)* kernel, also known as the Gaussian kernel that corresponds to an infinite-dimensional feature space (it considers all possible polynomials of all degrees, but the importance of the features decreases for higher degrees) [39].

They are powerful methods that work well with sparse data, are less prone to overfitting and work well with both few and many features. Nonetheless, it is necessary to be careful with the pre- processing of data [39, 41].

3.2.7 *Artificial neural networks and deep learning*

Artificial neural networks (ANN) and deep learning are inspired by the idea of reproducing the functioning of the human brain with artificial neurons (processing unit) arranged in input, output and hidden layers [40, 47]. They are comprised of neurons arranged in layers and connections between neurons. The first layer is the input layer and the last one the output layer, all the layers between are hidden layers. In the hidden layers, each neuron receives input signals from other neurons, integrates those signals and then uses the result in a straightforward computation, i.e., each node on the left represents an input feature, the connecting lines represent the learned coefficients, and the node on the right represents the output, which is a weighted sum of the inputs. The connections between neurons are numerical weight values that are optimized during the network training, representing the stored knowledge of the network [40]. Examples of these models are the multilayer feed-forward network [47].

They are accurate models and they can work well in large amounts of data and complex models but require preprocessing of data as scaling, working best with 'homogenous' data. They also take considerable time to train [39].

Deep learning approaches are usually based on neural networks, are high dimensional data reduction techniques for constructing high-dimensional predictors in input-output models [48]. They are characterized for using a cascade of multiple layers (each layer uses the output from the previous) of nonlinear processing units for feature extraction and transformation [49]. They can handle very large data sets, find hidden structure within them while making accurate predictions [50].

3.2.8 *Ensembles*

Often, multiple models achieve a better performance when compared to a single model. Ensembles are methods that combine multiple machine learning models to create more

powerful models. The idea is to develop approaches to generate numerous independent models that, when combined into an ensemble improve generalizability and robustness over a single estimator [51, 52]. The final result is obtained by combining the values of models and returning a single value, for example, the classification made by more individual models [53]. Ensembles can be a combination of different algorithms or a combination of similar algorithms but varying their internal values [40].

A low-level algorithm, base learner, is a single ML algorithm that gets used by ensemble models [51]. An upper-level algorithm manipulates inputs to base learner in order to generate independent models. Upper level algorithms include methods that resample training data to create new models for each sample (bagging and boosting) and methods that introduce random choices in models or modify some algorithms initial parameters) [51].

Ensemble methods can be averaging methods or boosting methods.

In averaging methods, several estimators are built independently, and the predictions are the result of the average of their results. The combined estimator is usually better than a single estimator once its variance is reduced. Bagging and random forests are examples of this kind of methods [52].

Bagging, or bootstrap aggregating, takes repeated unweighted samples from data, the samples are without replacement within a sample (each observation is included in a sample only once) and within replacement across samples (observations can be included in all samples). As samples are unweighted, misclassified observations do not have weight from previous samples. The user selects the number and size of samples to generate. However, if a small percentage of observations is selected, the number of samples must be larger [53].

Random Forest (RF) are collections of decision trees, where each tree is slightly different from the others. The difference between trees is not only based on the sampling of observations used as training data (trees would split at similar features at each node) but on the selection of the features used. These models decide where to split based on a random selection of features, i.e., instead of searching for the most important feature and split the node, it searches the best feature among a random set of features. This way, each tree will split based on different features [54]. Feature importance is obtained by aggregating features importance of the singular trees. They don't tend to perform well on high dimensional sparse data but work well on large datasets [39].

The combination of a large number of trees that are then used to vote in some manner to build a stable and strong classifier, which allows retaining the predictive power of trees, reducing the tendency to overfit [53].

In boosting methods, the base estimators are built sequentially and try to reduce the bias of the combined estimator. Boosting algorithms include AdaBoost or Gradient Tree boosting. Boosting is similar to bagging; however, the weights of misclassified observations are increased in order to make them more likely to be selected in the sample. This way,

reweighting after each sample and 'boosting' the performance of the model [53]. Gradient boosted regression trees, as random forests, also combine multiple trees to create a more powerful model that can be used for regression. However, they build trees in a serial manner, where each tree tries to correct the mistakes of the previous one [39].

The computational cost of constructing an ensemble is not significantly larger than creating a single learner. The algorithms to combine base learners are simple and the creation of multiple models is comparable to the cost of creating a single learner that needs to run several times for hyper-parameter tuning [54].

3.3 Supervised machine learning workflow and key concepts

The process of developing ML algorithms may be decomposed in several steps. First, it is necessary to collect data and select a subset of data attributes that might be useful to solve the problem. The features should be extracted, the data transformed and features to use in the model should be selected. Additionally, data must be sampled in train, test and independent/validation dataset. The next step is where the model is trained. The training set is used by the algorithm to extract knowledge and, when applied to the test data set, allows the calculus of the performance metrics by comparing the real and predicted values of the target variable. It is also possible to tune the hyperparameters and perform model optimization. Finally, the validated model can be applied to previously unseen data. A schematic view of the supervised ML pipeline can be observed in Figure 2.

3.3.1 *Data preparation*

The first part of ML pipeline is to obtain and prepare the data to be used in the model building and training. This includes data collection and dataset construction, preprocessing, feature extraction and selection and splitting of the data into training, test and validation sets. All these topics are described below.

Data collection and dataset construction

The first step is to collect the data available and select a subset of data attributes that might be useful to solve the problem. The set of features extracted from an object can be considered as a signature describing the object and is usually organized in a feature vector [45]. Features (input variables or attributes) represent the data in a fixed length vector and can be binary, categorical or continuous [55].

A dataset is a matrix of data, where each column represents an attribute, and each row represents a different instance. Generally, the last column represents the output attribute

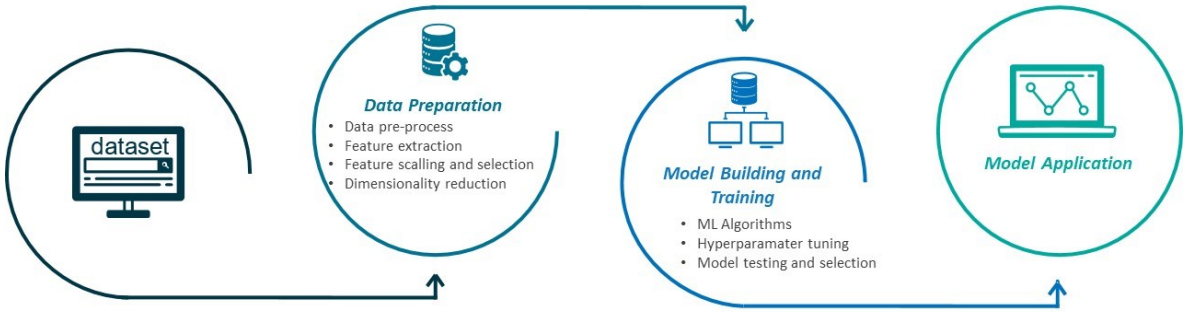


Figure 2: Supervised learning pipeline

and the remaining represent the input attributes. Building the benchmark dataset with reliable data is very important as ML depends heavily on data.

For classification problems the dataset must contain a subset with instances representing class members and a subset with instances representing non class members [56].

For training a classifier, one would usually select a positive/negative ratio of 1:1, however, in some cases, re-balancing data techniques achieve better results [24].

Feature extraction

Feature extraction transforms raw data, such as text or images, into features suitable for modelling. In models that take as input numerical values, categorical variables can be converted to one or more new features with values 0 and 1 using for example one-hot encoding. In One-hot encoding features are encoded as one-hot numeric array (one bit as 1 and all the others 0) [39]. Feature extraction can reduce the dimensionality of the dataset by transforming the original measure data with raw variables to features with strong pattern recognition (achieved for example using PCA algorithms) [57].

Preprocessing data / Feature transformation

Preprocessing data is one of the most crucial steps in machine learning since the raw data collected rarely comes in the form and shape that is necessary for the optimal performance of a learning algorithm [58].

Preprocessing includes formatting of data [33], and the removal or substitution of corrupt and missing data [33, 40]. One of the most used preprocessing transformations is standardization. Features can have different scales although they refer to comparable objects. Standardization is a common requirement for many machine learning estimators (for example SVM and linear models) as they need selected features to be on the same scale for optimal performance [58]. Standardization re-scales the data so it has the properties of a standard normal distribution with mean 0 and standard deviation of 1 [52, 55].

Feature selection

Feature selection is the process of selecting a subset of features that, ideally, is necessary and sufficient to describe the target concept. This selection is important since the feature set is the source of information for the learning algorithm [59]. Feature selection allows to deal with the curse of dimensionality by eliminating the redundant and irrelevant features, reducing the number of features to the most important ones. This allows the ML algorithm to train faster, reduce the complexity, improve accuracy and reduces overfitting of the model. Besides this, it can provide some understanding and knowledge about data [55]. Feature selection can be done using filter, wrapper or embedded approaches [60].

Filter techniques select features independent of machine learning algorithms and can be used as a preprocessing step. Features are selected based on scores in statistical tests for their correlation with the outcome variable. They are computationally simple and fast and are independent of the algorithm, however, they ignore the interaction with the algorithm [55, 60]. Univariate feature selection, for example, can include methods such as Pearson Correlation Coefficient, which measures linear correlation between two variables (from -1 meaning perfect negative correlation to 1 meaning perfect positive correlation, zero means no correlation with data), *Linear discriminant analysis (LDA)*, *analysis of variance (ANOVA)*, Chi-square or mutual information [37, 58, 60, 61].

Wrapper methods use a subset of features to train the model and assess its performance. The feature space is tailored to the specific algorithm used. Wrappers include the interaction between feature subset and model and can account for feature dependencies, however, these approaches are computationally expensive and prone to overfitting [55, 60]. Some examples may include forward selection (starts with no feature in the model, in each iteration, we add the feature which best improves our model), backward elimination (starts with all features and, at each iteration, the least significant feature is removed) or recursive feature elimination (greedy algorithm that repeatedly creates models and keeps aside the best or

the worst feature, constructing models until all the features are exhausted to, and ranking the features based on order of elimination) [37, 58, 60, 61].

Embedded techniques incorporate feature subset generation and evaluation in the training algorithm, i.e., the search for an optimal subset of features is built into the classifier construction. They account for the interaction with the model, being less computationally intensive than wrapper methods [55, 60, 62]. Popular methods include LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting and improve generalization. LASSO regression performs L1 regularization, it adds penalties equivalent to absolute value of the magnitude of coefficients, forcing weak values to have zero as coefficients, producing sparse solutions. RIDGE regression performs L2 regularization, it adds penalties equivalent to the square of the magnitude of coefficients forcing the coefficient values to be spread out more evenly, being more useful for feature interpretation rather than feature selection [37, 58, 60, 61].

Sampling data

In order to estimate the error of different models, it is necessary to divide our data into training and test data. The training set is the data used to train the machine learning model. The test set is used to evaluate the performance of the final model and validate it [33]. After being trained, the effectiveness of training and the performance of the algorithm is assessed by evaluating the ability to accurately predict the output data on the test set. The division of the dataset must be random to guarantee that the model is a good generalization of the data and not only of the training set [58]. Data need to be sampled at regular or adaptive intervals [33] and samples divided should be representative of the whole population [40]. When the test set is used to tune hyperparameters of the models, the evaluation of performance may not reflect the true performance of the model in a real-world application. Therefore, a third set of independent data is created on which the model will run. As the model is running in a previous unseen data the performance values will be more accurate [14, 63].

3.3.2 *Model selection, training, evaluation and optimization*

To train the model, first, it is necessary to choose the algorithm to apply to the previous generated data. The algorithm's choice depends on the type of data and the problem posed. To have a more robust model, it is possible to take advantage of ensembles methods using a combination of different algorithms or a combination of similar algorithms but varying the values of their internal parameters [40].

After the model is trained, it must be evaluated. This evaluation allows optimization and selection of the best model and is made by exposing the trained model to the test dataset [33].

Errors are usually derived from model bias (incorrect assumptions in the algorithm that can result in the model missing underlying relationships), model variance (sensitivity to small fluctuations in training set like outliers, missing data, errors in measurement and calculation of training data) and irreducible errors, with the total error being the sum of these [39]. There are many models proposed and their performance depends on the algorithm they analyse the data with [45]. Each algorithm has its inherent biases and therefore, it is essential to compare different algorithms in order to select the best performing model [58].

Performance metrics

The assessment of performance depends on the type of problem and the metrics to be used must be chosen carefully [39].

In classification problems, confusion matrices are often used. Confusion matrices show, in a clear way, correct and incorrect classifications for each class. Rows correspond to real classes and columns to the predicted classes. This representation allows the easy assessment of *true positive (TP)* and *false positive (FP)* and *true negative (TN)* and *false negative (FN)* [39, 51, 58]. Accuracy, also known as *Percentage of Examples Correctly Classified (PECC)* can be calculated by the number of correct predictions (TP+TN) divided by the number of all samples and represents how often the classifier makes the correct prediction [39, 51, 58]. Precision (positive predictive value), measures the samples predicted positive that are actually positive. It is given by the number of TP divided by all the predicted positives (TP+FP). On the other hand, recall measures how many positive labels are successfully predicted amongst all positive labels. Recall is obtained by the division of TP by the sum of TP and FN [39, 51, 58].

F1 score is a harmonic mean between precision and recall. Being a harmonic mean, the F score tends toward the smaller of the two elements, this way, it is going to be small if either precision or recall is small. It ranges from 0 to 1, with larger values corresponding to better predictions.

Furthermore, a precision-recall curve, and a *receiver operating characteristic (ROC)* curve can be designed. A precision-recall curve describes the relation between the true positive rate, recall (x axis) and the positive predictive value, precision (y axis), using different probability thresholds (between 0 and 1). The ROC curve, plots False positive rate on the x axis and True positive Rate (recall) on y axis between the values of 0 and 1. The *Area under curve (AUC)* measures the area underneath the ROC curve, it takes the value of 0, if all the predictions are wrong and 1, if all the predictions are correct [39, 51, 58]. ROC curves are more appropriate when observations are balanced between each class (there are roughly equal numbers of observations for each class) whereas precision-recall curves can be appropriate when dealing with imbalanced datasets [64].

Additionally, specificity and negative predicted value can be calculated. Specificity (error type I) is obtained by the division of TN by the sum of TN and FP. The negative predicted value is obtained by the division of the TN by the sum of TN and FN.

Matthews correlation coefficient (MCC) is used as a measure of the quality of binary classifications. It takes into account true and false positive and negatives and is considered a balanced measure which can be used even if the classes are of very different sizes. It returns a value between -1 (total disagreement) and 1 (perfect prediction)[65]. MCC can be calculated as follow:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1}$$

On regression problems, error metrics are based on error presented by each example (difference between real and predicted value) and can be calculated for example through the *sum of squared errors (SSE)*, *Root mean squared error (RMSE)* or R-squared [66]. RMSE, also known as standard error, measures the difference between predicted values and true values. It ranges from 0, up with values close to zero representing better predictions. R-squared measures the correlation between predicted values and actual observed values. It ranges from 0, no correlation, to 1, perfect correlation [39, 51, 58].

Cross validation

Cross validation is a resampling method. In cross validation the training dataset is divided into training and validation subsets in order to estimate the generalization performance of the models [58]. In k-fold cross-validation, k is the number of folds in which the dataset is equally partitioned. Then, a sequence of models is trained using each of the folds as test set and the remaining as training set [39]. In each round, K-1 folds are training the model and the remaining is used as the test data. Each example is in the training set once, which improves the ability of generalization of the model. Leave-one-out is a cross validation technique where k equals the total number of samples, i.e., each fold is a single sample. Although being less arbitrary, is very time consuming when considering large datasets [39]. Cross validation techniques can give an accurate estimate of the true error. Considering the division of the dataset into k-subsets, the true error is the average of the errors obtained by training the different k models [44].

Underfitting and overfitting

Underfitting occurs when the model is not capable of adequately describing the relationship between inputs and outputs, having a high bias [40]. On the other end, overfitting occurs

when the model becomes too complex (for example due to a high number of parameters) and, although the accuracy representing training data is very high, the performance with test data is not [40]. In an overfitting scenario, the model fits too closely to the train set but not to the test set, not being able to generalize to new data. In an underfitting scenario, on the other hand, the model is too simple to capture the variability in data. The objective is to generate a model that generalizes as accurately as possible [39].

Hyperparameter optimization

Hyperparameters are model parameters not learned in the training phase, that represent the knobs of models. The default values for these parameters are typically not optimal for our specific problem and their alteration can improve or impair the learning process. In hyperparameter optimization their values are changed to a value close to optimal, regarding our problem specific task [40, 58]. A grid search can be performed where all the combinations of parameters of interest are tried. To do this, it is necessary to divide the dataset in three datasets as explained above using an independent test additionally to the train and test dataset or, using a cross validation technique on the train dataset [39].

3.3.3 *Model application and prediction*

Lastly, the validated model can be applied in an actual task of prediction. The process of training can coexist with the prediction task, this is, as new data are encountered, the model can be retrained following the same pipeline [33].

3.4 RELEVANT PACKAGES AND TOOLS

The code relevant for this thesis will be implemented in the python language. Taking that into account, the most relevant packages for machine learning are *NumPy*, *SciPy*, *Matplotlib*, *pandas*, *Scikit-learn*, *Keras* and *TensorFlow*.

NumPy is a package that contains functionality for multidimensional arrays and high-level mathematical functions [67]. *pandas* library provides high-performance data structures tools for data analysis [68]. *Matplotlib* is important for visualization of data [69]. *SciPy* contains a collection of functions for scientific computing in Python, including general and specialised tools for data management and computation and high performance computing [70].

Scikit-learn is a free machine learning library built on *NumPy*, *SciPy* and *Matplotlib*. It integrates a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems [52]. *Keras* is a deep learning python library capable of performing high level neural networks. It runs on top of packages such as *TensorFlow* [71].

TensorFlow is a machine learning library used as interface for expressing and manipulate machine learning algorithms [72].

4

MACHINE LEARNING APPLIED TO PEPTIDES

Biological sequences (DNA, RNA and protein sequences) and other relevant biological data are growing exponentially with the development of sequencing tools and other high-throughput technologies. However, due to the limitations of wet lab experiments, the gap between sequence data and structural data is increasing. Computational methods that analyse sequence data have been applied in an effort to understand structure, function and drive the development of applications of this knowledge (e.g. disease research or precision medicine) in a fast, inexpensive and efficient way. As the majority of tasks in the field of sequence analysis are binary or multitask classification tasks, machine learning has a key role in this area and has been applied for example to cancer detection and diagnosis or protein secondary structure prediction [14].

Proteins are the major components of cell activities. Peptides scanned from whole protein sequences are the core information to functional site prediction, protein structure identification or protein function recognition. Therefore, peptide classification aims to find a model that maps peptide residues to functional status [63]. The classification model can be built using a learning algorithm with similar principles to those used in pattern classification systems and described in the previous chapter. The major steps to analyse biological sequence data are therefore, beginning from the benchmark dataset, the feature extraction and selection, the predictor construction and the performance evaluation.

4.1 PROTEIN FEATURE EXTRACTION

Feature extraction is the first step to build a computational model for biological sequences. A biological sequence can be represented as a succession of L residues, where L is the length of the sequence [73].

The simplest method to describe a protein is its entire aminoacid sequence. This sequence can be used in sequence similarity search based tools (e.g. BLAST). However, this fails when the query protein does not have significant homology to proteins with known function [74].

Besides that, ML methods as SVMs, correlation coefficient method or RF, are based on vector mode rather than sequential and cannot perform directly on the sequence. This

way, protein sequences should be converted into fixed length feature vectors that contain information regarding patterns and sequence-order effects on residues. When compared to DNA or RNA sequences, feature extraction methods for proteins raise some difficulties due to the diversity of aminoacids and the various structures and functions of proteins [8, 14]. Taking this into account, it is necessary to have methods that can properly identify protein characteristics from the primary sequences of proteins [75].

Feature extraction methods may include physicochemical, aminoacid composition, pseudo aminoacid composition, autocorrelation based features, composition, transition and distribution, conjoint triad and Quasi sequence order descriptors. In alternative, binary profiles can be generated. These descriptors are further detailed below.

4.1.1 *Physicochemical descriptors*

Physicochemical features are highly useful to represent and distinguish proteins or peptides of different structural, functional and interaction properties and have been widely used in protein prediction problems [76]. This group of descriptors includes mostly one dimension peptide representations such as length, charge, molecular weight, hydrophobic ratio, *grand average of hydropathy (GRAVY)*, aromacity score, isoelectric point, number of C,H,N,O and S atoms (atomic composition), number of each bond type and others. Boman index, a descriptor proposed by Boman can also be computed [77]. This descriptor computes the potential protein interaction index based in the amino acid sequence of a protein. The index is equal to the sum of the solubility values for all residues in a sequence, and it might give an overall estimate of the potential of a peptide to bind to membranes or other proteins [77]. Descriptors of this kind have been used in several studies to distinguish peptides [24, 12, 78].

4.1.2 *Residue composition descriptors*

The primary sequences of proteins are composed of 20 amino acids. This sequence can be described in several ways.

Aminoacid composition

Aminoacid Composition (AAC) represents the fraction of each amino acid type within a protein [8, 12, 74, 75]. The fraction of all 20 natural aa are calculated as:

$$f(r) = \frac{Nr}{N} \qquad r = 1, 2, 3, ..., 20 \tag{2}$$

where Nr is the number of the amino acid type r and N is the length of the sequence.

The AAC is a simple but powerful feature, is computationally tractable and, as described in a paper by Roy et al [79], can be used to predict protein interactions with good performance. In previous machine learning models, it has been demonstrated that anticancer and non-anticancer peptides have significant differences considering the AAC [12, 75, 80].

Dipeptide composition

The *Dipeptide Composition (DPC)* describes the total number of dipeptides normalized by all the possible combinations of dipeptides present in the given peptide sequence. It returns a 400-dimensional descriptor [81] and is defined as:

$$f(r,s) = \frac{Nrs}{N-1} \qquad r,s = 1,2,3,...,20 \tag{3}$$

where Nrs is the number of dipeptide represented by amino acid type r and s.

Adjoining DPC reflects the correlation between two adjoining amino acids. However, in a 3-Dimensional space, and taking into consideration the secondary structures of proteins, two amino acids with g-gap residues may be adjacent. A value of g of 0 is the adjoining DPC, g of 1 describes the correlation of two residues with one residue interval and so forth [82]. DPC composition, and g-gap DPC (reduce dimensionality) has demonstrated promising results in computational proteomics and has been used in anticancer peptides classification [7, 8, 9, 12, 80].

Tripeptide composition

Tripeptide composition (TPC) describes the total number of tripeptides normalized by all the possible combinations of tripeptides present in the given peptide sequence [81], it gives a 8000 feature vector and is defined as:

$$f(r,s) = \frac{Nrst}{N-2} \qquad r,s,t = 1,2,3,...,20 \tag{4}$$

where Nrst is the number of tripeptide represented by amino acid type r, s and t.

Reduced aminoacid composition

Reduced amino acid composition (RAAC) represents a solution to overcome the high dimensional feature vector issue. Here, the basic aas are grouped together into a smaller number of representative residues based on physicochemical properties. This way, RAAC allows for the minimization of the complexity vectors and enhances the ability to find structural similarity of the peptides. RAAC has been used for protein family characterization [8, 75].

Using AAC based methods, all the sequence order effects are lost; for example, correlations among the amino acids in proteins are ignored (residues apart in sequences may be neighbouring in protein 3-dimensional structure) [14, 74].

4.1.3 *Autocorrelation based descriptors*

Autocorrelation descriptors are defined based on the distribution of amino acid properties along the sequence. Autocorrelation descriptors include the Normalized Moreau-Broto autocorrelation descriptor, Moran autocorrelation and Geary autocorrelation Descriptor [81, 83].

They quantitatively measure the autocorrelation information of aminoacid residues, based on eight properties: hydrophobicity scale, average flexibility index, polarizability parameter, free energy of amino acid solution in water, residue accessible surface area, aa residue volume, steric parameters and relative mutability [81, 83]. All the aa indices are centralized and standardized before the calculation [81, 83].

Normalized Moreau-Broto autocorrelation descriptors

Moreau Broto autocorrelation descriptors for protein sequence may be defined as:

$$AC(d) = \sum_{i=1}^{N-d} P_i P_{i+d} \qquad d = 1, 2, 3, ..., nlag \tag{5}$$

where d is called the lag of the autocorrelation and Pi and Pi+d are the properties of the amino acids at position i and $i + d$, respectively. $nlag$ is the maximum value of the lag.

The normalized Moreau-Broto autocorrelation descriptors are defined as:

$$ATS(d) = \frac{AC(d)}{N - d} \qquad d = 1, 2, 3, ..., nlag \tag{6}$$

Moran autocorrelation descriptor

Moran autocorrelation descriptors to protein sequence may be defined as:

$$I(d) = \frac{\frac{1}{N-d} \sum_{i=1}^{N-d} (P_i - \overline{P})(P_{i+d} - \overline{P})}{\frac{1}{N} \sum_{i=1}^{N} (P_i - \overline{P})^2} \qquad d = 1, 2, 3, ..., 30 \tag{7}$$

where d is called the lag of the autocorrelation and P_i and P_{i+d} are the properties of the amino acids at position i and $i + d$, respectively. $\overline{P}$ is the average of the considered property P along the sequence, i.e.,

$$\overline{P} = \frac{\sum_{i=1}^{N} P_i}{N} \tag{8}$$

Geary autocorrelation descriptor

Geary autocorrelation descriptors application to protein sequence may be defined as:

$$C(d) = \frac{\frac{1}{2(N-d)} \sum_{i=1}^{N-d} (P_i - P_{i+d})^2}{\frac{1}{N-1} \sum_{i=1}^{N} (P_i - \overline{P})^2} \qquad d = 1, 2, 3, ..., 30 \tag{9}$$

where d is called the lag of the autocorrelation and P_i and P_{i+d} are the properties of the amino acids at position i and $i + d$, respectively, $\overline{P}$ is the average of the considered property P along the sequence.

For each amino acid index, there will be $30 \times nlag$ autocorrelation descriptors, being possible to calculate 240 features for each autocorrelation type [81, 83].

These features are especially useful for protein remote homology detection and fold recognition because they are able to extract the sequence patterns among proteins sharing low sequence similarities [14]. They are described to have good results in protein predictions problems [84, 85, 86].

4.1.4 *Composition, transition and distribution*

Composition, Transisition, Distribution (CTD) feature is composed of three descriptors, *Composition (C)*, *Transition (T)* and *Distribution (D)*, which are based on 7 physicochemical attributes: hydrophobicity, polarity, polarizability, charge, secondary structures, solvent accessibility and normalized Van der Waals volume. As described in table 1, the amino acids are divided in three classes according to their attribute with each amino acid being encoded by an index (1, 2 or 3) according to which class it belongs. After setting the amino acid classes, C is calculated. C is the global percentage for each encoded class in the sequence (number of amino acids of a particular property (such as hydrophobicity) divided by the total number of amino acids in a protein sequence). T represents the percent frequency with which class is followed by another (e.g. 1 followed by 3 or 3 followed by 1) in the encoded sequence. D characterizes the distribution patterns of amino acids of each class in the sequence. It represents the position percentages in the whole sequence for the first residue, 25% residues, 50% residues, 75% residues and 100% residues for a specific encoded class [81].

This feature can be applied in various biological problems, such as the prediction of antimicrobial peptides with high accuracy [24], and is described in literature as having good results in protein predictions problems [84, 85, 86].

Table 1: Amino acid attributes and the division of the amino acids into three groups for each attribute to calculate CTD descriptors. Adapted from *PyDPI* package manual [81].

	Group 1	Group 2	Group 3
Hydrophobicity	Polar R,K,E,D,Q,N	Neutral G,A,S,T,P,H,Y	Hydrophobic C,L,V,I,M,F,W
Normalized van der waals	0-2.78 G,A,S,T,P,D	2.95-4.0 N,V,E,Q,I,L	4.03-8.08 M,H,K,F,R,Y,W
Polarity	4.9-6.2 L,I,F,W,C,M,V,Y	8.0-9.2 P,A,T,G,S	10.4-13.0 H,Q,R,K,N,E,D
Polarizability	0-1.08 G,A,S,D,T	0.128-0.186 C,P,N,V,E,Q,I,L	0.219-0.409 K,M,H,F,R,Y,W
Charge	Positive K,R	Neutral A,N,C,Q,G,H,I,L, M,F,P,S,T,W,Y	Negative D,E
Secondary Structure	Helix E,A,L,M,Q,K,R,H	Strand V,I,Y,C,W,F,T	Coil G,N,P,S,D
Solvent accessibility	Buried A,L,F,C,G,I,V,W	Exposed R,K,Q,E,N,D	Intermediate M,S,P,T,H,Y

4.1.5 *Conjoint triad descriptors*

Conjoint triad (CTriad) descriptors were proposed in 2007 by Shen et al. to predict *Protein-Protein Interactions (PPI)* [87]. CTriad consider the properties of the aa and regard any three continuous aa as a unit. The 20 aa were clustered into seven classes according to dipoles and volumes of the side chains as they reflect electrostatic and hydrophobic interactions which are essential for protein-protein interactions and also for the interaction with other molecules, such as lipids [81, 83]. In this way, there are $7 \times 7 \times 7$ (343) different possible triads and the feature vector produced will reflect the frequency of each triad in the protein sequence. The triads can be differentiated according to the classes of aa, i.e., triads composed by three aa belonging to the same classes could be treated identically, as they may play similar roles [87].

Considering V the vector space of the sequence features, each feature V_i represents a sort of triad type, F the frequency vector corresponding to V and fi the frequency of type V_i appearing in the protein sequence, the detailed description for (V, F) is illustrated in Fig 3 [81, 83].

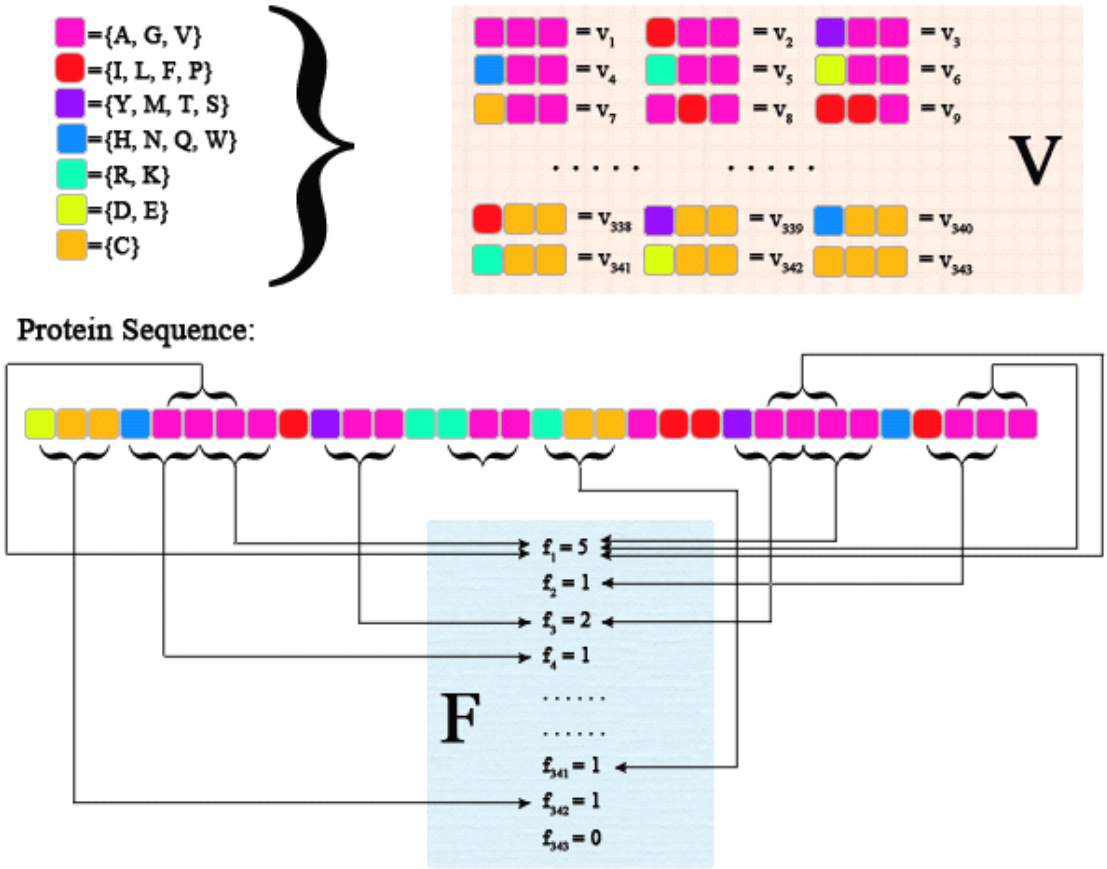


Figure 3: Schematic diagram for constructing the vector space (V,F) of protein sequence for Conjoint Triad descriptors. V is the vector space of the sequence features; each feature (V_i) represents a triad composed of three consecutive amino acids; F is the frequency vector corresponding to V, and the value of the ith dimension of $F(f_i)$ is the frequency that vi triad appeared in the protein sequence. From [81, 83].

The length of a protein, number of aa residues, influences the value of f_i. In general, a long protein would have a large value of f_i, which complicates the comparison between two heterogeneous proteins. Because of this, f_i is normalized with the following equation:

$$d(i) = \frac{f_i - min\{f_1, f_2, f_3, ...f_{343}\}}{max\{f_1, f_2, f_3, ...f_{343}\}} \tag{10}$$

The numerical value of d_i of each protein ranges from 0 to 1, which thereby enables the comparison between proteins.

Conjoint Triad have been used to study protein-protein interaction [87], levels of EC hierarchy and enzyme subfamily [88, 89].

4.1.6 *Sequence order descriptors*

Sequence order descriptors were proposed by K.C. Chou [90], being derived from both the Schneider-Wrede physicochemical distance matrix and the Grantham chemical distance matrix between each pair of the 20 aminoacids. The physicochemical properties computed include hydrophobicity, hydrophilicity, polarity and side-chain volume. These features are able to represent aa distribution patterns of a specific physicochemical property along peptide sequence [84].

Here, we can distinguish two type of features, sequence order coupling numbers and quasi-sequence order. For a protein chain of n aa residues $R_1, R_2, R_3 \dots R_n$, the sequence order effect can be described through a set of sequence order coupling numbers that reflect the coupling mode between all of the most contiguous residues along a protein sequence [81, 83]. A schematic view is presented in Fig 4. The dth-rank sequence-order-coupling number is defined as:

$$\tau_d = \sum_{i=1}^{N-d} (d_{i,i+d})^2 \qquad d = 1, 2, 3, ..., maxlag \tag{11}$$

where $d_{i,i+d}$ is the 'physicochemical' distance between the two amino acids at position i and $i + d$. $maxlag$ is the maximum lag and the length of the protein must be not less than $maxlag$.

Quasi-sequence order is derived from the coupling numbers but takes into account the frequency of each aa and the sequence order coupling number [81, 83].

For each aa a *Quasi Sequence order descriptors (QSO)* can be defined as:

$$X_r = \frac{f_r}{\sum_{r=1}^{20} f_r + W \sum_{d=1}^{maxlag} \tau_d)} \qquad r = 1, 2, 3, ..., 20 \tag{12}$$

where f_r is the normalized occurrence for amino acid type i and w is a weighting factor (n=0.1). these are the first 20 quasi-sequence-order descriptors. The other 30 quasi sequence order are defined as:

$$X_r = \frac{w\tau_{d-20}}{\sum_{r=1}^{20} f_r + w \sum_{d=1}^{maxlag} \tau_d)} \qquad r = 21, 22, 23, ..., 20 + maxlag \tag{13}$$

The number of descriptors produced varies accordingly to the $maxlag$ chosen.

QSO descriptors have been used to predict protein subcellular location [90], proteins binding affinity [91] or protein functional families [84].

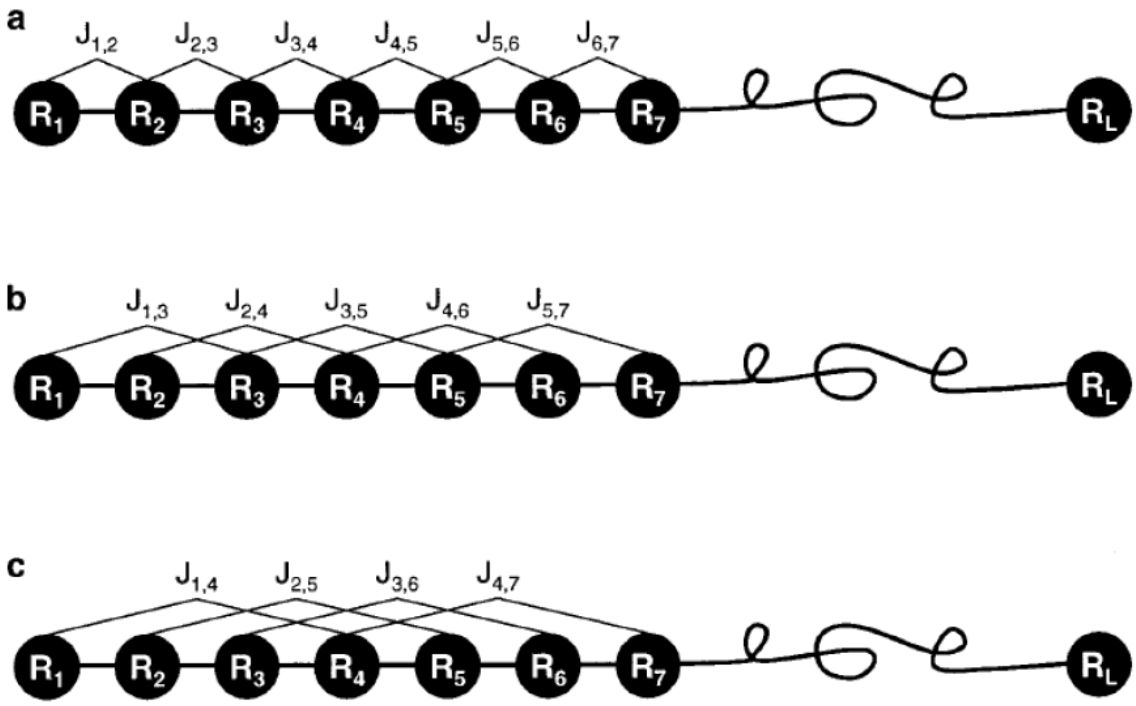


Figure 4: Schematic drawing to Sequence Order Coupling numbers of the (a) the 1st-rank, (b) the 2nd-rank, and (c) the 3rd-rank. (a) Reflects the coupling mode between all the most contiguous residues, (b) that between all the 2nd most contiguous residues, and (c) that between all the 3rd most contiguous residues. From [90].

4.1.7 *Pseudo aminoacid composition descriptors*

Chou's *Pseudo aminoacid composition (PAAC)* of a protein allows to deal with the aminoacid composition considering sequence order correlation. The sequence order correlation is calculated based on properties that play an important role in protein folding, interaction with both the environment and other molecules and function: the values of position, hydrophobicity, hydrophilicity or side chain mass of aa [8, 11, 14, 74]. For example, many helices in proteins are amphiphilic, this is, they are formed by the hydrophobic and hydrophilic aa according to a special order along the helix chain [92].

In PAAC descriptors, whereas the first 20 components reflect the conventional AAC, the remaining PAAC components reflect the correlation patterns, hence incorporating sequence order correlation patterns [74].

These descriptors englobe the PAAC, also called the type 1 pseudo-aminoacid composition and the *Amphiphilic pseudo aminoacid composition (APAAC)*, also called the type 2 pseudo-aminoacid composition.

These concepts proposed by Chou in 2001 and in 2005 have been extensively utilized in various fields of protein structure and function prediction [74, 84] such as homology detection, DNA-binding protein identification [8, 14], prediction of subfamily enzyme classes [92] and prediction of anticancer peptides [11].

4.1.8 *Base class peptide descriptors*

Base class peptide descriptors englobe the calculus of moment of a sequence and auto and cross correlation of aa values. For this calculus, several scales can be applied, like Eisenberg hydrophobicity consensus aminoacid scale, aminoacid side chain flexibilitiy scale, GRAVY hydrophobicity aminoacid scale, aminoacid side chain flexibilitiy scale, aminoacid polarity scale, amino acid transmembrane propensity scale and several others [93].

4.1.9 *Binary profiles*

In binary profiles, the peptide segment is presented as binary numerical numbers. In this features it is necessary to take into account that the length of the feature produced depends on the length of the given sequence and therefore, sometimes may be recommended to input sequences with equal length.

Considering aa composition, each aa is represented by a vector with 21 numerical values (20 units for 20 aas and a dummy variable if necessary adding non-natural aa to the sequence). For example A is presented by the vector (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) and C by (0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0).

Other way to implement a binary profile is to consider residue properties profile. For example, considering a property-profile for positive charge residues, an aa will be presented by ”1” if it is positive charge otherwise ”0”. It is possible to create property profiles for 25 type of physicochemical properties like hydrophobicity, hydrophilicity or polarity. For example, hydrophobicity profile for amino acid sequence DPARAAGAHQ” will be (0,1,1,0,1,1,0,1,0,0) as amino acid ”P” and ”A” are hydrophobic [94].

Binary profiles have been widely used for residue level annotation that includes prediction of protein’s secondary structure as well as nucleotide or ligand binding sites in proteins [95, 96].

In order to calculate the features of a protein sequence various python packages are available such as *Propy* [3], *PyDPI* [81], *PROFEAT* [76, 97], *ProFET* [98], *modlAMP* [93], *BioSeq-Analysis* [14], *PyBioMed* [83], *pfeature* [94], *PyFeat* [99], *iFeature* [100] or *iLearn* [101].

4.2 RELEVANT PREVIOUS WORK ON PEPTIDE CLASSIFICATION

This section intends to briefly describe the already existing packages for peptide classification and to address the work already done for the defined case studies, this is: antimicrobial, anticancer and viral fusion peptides or general peptide classification.

4.2.1 *Packages and tools for peptide classification*

Several web servers and tools have been developed to analyse biological sequences. However, many of them focus on only one of the steps of the machine learning pipeline (feature extraction, predictor construction or performance evaluation). Besides that, at the time of the beginning of this project, there were few packages available to calculate features of proteins developed in python 3 and to perform machine learning with protein related problems.

One of the main tasks of peptide machine learning is the calculus of features. Here, the best known packages like *Propy* [3], *PyDPI* [81] or *PyBioMed* [83] are developed in python 2, which makes them less usable nowadays. *pfeature*, is another package, developed in April 2019 in python 3, that allows to compute protein features working on the command line. These packages only do the calculus of features.

modlAMP is a Python-based software package for the design, classification and visual representation of peptide data. It offers functions for molecular descriptor calculation and the retrieval of amino acid sequences from public or local sequence databases [93].

Packages like *ProFET* [98], *PyFeat* [99], *modlAMP* [93], *PROFEAT* [102], *BioSeq-Analysis* [14], *IFeature* [100] and *iLearn* [101] are packages that calculate features and have functions to perform machine learning. *ProFET*, *PyFeat* and *modlAMP* have significantly less functions to calculate features and to perform other machine learning tasks. *PROFEAT* computes a large number of descriptors but is more focused on network descriptors. *BioSeq-Analysis*, a web server (available as stand-alone program) that automatically does feature extraction, predictor construction and performance evaluation only needing the upload of the benchmark dataset for analysis of DNA, RNA and protein sequences [14], does not allow the user to have full control over all the steps of the protocol and to adapt the code to its specific problem, having less choice to the calculus of features and data treatment and analysis. *iLearn* is the package more complete, having functions to calculate features, perform clustering, feature selection and dimensionality reduction for DNA, RNA and protein sequences [101]. *iLearn* works by command line.

In a general context, user friendly machine learning with proteins programs are scarce. There is the need for packages that agglomerate all the main steps to construct a predictor and are easy to use. Besides that, the tools and web servers available do not allow the user to have full control over all the steps of the protocol and cannot adapt the code to its specific

problem. Nonetheless, there are no available tools specifically focused in analysing and/or distinguishing membrane interacting peptides.

4.2.2 *Previous work on membrane-active peptides*

In the subsections below are summarized some of the tools or web servers that yield high performance metrics on protein classification problems using machine learning, specifically on antimicrobial, anticancer and antiviral peptides.

Antimicrobial peptides

ML approaches have been previously applied to differentiate AMPs from non-AMPs [103].

The *AntiBP2* webserver classifies antibacterial peptides based on SVMs using amino acid composition of the whole peptide, and was developed in 2010 [104]. In 2013, the Multilevel classifier *iAMP-2L* was designed to predict AMPs and their activities although the web server is no longer available. It was based on the PAAC and fuzzy K-nearest neighbor algorithm, where the components of PAAC were featured by incorporating physicochemical properties [105].

In 2016, an SVM-based classifier to investigate α-helical AMPs and the interrelated nature of their functional commonality and sequence homology was developed. Several descriptors (from AAC, PAAC, CTD, QSO and others) were used to distinguish membrane-active sequences from non–membrane-active sequences. In the end, the method showed to be useful as a general detector of membrane activity in peptide sequences and found an unexpectedly diverse taxonomy of sequences that are just as membrane-active as known AMPs, including endogenous neuropeptides, viral fusion proteins, topogenic peptides, and amyloids [6].

In contrast to the previous tools, which predict AMPs based on physicochemical features, *CAMPSign* predicts AMPs based on the presence of conserved AMP family-specific sequence composition. *CAMPSign* is a webserver developed in 2016 that allows the identification of antimicrobial peptides belonging to one of the 45 major AMP families present in CAMP database. It can perform RF, SVM, ANN and *discriminant analysis (DA)*. The tool utilizes family-specific signatures captured using patterns and HMMs for identification of peptides belonging to a particular AMP family [106].

In 2017, an SVM-based AMP classifer, *iAMPpred*, was created. The approach is based on SVM with compositional (AAC, normalized AAC, PAAC), physicochemical and structural features of the peptides and is able to predict its activity as antibacterial, antifungal, or antiviral [107].

In 2018, the method *AmPEP* was proposed. It uses the RF algorithm and is based on the distribution of aminoacid properties along the sequence. The feature set is composed of 105

distribution descriptors (D from CTD features) covering seven physicochemical properties of peptides (hydrophobicity, normalized van der Waals volume, polarity, polarizability, charge, secondary structure, and solvent accessibility) but is further reduced to obtain a minimal feature set of 23 features. This method showed to outperform both *iAMP-2L* and *iAMPpred* [24]. Also in 2018, a deep learning approach was used to build a model to recognize antimicrobial peptides obtaining good results [108].

Anticancer peptides

In order to predict anticancer peptides, several computational approaches have been used. The *AntiCP* developed in 2013 is a web server for the design of anticancer peptides that employs SVM models based on aminoacid composition (AAC and DPC) and binary profile features (only one at time) [25].

In 2014, Hajisharifi et al., proposed a model to identify anticancer peptides, based on local alignment kernel by using an SVM and PAAC as features [11].

iACP is a sequence-based predictor available as web server developed by the approach of optimizing the g-gap dipeptide components and using SVM [9]. Li and Wang in the same year, 2016, developed an improved predictor to identify the anticancer peptides using as features a hybrid composition of the AAC, *average chemical shifts (acACS)* and the RAAC using an SVM [75].

One year later, *iACP-GAEnsC* was developed. *iACP-GAEnsC* is a tool that uses a genetic algorithm-based ensemble model for the identification of anticancer peptides. The features used are the APAAC, g-Gap DPC, and RAAC. It uses an ensemble model formed by combining SVMs, *Probabilistic Neutral network (PNN)*, RF, *Generalized regression neural network (GRNN)* and KNN [8].

MLACP, developed also in 2017, is a web server that employs SVM and RF algorithms and uses a combination of all composition- and property-based features as inputs AAC, DPC, *atomic composition (ATC)* and physicochemical properties being one of the first method using a combination of features and describing the use of physicochemical and ATC features for ACP prediction and demonstrated to outperform both *antiCP* and *iACP* in accuracy [12].

Khan et al, developed a model to predict anticancer peptides based on two different classifiers: SVM and KNN and used as features the *Split amino acid composition (SAAC)*, DPC and PAAC [80].

In 2018, a sequence-based model for identifying ACPs (*SAP*) was proposed. In *SAP*, the peptide is represented by adjoining or g-gap dipeptide features, and then the unrelated features are pruned using the maximum relevance-maximum distance method [7]. There are also works employing deep learning pipelines regarding anticancer peptides [109, 13].

The majority of anticancer predicting methods showed high accuracies, however there is still room to improve the existing methods. The majority uses only the ZOH benchmark

dataset which does not have experimentally verified non-anticancer peptides. Furthermore, the role of different types of features is not yet understood as there are very few models that took advantage of feature selection methods to rank the importance and contributions of the features. Besides that, not all of these models are publicly available as web servers or tools which limits the access to the model's prediction capability [21].

Viral fusion peptides

Models to predict viral fusion peptides are scarce and there is no general ML-based method that can be applied to different viral proteins. Nonetheless, there is an SVM model using sequence-based statistical scores of self-derived peptide inhibitors as input features to correlate with their activities. The predictive model is able to predict peptides of envelope proteins and would be useful in development of antiviral peptide inhibitors targeting the virus fusion process [110].

All aforementioned tools and Web servers have been used in peptide classification, however further work is necessary to improve these classifiers. It is also visible that there is lack of models that can accurately predict viral fusion peptides or methods that can distinguish between several types of membrane peptides. Furthermore, although there are some deep learning tools available for DNA, RNA and protein classification, not many were found for classification of membrane peptides, despite the good results that seem to reveal [108], it may be other possible direction for further works. A summary of the above mention methods can be seen in Table 2.

Table 2: Summary of methods available for antimicrobial and anticancer classification

Peptide class	Name	Features used	ML algorithm	Tool/web server	Ref.	Date
Antimicrobial peptide	AntiBP2	AAC	SVM	web server	[104]	2010
	iAMP-2L	PAAC	Fuzzy KNN	webserver (not available)	[105]	2013
	Lee et al.	several	SVM	model	[6]	2016
	CAMPSign	AMP family-specific signatures	RF, SVM ANN or DA	web server	[106]	2016

Peptide class	Name	Features used	ML algorithm	Tool/web server	Ref.	Date
	iAMPpred	AAC, PAAC, NAAC, physico-chemical, structural	SVM	web server	[107]	2017
	AmPEP	D (from CTD)	RF	tool	[24]	2018
	Vetri et al.	model automatically extracted	deep learning	model	[108]	2018
Anticancer peptide	AntiCP	AAC, DPC, Binary composition	SVM	web server	[25]	2013
	Hajisharifi et al	PAAC	SVM	model	[11]	2014
	iACP	g-Gap DPC	SVM	webserver	[9]	2016
	Li and Wang	AAC, acACS and the RAAC	SVM	tool	[8]	2017
	iACP-GAEnsC	APAAC, g-Gap DPC and RAAC	Ensemble SVM, PNN, RF, GRNN and KNN	web server	[107]	2017
	MLACP	AAC, DPC, ATC, Physico-chemical	SVM and RF	webserver	[12]	2017
	Khan et al	SAAC, DPC and PAAC	SVM, KNN	model	[80]	2017
	SAP	DPC or g-gap DPC	SVM	model	[7]	2018

5

DEVELOPMENT

This section will present the project pipeline and the description of the methods used to develop the tool to perform machine learning methods in protein classification problems. This general-purpose Python package can be used in different problems involving the classification of peptides/proteins based on their physicochemical properties.

All the code developed for this thesis was written in python 3.6 using the *Anaconda Data science* platform and *Pycharm professional 2019.1* as IDE. The most important libraries used were *Scikit-learn*, *NumPy*, *SciPy* and *pandas*. Additionally, the packages of *modlAMP*, *PyDPI*, *pfeature* and *Biopython* were used.

5.1 DEVELOPMENT OF THE PYTHON PACKAGE

The package was built in a modular way, allowing users to have control over the different steps and to adapt/extend the code to fit their specific needs. In order to use the modules, it is necessary to create a class object and call the desired methods from each of the modules. The user can set specific values for the majority of the parameters, but default values are established.

The package is composed by the following modules:

1. **Read sequence module**: to read and/or change sequences or generate subsequences;
2. **Descriptors module**: to compute different types of protein descriptors;
3. **Preprocessing module**: to do a preprocessing of the feature vectors and 'clean' the dataset;
4. **Feature reduction module**: to reduce the number of features based on the unsupervised technique PCA;
5. **Feature selection module**: to discover and select the most valuable features;
6. **Cluster module**: to perform and plot the results of a clustering analysis;

7. **Machine learning module**: to implement supervised machine learning algorithms.

Additionally, the package contains a test module to validate if all the functions are working properly and an example file of a possible implementation of a pipeline using the package. A schematic view of the package can be seen in Figure 5.

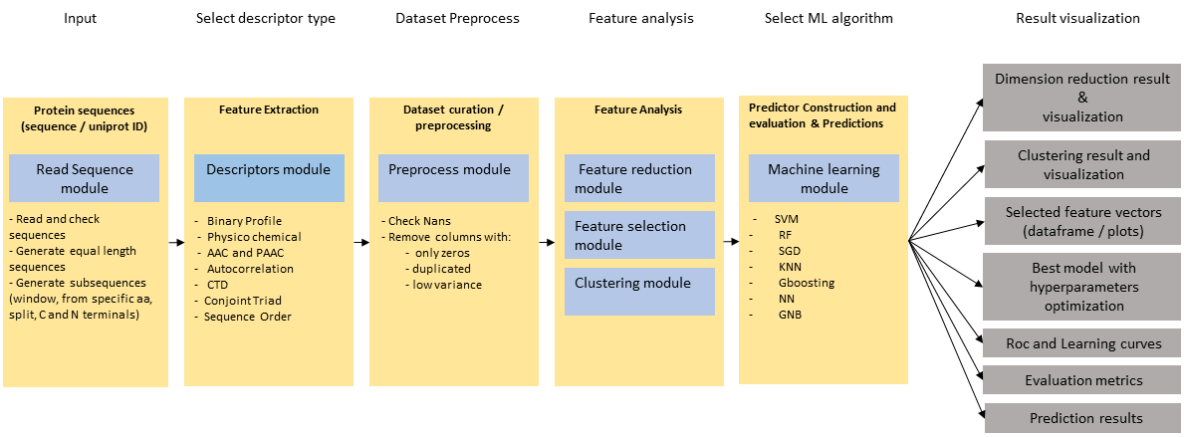


Figure 5: Schematic representation of the modules in the built package

For the creation of the class ReadSequence and Descriptor, functions from the *PyDPI* package were used. This package allows the download of protein sequences from UniProt, of properties from AAindex database, to calculate protein descriptors individually or to create a protein object [81]. As *PyDPI* is written in python 2.7, the first thing was to convert files related to reading sequences and calculating of features of proteins to python 3 using the *to3* library.

5.1.1 *Read sequence module*

The read sequence module contains the ReadSequence class that is used to read or change sequences and obtain subsequences. The class allows to:

- Read sequences from string or from an Uniprot ID (it is also possible to retrieve sequences from text with UniProt IDs) retrieving sequence objects used to calculate descriptors (following module);
- Check if it is a valid aa sequence;
- Obtain sequences with a given size from a list of sequences, adding or cutting from both the N and C terminals. This may be specially relevant to calculate features that are length sequence dependent;

- From one sequence, generate a list of sub-sequences based on a sliding window approach, from specific aa, from the terminals or dividing the sequence in parts. Beginning with only one sequence it will generate a list of sub sequences. A sliding window approach can be particularly helpful in screening sites problems, the division by terminals and from specific AA are most used in biological approaches.

A summary table is show below (Table 3).

Table 3: Summary table of the methods available in the class ReadSequence

Class ReadSequence	**Aim**	**Method**
Read sequences	Read a protein sequence	*ReadProteinSequence()*
	Downloading a protein sequence by uniprot id	*GetProteinSequenceFromID()*
	retrieve sequences from a txt with uniprot ID	*GetProteinSequenceFromTxt()*
Check protein	Check if sequence is a valid aminoacid sequence	*Checkprotein()*
Equal length	Cut or add aa to obtain sequences with equal length	*GetSizedSeq()*
Generate sub seqs ($1 \rightarrow n$ sequences)	Sliding window of the protein given. It will generate a list of n sequences with length equal to the value of window and spaced by a gap value	*Get_sub_seq_sliding_window()*
	Get all 2*window+1 sub-sequences whose center is ToAA in a protein	*Get_sub_seq_to_aa()*
	Split the original seq in n number of subsequences	*Get_sub_seq_split()*
	Divide the sequence in the N terminal and C terminal with sizes defined by the user	*Get_sub_seq_terminals()*

5.1.2 *Descriptors module*

The descriptors module aims to compute different types of protein descriptors and is carried out with the class named Descriptor. The descriptors functions are retrieved from the packages *Biopython*, *modlAMP*, *pfeature* and *PyDPI*. The class takes as input a protein sequence (from the previous model) that is used to calculate a variety of protein descriptors.

PyDPI, besides reading sequences, allows to download properties from AAindex database and calculate protein descriptors individually or creating a protein object [81]. *modlAMP* is a

package to work with peptides, proteins or any sequence of natural aminoacids that allows to generate sequences, calculate descriptor values (different from the ones from *PyDPI*), plot features, feature scaling, analysis and machine learning [93]. *pfeature* is a software package that computes protein features, such as composition based features, binary profiles, evolutionary information, structure based features and patterns [111]. *Biopython* is a set of freely available tools for biological computation, it allows to calculate some protein features with the *ProtParam* module [112].

The descriptors are divided into categories and it is possible to calculate the descriptors with the individual functions, all the functions in same category or calculate all the descriptors available by calling the *GetAll* function. If the user desires to choose the features, the function *adaptable* receives as input a list containing the numbers of the desired descriptors to be calculated.

This Descriptor class allows the calculation of:

- Binary profiles for both sequence and for 25 physicochemical properties of each residue of the sequence;
- Physicochemical descriptors. It has 16 features available that include length, charge, charge density, formula (number of C,H,N,O and S in the sequence), number of 4 types of molecular bonds (total, single,double and hydrogen bonds), the molecular weight, GRAVY, aromacity, isoelectric point, instability index, values for secondary structures (a-helix, turns and b-sheets), molar extinction coefficient, flexibility, aliphatic index, boman index and hydrophobic ratio;
- Aminoacid composition functions. It retrieves the aminoacid composition, the Dipeptide composition, and the tripeptide composition;
- Pseudo aminoacid composition and the amphiphilic aminoacid composition;
- Autocorrelation descriptors including Normalized Moreau-Broto autocorrelation, Moran Autocorrelation and Geary autocorrelation values;
- Composition, Transition and Distribution descriptors;
- Conjoint Triad descriptors;
- Sequence order descriptors. It calculates sequence order coupling numbers and quasi sequence order values;
- Base class peptide descriptors. It allows to calculate the sequence moment, the global averaging descriptor, hydrophocity moments, arcs, autocorrelation and cross correlation of amino acid values for a given descriptor value.

Overall, 38 descriptor and 8 agglomerative functions are available. A summary table can be seen below (Table 4).

Table 4: Summary table of the functions available in the module Descriptor. K means the length of the sequence

Class Descriptor	**N°**	**Aim**	**Method**	**N°Descriptors**
Binary Profile	1	Binary profile of aminoacid composition	*Getbin_aa()*	20*k
	2	Binary profile of residues for 25 phychem feature	*Getbin_resi_prop()*	Max 25*k
Physico chemical	3	Length of sequence	*Getlength()*	1
	4	Charge of sequence	*GetCharge()*	1
	5	Charge density of sequence	*GetChargeDensity()*	1
	6	Number of C,H,N,O and S of the aa of sequence	*GetFormula()*	5
	7	Sum of the bond composition for each type of bond	*GetBond()*	4
	8	Molecular weight	*GetMW()*	1
	9	Gravy from sequence	*GetGravy()*	1
	10	Aromacity	*GetAromacity()*	1
	11	Isolectric Point	*GetIsoelectricPoint()*	1
	12	Instability index from sequence	*GetInstabilityIndex()*	1
	13	Fraction of aa which tend to be in helix, turn or sheet	*GetSecStruct()*	3
	14	Molar extinction coefficient	*GetMolarExtinction Coefficient()*	2
	15	Flexibility according to Vihinen, 1994	*Getflexibility()*	-
	16	Aliphatic index of sequence	*GetAliphatic_index()*	1
	17	Boman index of sequence	*GetBoman_index()*	1

Class Descriptor	Nº	Aim	Method	NºDescriptors
	18	Hydrophobic ratio of sequence	*GetHydrophobic_ratio()*	1
	19	All 15 geral descriptors	*GetAllphysicochemical()*	-
Aminoacid Composition	20	Aminoacid compositon	*GetAAComp()*	20
	21	Dipeptide composition	*GetDPComp()*	400
	22	Tripeptide composition	*GetTPComp()*	8000
	23	All descriptors from Aminoacid Composition	*GetAllAAC()*	8420
Pseudo aminoacid Composition	24	Type I Pseudo aminoacid composition	*GetPAAC()*	Min 30, depends lambda
	25	Type I Pseudo aminoacid composition for a given property	*GetPAACp()*	Min 30, depends lambda
	26	Type II Pseudo aminoacid composition - Amphiphilic	*GetAPAAC()*	Min 30, depends lambda
	27	Calculate PAAC and APAAC	*GetAllPAAC()*	Min 60
Auto correlation	28	Normalized Moreau-Broto autocorrelation	*GetMoreauBrotoAuto()*	240
	29	Moran autocorrelation	*GetMoranAuto()*	240
	30	Geary autocorrelation	*GetGearyAuto()*	240
	31	Calculate all descriptors from Autocorrelation	*GetAllCorrelation()*	720
CTD	32	Composition Transition Distribution	*GetCTD()*	147
Conjoint Triad	33	Conjoint Triad	*GetConjT()*	343
Sequence Order	34	Sequence order coupling numbers	*GetSOCN()*	90 (default)
	35	Sequence order coupling numbers	*GetSOCNp()*	90 (default)

Class Descriptor	Nº	Aim	Method	NºDescriptors
	36	Quasi sequence order	*GetQSO()*	100 (default)
	37	Quasi sequence order	*GetQSOp()*	100 (default)
	38	Calculate all values for sequence order descriptors	*GetAllsequenceorder()*	190 (default)
	39	Moment of sequence	*calculate_moment()*	1
	40	Global / window averaging descriptor	*calculate_global()*	1
Base Class Peptide	41	Hydrophobicity or hydrophobic moment profiles	*calculate_profile()*	2
	42	Calculates arcs	*calculate_arc()*	5
	43	Autocorrelation of aa values for a given descriptor scale	*calculate_autocorr()*	-
	44	Cross correlation of aa values for a given descriptor scale	*calculate_crosscorr()*	-
	45	All functions from Base class	*GetAllBaseClass()*	-
All	46	Calculate all possible descriptors (except tripeptide)	*GetAll()*	-
Adaptable	47	Choose which functions to calculate	*adaptable()* (list_of_functions=[])	-

5.1.3 *Preprocessing module*

The preprocessing module allows the transformation of the feature vectors into representations suitable for downstream estimators. By 'cleaning' the dataset from redundant features, it removes a large number of non relevant columns from the dataset. It is callable by the class Preprocess and allows for the:

- Elimination of columns with only zero values;

- Elimination of repeated columns;
- Elimination of low variance columns (by default zero variance, but the user can adapt, this function has an imbued scaler).

It is possible to call a single function *preprocess* to run all the functions above. The functions names and aims can be seen in Table 5.

Table 5: Summary table of the methods available in the class Preprocess

Class Preprocess	**Aim**	**Method**
Check data	Check nans	*missing_data()*
Remove low relevance columns	Remove columns that have all values as zero	*remove_columns_all_zeros()*
	Remove duplicated columns	*remove_duplicate_columns()*
	Remove all features whose variance does not meet some threshold	*remove_low_variance()*
	Remove columns that have all values as zero, duplicated and low variance columns	*preprocess()*

5.1.4 *Feature reduction module*

In ML, feature reduction techniques allow to summarize the essential characteristics of a high dimensional data representation reducing the number of features [39]. This way, the class Feature_reduction enables to reduce the number of features on a dataset based on the unsupervised technique PCA. PCA is a statistical procedure that orthogonally transforms the original coordinate system of a (numerical) data set into a new set of coordinates, the principal components, that capture well the variance of original features. For better PCA analysis and perception, this module enables the production of two plots. The main purposes of this class are to:

- Derive the PCA of the data. The function receives a dataset, scales it (using standard scaler as default) and applies the *Scikit-learn* PCA algorithm retrieving the fitted and the transformed dataset [1];

- Access the contribution of each feature for the component and the variance ratio of each component;
- Produce a bar plot with percentage of explained variance ratio by components;
- Produce a scatter plot that represents the different classes based on two *Principal component (PC)*s (by default the first two). As the PCA components are orthogonal to each other (not correlated), this usually allows to distinguish classes in a clear way.

It is worth noting that PCA is an unsupervised method, meaning that PCA components are calculated only from features and no information from classes are considered. Besides this, each component does not represent a feature but a mixture of the original features.

A summary table of the functions that can be implemented with this class is given below (Table 6).

Table 6: Summary table of the methods available in the class Feature_reduction

Class Feature_reduction	**Aim**	**Method**
PCA	Perform the PCA analysis	*PCA()*
	Measure the variance ratio of the principal components	*variance_ratio_components()*
	Retrieve a dataframe containing the contribution of each feature (rows) for component	*contribution_of_features_to_components()*
Design Graphics	Derive a bar plot representing the percentage of explained variance ratio by PCA	*pca_bar_plot()*
	Scatter plot of the labels based on two components (by default the first ones)	*pca_scatter_plot()*

5.1.5 *Feature selection module*

Feature selection is another way to reduce the dimensionality of the dataset by selecting the most valuable features to improve estimators' accuracy and/or boost their performance on very high dimensional datasets [52]. This selection allows to gain knowledge and understanding of the data, quite important in biological questions [55].

There are three major methods to select features - filter methods, wrapper methods and embedded methods [60] and therefore, this package has methods to perform univariate feature selection, *recursive feature elimination (RFE)* and selection of features based on model like a tree or SVM. The package class Feature_selection is based on *Scikit-learn* functions from the *feature.selection* module.

This module allows to:

- Perform univariate feature selection. The function has a configurable strategy and the user will decide which score function to use and how the features will be selected exactly like in *Scikit-learn*. It is possible to select the k highest scoring features, specified percentage of features and features based on false positive rate, false discovery rate or family wise error. The scoring functions (return scoring values and p-values) include *chi2*, *f_classif* or *mutual_info_classif*;
- Perform recursive feature elimination with or without cross validation. The user can decide the number of features to select;
- Retrieve scores from recursive feature elimination rankings;
- Select features based on embedded techniques using user selected estimators (e.g. TreeClassifier, logistic regression and linear SVC);
- Obtain the features names and scores of importance for univariate and embedded techniques, retrieving a dataframe with features names and scores of univariate tests or select from model by order of relevance.

Univariate feature selection selects the best features (high correlation with outcome variable) based on univariate statistical tests and can be used as preprocessing step. This test is only reliable if the variables are fully independent. They are independent of any machine learning algorithm, with the features being selected solely based on their correlation with the outcome variable. It is important to take into account that the methods based on F-test estimate the degree of linear dependency between two random variables whereas, mutual information methods can capture any kind of statistical dependency, but being non parametric, they require more samples for accurate estimation. *Chi2* will only work for non negative features. If the parameters are not given the function will calculate based on the scoring function *mutual_info_classif* and selecting the best 10^{-5} percentile of features. It returns the dataset fit and transformed, the new dataset and the columns selected. The higher the score the more representative is the feature.

Wrapper methods use a subset of features to train and assess the performance of a model, being the subset of features tailored to this model. RFE uses an external supervised learning estimator that provides information about feature importance (a coef_ attribute or a features_importances attribute), such as SVM; from the initial set of features, the estimator

assigns weights to each feature and eliminates the ones with smallest weights, repeating the process until the desired number of features is achieved. *The recursive_feature_elimination* function allows to recursively eliminate features less important with or without cross validation. The user can decide the number of features to select. The function retrieves the dataset fit and transformed, the original dataset with the features selected, the columns names and the features ranking. Using the function *rfe_ranking* a dataframe containing the features names and the ranking by order is obtained. Rankings of 1 corresponds to the ones selected.

It is also possible to select features based on embedded techniques. Here, the optimal set of features is built into the classifier construction as the estimator retrieves a *coef_* or a feature_importances vector. The features considered unimportant, below a threshold, are removed. The threshold can be a 'median', 'mean', floats of the strings or any numerically valuable. It is possible to do this in two ways:

- Considering linear models penalized with the L1 norm for classification, such as logistic regression and linear *Support Vector Classification (SVC)*. They retrieve sparse solutions and negative values. The data should not be very noisy and the features should be independent. The most important features have the highest scores and features uncorrelated with the output variable should have coefficients close to zero. If the correlation coefficient is negative, it provides statistical evidence of a negative relationship between the variables. The increase in the first variable will cause the decrease in the second variable.

- Considering tree based estimators that compute features importance and can be used to discard irrelevant features.

A view of the methods available in this class can be seen in the table below (Table 7).

Table 7: Summary table of the methods available in the class Feature_selection

Class Feature_selection	Aim	Method
Feature selection	Univariate feature selector	*univariate()*
	Recursive feature elimination	*recursive_feature_elimination()*
	Select from model	*select_from_model_feature_elimination()*
Retrieve features dataframes	Retrieve a dataframe with features names and scores of importance resulting of the univariate tests and from model selection	*features_scores()*
	Retrieve a dataframe with features names and its ranking position ordered	*rfe_ranking()*

5.1.6 *Clustering module*

The class Cluster aims to perform and plot clustering analysis. Clustering is a type of unsupervised machine learning useful to find homogenous subgroups, such that objects in the same group (cluster) are more similar to each other than with others [19]. If the labels are already available, clustering analysis can be useful to see how the samples are grouped and if they match indeed the labels defined. Besides that, clusters can also be used as features in a supervised machine learning model.

When the class is initialized, a dataset corresponding to the X dataset (input dataset without the target column) and the target column must be provided. Additionally, the test size to split the data into training and test dataset must be provided (established by default as 0.3).

This class implements methods for:

- Performing K means classification;
- Performing hierarchical clustering;
- Applying the K-means algorithm to train data and predict the test set. These labels can be added as a feature or can replace the previous labels in the dataset. It is possible to see how they perform with the function *classify* [3].

The k-means algorithm clusters data by dividing observations into k groups of equal variance - clusters. It can be easily used in classification where we divide data into clusters which can be equal to or more than the number of classes. It scales well to large number of samples and is one of the most widely used clustering algorithms. The algorithm divides the samples into clusters, each described by the mean of the samples of the cluster – centroids. Initially, the algorithm chooses the initial centroids randomly; then, it will loop between assigning each sample to its nearest centroid and changing the centroids by taking the mean value of all the samples assigned to each previous centroid, repeating until the centroids do not move. The function *Kmeans* applies the Kmeans algorithm to the dataset. It is possible to edit the maximum number of iterations (300 per default) and the number of clusters (by default the number of existing labels). The method implements the k-means initialization scheme to address the primary choice of the centroids distant from each other leading to better results.

The class has functions to visualize data with *hierarchical clustering (HC)* as well. Hierarchical clustering builds nested clusters by merging or splitting them successively, with this hierarchy being represented as dendograms (trees). The root of the tree is a unique cluster with all the samples and the leaves are clusters with only one sample. The HC uses a bottom up approach where each observation starts in its own cluster and where clusters are successively merged together [4]. The algorithm stops when only one cluster remains, the root.

To build the hierarchical clusters, the *SciPy* library *cluster.hierarchy* was used. It is possible to calculate the distance between clusters using the method 'single' that minimizes the distance of all the pairs of clusters, the method 'complete' that maximizes that distance or the method 'average' that represents the average of all distances. Other methods such as 'weighted', 'centroid', 'median', 'ward' are available as well [5]. The metric used to pairwise distances between observations in n-dimensional space can be chosen as well, taking values as 'correlation', 'euclidean', 'hamming' or others [6]. The function *hierarchical* takes as input the metric and distance parameters and returns the graphic of the hierarchical division of samples.

A view of the methods available in this class can be seen in the table below (Table 8).

Table 8: Summary table of the methods available in the module Cluster

Class Cluster	Aim	Method
Kmeans	Perform K means cluster	*Kmeans()*
	Perform the kmeans to train data and predict the test set. If add, the labels produced by clustering will be added as features. If replace, labels produced will replace the old labels	*Kmeans_predict()*
	Fit the model in train datasets and predict on the test dataset, returning the accuracy	*classify()*
Hierarchical	Perform hierarchical clustering	*hierarchical()*

5.1.7 *Machine learning module*

In order to implement supervised machine learning algorithms, the class Machine_learning was created. The machine learning module intends to facilitate the application of ML models to the classification of peptides. It is possible to do model selection using grid search for hyperparameters tuning and selecting the best model, do model evaluation, return feature importance (as dataframe or bar plot), and plot validation and learning curves. The user must create an object Machine_learning giving the X dataset (inputs) and the column of labels and the size of the test set. The function will load and split the data.

This class allows to:

- Perform a grid search on different model parameters, returning the best fitted model. As models, it is possible to choose between RF, GB, SVM, KNN, *stochastic gradient descendent (SGD)*, *gaussian naive bayes (GNB)* and ANN;
- Plot a validation curve for the specified classifier on any parameter in grid search;
- Plot Learning curves;
- Retrieve test set scores for the specified scoring metrics allowing for an evaluation of the performance of the model through test-set prediction. It retrieves the values for MCC, accuracy, precision, recall, f1, ROC - AUC, TN, FP, FN, TP, *False discovery rate (FDR)*, sensitivity and specificity derived from metrics *Scikit-learn* module;

- Plot a ROC curve;
- Retrieve the features importance of the final model for classifiers SVM, RF, GB, SGD. It retrieves a dataframe and draws a bar plot. For both SVM and SGD the retrieved values are the coefficients of the features and for the RF it is possible to retrieve the values of feature importance;
- Make predictions for unseen sequences (ultimate goal for ML models). The function *predict* can be used to predict novel peptides with a trained classifier model returning a dataframe with predictions using the specified estimator and test data. If the true class is given, the scoring value for the test data is provided;
- Scan a protein using a sliding window approach to predict sites with a trained classifier model. It will return a dataframe with the subsequences generated, positions in the original sequence, probability of belonging to the class and a class probability if the probability is bigger than 0.99, 0.95, 0.9, 0.8, 0.7, 0.6 or predicted as a negative.

In the search for the best model, it is possible to adjust several parameters. The pipeline includes the use of a scaler, *standardscaler* by default, but the user, can choose any other from *Scikit-learn* such as *normalizer*, *minmaxscale* or None. The score used to find the best model in the grid search can be chosen by the user, being by default the MCC. The MCC is often used in machine learning to measure the quality of binary classifications, being a balanced measure that can be used even if the classes are of different sizes. The values are between 1, a perfect prediction, and -1, total disagreement between predictions and observations. Values of 0 are considered to be no better than random predictions. Sample weights for training data, number of parallel jobs (by default -1) and number of cross validation folds (by default 10) can also be tuned by the user. In all the models available, the parameters can be tuned, however, if none is given, a default parameter grid is set for each model.

A learning curve determines cross validated training and test scores for different training set sizes, being useful to determine if it is beneficial to add more training data and whether the estimator suffers more from a variance error or a bias error (for example: if both validation and training cross converge to a value that is low when increasing the training dataset, the model will not benefit from the addition of more samples).

A cross validation generator splits the dataset k times in training and test data, a score for each training subset and test will be computed and the scores will be averaged over all k runs for each training subset size. The function receives the estimator, title for the graph, train sizes to test and cross validation parameters and retrieves the plot learning curve for that model, numbers of training examples, scores on training sets and scores on test set. It is based on *Scikit-learn* functions.

The ROC curve is a probability curve and is a good way to see how much the model is capable of distinguishing between classes. The higher the AUC, i.e., close to 1, the better the

model. The X axis corresponds to the false positive rate and the Y axis to the true positive rate. A larger area under the curve, and the plot on top left corner are ideal. The function *plot_roc_curve* receives a classifier, and test sets and automatically retrieves the plot.

A view of the methods available in this class can be seen in the table 9.

Table 9: Summary table of the methods available in the class Machine_learning

Class Machine_learning	**Aim**	**Methods**
Build model	Parameter grid search on a selected classifier model and peptide training data set	*train_best_model()*
	Plots a cross-validation curve for the specified classifier on all tested parameters given in the option 'param range'	*plot_validation_curve()*
	Test set scores for the specified scoring metrics in a pandas.DataFrame	*score_testset()*
	Function to plot a ROC curve	*plot_roc_curve()*
	Function that given a classifier retrieves the features importances as a dataset and represent as barplot	*features_importances()*
	Plot a learning curve to determine cross validated training and test scores for different training set sizes	*plot_learning_curve()*
Predict	Predict novel peptides with a trained classifier model	*predict()*
	Scan a protein in a sliding window approach to predict novel peptides with a trained classifier model	*predict_window()*

5.1.8 *Other functions*

Besides these models previously described, the package contains a test directory that contains a python file to test all the functions of the package.

A file *Scores.py* (in the directory of the feature selection and reduction) allows to calculate scores for SVM, SVC, RF and GNB comparing two given X datasets and labels considering a

10 fold cross validation. This function can be used as a simple way to compare the results of two datasets for example, when testing methods of feature selection.

5.2 Outcomes and discussion of the package developed

The package built during the development of this thesis facilitates the major tasks of machine learning and it includes modules to read and alter sequences, to calculate protein features, perform dataset preprocessing, feature reduction and selection, execute clustering algorithms and build machine learning models and make predictions. The package is directed to handle protein related problems, but its modular construction allow users to use it in other problems. The same way, as it is built in a modular way, the user retains the power to manipulate the functions and to use other functions outside of the package.

One of the main tasks of this tool is the calculation of features. Before this project, there were few packages available to calculate features of proteins developed in python 3 and to perform machine learning with protein related proteins.

Indeed, the best known packages like *Propy*, *PyDPI* or *PyBioMed* are developed in python 2, which makes them less usable. *pfeature* is another package, developed in April 2019 in python 3, that allows to compute protein features working on the command line. The majority of the features presented on *pfeature* are included in the package here described, not all were integrated due to its complexity or because they would increase exponentially and unnecessarily the number of features. Interesting features presented in *pfeature* or *iLearn* that are not contained within this package include AAindex schemes, *position-specific weight matrix (PSSM)* and *Blocks of Amino Acid Substitution Matrix (BLOSUM)* encoding schemes, as the protein secondary structures.

Most of the packages above do not have functions besides the calculation of features. Packages like *ProFET* (2015), *PyFeat* (2019) , *modlAMP* (2017), *BioSeq-Analysis* (2017), *Ifeature* and *iLearn* (2019) calculate features and have functions to perform machine learning. *ProFET*, *PyFeat* and *modlAMP* have significantly less functions to calculate features and to perform other machine learning tasks. *PROFEAT* computes a large number of descriptors but is more focused on network descriptors. *BioSeq-Analysis* automatically does feature extraction, predictor construction and performance evaluation only needing the upload of the benchmark dataset for analysis of DNA, RNA and protein sequences [14]. However, the user does not have full control over all the steps of the protocol and cannot adapt the code to its specific problem, having less choice to the calculation of features and data treatment and analysis. *iLearn* is the more complete package, having functions to calculate features, perform clustering, feature selection and dimensionality reduction for DNA, RNA and protein sequences. *iLearn* works by command line and offers different options than the package here developed.

Although it meets all the tasks it was built for, the package could be improved. Functions to accept FASTA files, to calculate directly N and C terminal features, to calculate PSSM or other functions including evolutionary and structure information could be interesting as features. Besides that, the prediction methods could include various classifiers at the same time, or allow ensemble models. A user friendly feature to add would be, the retrieval of a sequence image highlighted in the method *predict_window*; the user would not only see through the dataframe produced the potential positive locals, but would have a schematic view of the predicted sites in the sequence. The code itself could be improved in order to be more efficient. The package would also benefit if a Deep learning pipeline was included as no package until nowadays features it. The availability of a web server would also be a plus.

6

VALIDATION

This section will present a comparative analysis to demonstrate the application and performance of the developed package for addressing sequence-based prediction problems. The comparative analysis will be made with two test cases: antimicrobial peptides [24] and with anticancer peptides [12], both membrane active peptides.

6.1 ANTIMICROBIAL PEPTIDES - AMPEP

Antimicrobial peptides are promising candidates to fight multi-drug-resistant pathogens having a broad range of activities and low toxicity. The development of computational tools to predict AMP are crucial as the identification through wet lab experiments is still expensive and time consuming. To assess the predictive ability of our package to address sequence based prediction problems we tested against the *AmPEP* method which is described to highly perform on AMP prediction methods. In the publication, Bhadra et al., used a dataset with a positive:negative ratio (AMP/non-AMP) of 1:3, based on the distribution patterns of aa properties along the sequence (CTD features), with a 10 fold cross validation RF model. The collection of data with sets of AMP and non-AMP data is freely available at https://sourceforge.net/projects/axpep/files. Their model obtained a sensitivity of 0.95, a specificity and accuracy of 0.96, MCC of 0.9 and AUC-ROC of 0.98 [24].

Firstly, based on the available collection of data available of AMP and non AMP a dataset constituting of a 1:3 ratio was built. Taking this dataset as base, two datasets were assembled.

On the first one, CTD descriptors were calculated. A derived dataset was constructed restraining the features to the D feature. These two datasets were used to mimic the model published.

To understand if adding features would alter the performance of the model a second dataset was built. Physicochemical (15), AAC and DPC (420), CTD (147) and CTriad (343) descriptors were calculated. To reduce the number of features and select the more important ones, the dataset was scanned for invariable columns, and a univariate feature selector was used to reduce the number of features to 250 (mutual info classif used as function, selecting

the best k=250 features). This dataset was standardized. After, a L1 logistic regression model (C=0.01) was applied, being the final dataset made of 160 features.

Using these two datasets, RF models were built using the parameters of the article [24], with RF models performing grid search. Additionally, SVM models using grid search were also tested.

To mimic the model published, a RF model using the D from CTD descriptors with 105 estimators, the number of input variables tried at each split,*mtry*, defined as the square root of the number of features and a CV of 10 was built.

This model obtained a sensitivity of 0.91, a specificity of 0.93, accuracy of 0.96, MCC of 0.90 and AUC-ROC of 0.95 against the test set. With the same descriptors, but using a grid search approach, the model yielded similar results. Using all the CTD features a model was obtained with a sensitivity of 0.98, a specificity of 0.93, accuracy of 0.96, MCC of 0.90 and AUC-ROC of 0.95 yielding as described in the article slightly better results. Using the dataset with more features, the resulting model achieved similar results.

To test if a SVM model would outperform a RF based one, two models using the grid search approach were built. Using only the CTD features, the model obtained a sensitivity of 0.96, a specificity of 0.89, accuracy of 0.94, MCC of 0.86 and AUC-ROC of 0.93 against the test set, whereas using the dataset with more features the model obtained a sensitivity of 0.96, a specificity of 0.91, accuracy of 0.95, MCC of 0.87 and AUC-ROC of 0.94 against the test set.

A summary of these models and their performance is available on table 10.

The model mimicking the *AmPEP* predictive model yielded slightly different results, achieving more sensitivity but less specificity and having the same accuracy and MCC scores. This result shows that the package can be used to build models as it performs with similar performance to the best ones described in literature. Taking the results into account, it is also notorious that adding more sequence descriptors in RF models did not lead to better models whereas in SVM models more features led to better performance results. Both SVM models performed worse than any RF model, which was also in concordance with article that reports RF models performing better than SVM ones. The small differences observed when using the same model may be due to the methods used to perform RF or the scoring functions used to choose the best performance models. In the article, the authors did not specify which measure they took into account to select the best models. Here, to perform grid search, MCC score was used.

Table 10: Summary of the scores of the models produced with the package comparing to *AmPEP*

	Model	Parameters	Feature Set	Sens.	Specif.	Acc	MCC	AUC ROC
AmPEP	RF	AmPEP parameters	CTD_D (105)	0.95	0.96	0.96	0.90	0.98
Package developed	RF	AmPEP parameters	CTD_D (105)	0.91	0.93	0.96	0.90	0.95
	RF	Grid search	CTD (147) Features	0.98	0.93	0.96	0.90	0.95
	RF	AmPEP parameters	Calculus Selection (160)	0.97	0.93	0.96	0.90	0.95
	SVM	Grid Search	CTD (147) Features	0.96	0.89	0.94	0.86	0.93
	SVM	Grid search	Calculus Selection (160)	0.96	0.91	0.95	0.87	0.94

6.2 ANTICANCER PEPTIDES - MLACP

Anticancer peptides are promising drug candidates for cancer treatment. However, similarly to AMPs, identification of ACP through wet lab experiments is expensive and time consuming and, therefore, the development of an efficient computational method is essential to identify potential ACPs candidates prior to ***in vitro*** and ***in vivo*** experimentation.

To have a second comparative analysis with a different dataset, the package built was used to mimic the model developed by Manavalan et.al., *MLACP* [12]. In this publication, SVM and RF ML methods were developed. The features used to predict ACPs were calculated from the aminoacid sequence, including AAC, DPC, ATC, and from physicochemical properties. Tyagi-B datasets were used to train the models (free available at: http://www.thegleelab.org/MLACPData.html).

As the physicochemical properties defined in the article are different from the ones available in the package, the features AAC, DPC, ATC were considered to compare results. To this end, RF and SVM models were built using as features AAC, DPC and a hybrid model with the features AAC+DPC. A grid search with a cross validation of 10-fold was performed as described in the article. In SVM, a 'rbf' kernel was defined and the parameters C and γ optimized (parameter range: 0.001, 0.01, 0.1, 1). In RF models, the grid search contained the parameters: number of estimators (100, 300, 400, 500), number of maximum features ('sqrt'

or 2,3,5,7) and minimal number of sample splits [5, 6, 7, 8]. The parameters changed were the same as used in the article, however, and due to computational limitations, a significantly smaller range of parameters were tested. A comparative analysis of the performance of the models in *MLACP* article and with the package can be seen below (Table 11).

Table 11: Comparative analysis of the performance of RF and SVM models in *MLACP* article and with package.

Model		**RF**				**SVM**			
Features	**Package**	**MCC**	**Acc**	**Sens.**	**Specif.**	**MCC**	**Acc**	**Sens.**	**Specif.**
AAC	MLACP	0.66	0.85	0.70	0.93	0.69	0.87	0.70	0.95
	Developed	0.62	0.83	0.90	0.70	0.71	0.87	0.91	0.79
DPC	MLACP	0.65	0.85	0.70	0.92	0.64	0.85	0.60	0.97
	Developed	0.65	0.84	0.96	0.62	0.62	0.83	0.90	0.70
AAC+DPC	MLACP	0.67	0.86	0.70	0.93	0.66	0.86	0.61	0.97
	Developed	0.65	0.84	0.92	0.70	0.60	0.82	0.88	0.70

As shown in Table 11, the models constructed with the package developed have similar results to the ones reported in the original article. Bigger differences are found in sensitivity and specificity values, showing a different trade-off. MCC and Accuracy are very similar with the values from the package developed. Differences may be due to the diverse feature set, parameter ranges or the use of different packages. Similarly with *AmPEP* article, the authors did not specify which measure they took into account to select the best models. Here, to perform grid search, MCC score was again used.

Overall, this comparative analysis evidences the performance and validates the package here described.

7

VIRAL FUSION PEPTIDE CASE STUDY

After testing and validating the package on membrane active peptides classification problems, it was applied to the study of viral fusion peptides, also membrane active. This chapter will present the analysis pipeline and the description of the methods used to construct the models, the results obtained and the analysis and discussion of results both from a computational and biological point of view of viral fusion peptides.

7.1 METHODS

The methods used for dataset construction and preprocessing, feature generation and selection, machine learning models construction, optimization and evaluation and the application of these models to predict the location of a viral fusion peptide in a viral fusion protein are described below.

7.1.1 *Datasets for model construction*

The datasets used were developed by Sara Pereira in her bioinformatics dissertation [113].

A total of three datasets of viral fusion peptides differing in the negative cases were tested. All the three datasets were composed by 222 instances, being 111 of viral fusion peptides (positive dataset) and 111 of non viral fusion peptides sequences (negative dataset). The positive samples were labelled as 'vfp' whereas the negative samples were given the label 'non_vfp'.

For the construction of the positive dataset, only viral fusion peptide sequences with experimental evidence were considered.

In the first dataset, the negative cases, contain randomly generated sequences from fusion proteins (excluding the fusion peptide) having the same length as the corresponding fusion peptide.

In the second one, the negative cases contain transmembrane domains. This dataset was used in order to evaluate the ability of the model to distinguish between fusion peptides and *Transmembrane domain (TMD)*s, both the most hydrophobic regions of fusion proteins.

In the third, the negative dataset is the combination of the first two datasets containing half of the randomly generated sequences from fusion proteins and half of TMDs.

The dataset is composed primarily of sequences belonging to class I fusion proteins (82%), followed by class II (15%) and only 5% belong to class III. From the retrieved sequences, the majority of them belongs to the ***Retroviridae*** family, which includes HIV virus, followed by ***Flaviviridae*** (includes the Dengue and Zika viruses) and ***Paramyxoviridae*** (includes Parainfluenza and Hendra virus).

7.1.2 *Generation and selection of features*

For the three major datasets, datasets with different sets of features were constructed using the methods from the class Descriptor. One dataset contained all possible features (except binaries) which include 15 physicochemical descriptors, AAC, DPC and TPC, type I and II of PAAC, autocorrelation descriptors including Moran, Geary and normalized moreau broto autocorrelation, CTD, Conjoint triad, sequence and quasi sequence order and 6 base class peptide descriptors. Additionally, datasets containing only AAC, PAAC and APAAC and CTD were set up. The features were generated using the functions *GetAll* and *adaptable* from Descriptor class. These datasets were preprocessed to remove equal columns, columns with only zeros and non variant columns (class Preprocess of package described in previous section).

Furthermore, the dataset with all possible features was subject to various types of feature selection to select the most predictive subset of features and eliminate redundant or irrelevant features. To do this, methods from the class Feature_selection (*select_from_model_feature_elimination* and *univariate*) were employed.

Four approaches for feature selection were used. First, two embedded feature selection techniques were applied, namely a tree and a SVC model.

The SVC model followed the approach made by Lee et al. [78], where a linear SVC is built employing a L1 norm. First a grid search to determine a near-optimal value of C parameter was carried out. Linear models penalized with L1 norm have sparse solutions with many coefficients estimated equal to zero and with the relevant features having non zero coefficients. The parameter C controls the sparsity (smaller the C, fewer features selected).

To build the tree classifier, we first performed a grid search to determine a near-optimal value of number of estimators parameter. This parameter represents the number of trees in the forest, the larger the better but it may take longer to compute and, beyond a critical

number of trees results stop being significantly better. The parameter *max_features* was left as the default [52].

As the number of features remained high, two approaches were added combining both SVC and tree models with a previous univariate feature selection.

Univariate feature selection allows to remove descriptors with low correlation with the outcome variable based on univariate statistical tests and was performed with the function *univariate*. The best half of the features (percentile=50) using the *mutual_info_classif* method were selected. The *mutual_info_classif* method was preferred to *f_classif* (compute the ANOVA F-value for the provided sample, i.e, estimate the degree of linear dependency between two random variables scoring features individually) because, although it requires more samples for accurate estimation, it can capture any kind of statistical dependency [52].

The dataset with no feature selection was used to compare results and feature selection techniques.

Models with features AAC, CTD, PAAC and APAAC did not went through feature selection.

The selection and features importances scores were obtained with functions from the class Feature_selection of the package. For each function, as defined by default in the package functions, the datasets were standardized using the standard scaler, removing the variance and scaling to unit variance. Each function retrieves the original dataset with features selected.

This way, for all the three datasets a total of 8 datasets were generated containing only AAC, CTD and PAAC features and all features available in the package with no feature selection, SVC, tree or combination of univariate and SVC and tree feature selection models. In the table below (Table 12) we summarize the number of features for each dataset created.

Table 12: Description of the number of features used for each of the three datasets

Features	**F.selection**	**Dataset 1**	**Dataset 2**	**Dataset 3**
All Features	None	2212	2089	2175
	svc	78	57	122
	tree	569	404	397
	mutual_svc	76	58	105
	mutual_tree	278	158	748
AAC	None	20	20	20
CTD	None	135	135	148
PAAC	None	60	60	60

All the features are exclusively numerical.

7.1.3 *Machine learning models: construction, optimization and evaluation*

The next step was to build and run several machine learning models and evaluate them. We ran the same machine learning pipeline for all the 24 generated datasets.

After creating a class object Machine_learning with the data and labels, the dataset is divided into train and test set with a split ratio of 0.3. The function *train_best_model* was used to retrieve the best model from a hyperparameter optimization through grid search on the training dataset. This function then performs standard scaling and retrieves the best model using 10 fold cross validation. The evaluation of the models created was made using the function *score_testset* that scores Matthews correlation coefficient, accuracy, precision, recall, f1 and area under the ROC curve of the predictions made by the model on the test dataset. The models SVM, RF, SGD, GB, KNN, ANN and GNB were tested.

Table 13 describes the hyperparameter grid used for each of the seven machine learning models tested.

Table 13: Hyperparameter values used in grid search for the models: SVM, RF, KNN, GB, SGD and NN

Model	**Parameter grid**	
SVM	'clf__C': 'clf__kernel': 'clf__gamma':	[0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10.0] ['linear'] [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10.0]
RF	'clf__n_estimators': 'clf__max_features': 'clf__bootstrap': 'clf__criterion':	[10, 100, 500] ['sqrt', 'log2'] [True] ["gini"]
KNN	'clf__n_neighbors': 'clf__weights': 'clf__leaf_size':	[2, 5, 10, 15] ['uniform', 'distance'] [15, 30, 60]
GB	'clf__loss': 'clf__n_estimators': 'clf__max_depth':	['deviance', 'exponential'] [10, 100, 500] [1,3,5,10]
SGD	'clf__loss': 'clf__penalty': 'clf__alpha':	['hinge', 'log', 'modified_huber', 'perceptron'] ['l2', 'l1','elasticnet'] [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10.0]
GNB	'clf__var_smoothing':	[1e-12, 1e-9, 1e-4]
NN	'clf__activation': 'clf__batch_size':	['identity', 'logistic', 'tanh', 'relu'] [0,5,10]

For SVM models, the grid search only included the linear kernel as the RBF one does not retrieve feature importance.

For the models of SGD, SVM, RF and GB the feature importance was assessed and plotted using the function *feature_importances*. All the other models do not retrieve feature importance.

Additionally, validation, learning curves and ROC curves were plotted.

7.1.4 *Application to predict the location of peptide sequence*

In order to select the models, the scores of MCC, accuracy, AUC-ROC, sensibility and specificity were taken into special consideration. The best models were used to predict the location of a known fusion peptide from a Dengue virus viral fusion protein sequence.

To this task, the package function *predict_window* was used, as it allows to use a sliding window approach to scan all the possible subsequences from a full protein sequence using an input trained ML model. A window size of 20 and a gap space of 1 was used and all the subsequences and respective probability percentages of being or not a viral fusion peptide were recorded.

7.2 RESULTS AND DISCUSSION

The results obtained for the three major datasets, their analysis and discussion are presented in this section.

7.2.1 *Dataset 1*

The first dataset was used to evaluate the models' capability to distinguish fusion peptides from other random sequences belonging to the fusion protein. Overall, this dataset obtained models with very good performances. Models with only AAC, CTD or only PAAC features performed well, with MCC in the order of the 80%. The models that used a more developed set of features, however, achieved scores higher than 90%, specially models of SVM and ANN.

Feature selection demonstrated also to be a good asset for the development of such models. It decreased the number of features and increased the performance of the models. However, it was not easy to perceive which feature selection scheme was better across the majority of models. In the table below (Table 14), it is possible to observe a selection of the best models for each set of features.

The results were better than expected and probably are overfitting. Nonetheless, the models were analysed. Learning curves for all models were also plotted, the majority showed higher training and lower validation scores. As validation scores could be increased with more training samples, the model could benefit from more training samples.

Table 14: Table with the best performing models in dataset 1 for all sets of features

Feature Selection	Number Features	ML Model	Scores				
			MCC	Acc	AUC-ROC	Sens.	Specif.
all mutual_svc	76	SVM	0.97	0.99	0.99	0.97	1
all tree	569	SVM	0.97	0.99	0.99	0.97	1
all mutual_tree	278	SVM	0.94	0.97	0.97	0.94	1
all svc	78	SGD	0.97	0.99	0.99	0.97	1
all svc	78	ANN	0.97	0.99	0.99	0.97	1
all tree	569	RF	0.89	0.94	0.94	0.89	1
AAC	20	KNN	0.91	0.96	0.96	0.94	0.97
CTD	135	RF	0.83	0.91	0.91	0.86	0.97
PAAC	60	SGD	0.84	0.91	0.92	0.83	1

When fed with a viral fusion protein of Dengue virus, to predict the location of viral fusion peptide, the models predict the fusion peptide sequence but also several other zones. This indicates that the model obtained with this dataset has a good recall, but its precision needs to be improved. However, we note that in this case recall is more important than precision, since our aim is to make predictions that can be validated or discarded by in ***in vitro*** experiments. The model that got closer was the RF fed with a dataset containing all features selected by a tree. In Figure 6, it is possible to observe the regions predicted with a confidence ≥ 0.99 in yellow, regions with confidence of 0.95-0.99 in blue and the actual location of FPep in red.

MRCIGISNRDFVEGVSGGSWVDIVLEHGSCVTTMAKNKPTLDFELIETEAKQPATLRKYCIEAKLTNTTTDSRCPTQ
GEPSLNEEQDKRFVCKHSMVDRGWGNGCGLFGKGGIVTCAMFTCKKNMKGKVVQPENLEYTIVITPHSGEEHAV
GNDTGKHGKEIKITPQSSITEAELTGYGTVTMECSPRTGLDFNEMVLLQMENKAWLVHRQWFLDLPLPWLPGAD
TQGSNWIQKETLVTFKNPHAKKQDVVVLGSQEGAMHTALTGATEIQMSSGNLLFTGHLKCRLRMDKLQLKGMSY
SMCTGKFKVVKEIAETQHGTIVIRVQYEGDGSPCKIPFEIMDLEKRHVLGRLITVNPIVTEKDSPVNIEAEPPFGDSYII
GVEPGQLKLNWFKKGSSIGQMIETTMRGAKRMAILGDTAWDFGSLGGVFTSIGKALHQVFGAIYGAAFSGVSWI
MKILIGVIITWIGMNSRSTSLSVSLVLVGVVTLYLGVMVQA

Figure 6: Dengue's fusion protein. Classification using dataset1, RF model and all features selected with tree. Regions predicted as containing fusion peptides with confidence ≥ 0.99 highlighted in yellow, 0.95-0.99 in blue. Actual fusion peptide displayed in the red box.

It is possible to observe that although the model predicted the FPep, it also predicted some sequences immediately before and after the actual FPep. Besides this, it also predicts as FPep a portion of the end of the fusion protein that is the TMD of the sequence.

The results showed that this dataset, although makes acceptable previsions it is not able to distinguish fusion peptides from transmembrane domains. However, it is able to distinguish fusion peptides and TMDs from the other parts of sequence. If we look into the relevant

features used in the RF model (Figure 7), it is possible to observe that PAAC, CTD features (especially the ones regarding the composition), some autocorrelation factors and QSO are relevant. Besides that, the aminoacids *phenylalanine (phe)* (F) and *glycine (gly)* (G) also appear to be relevant.

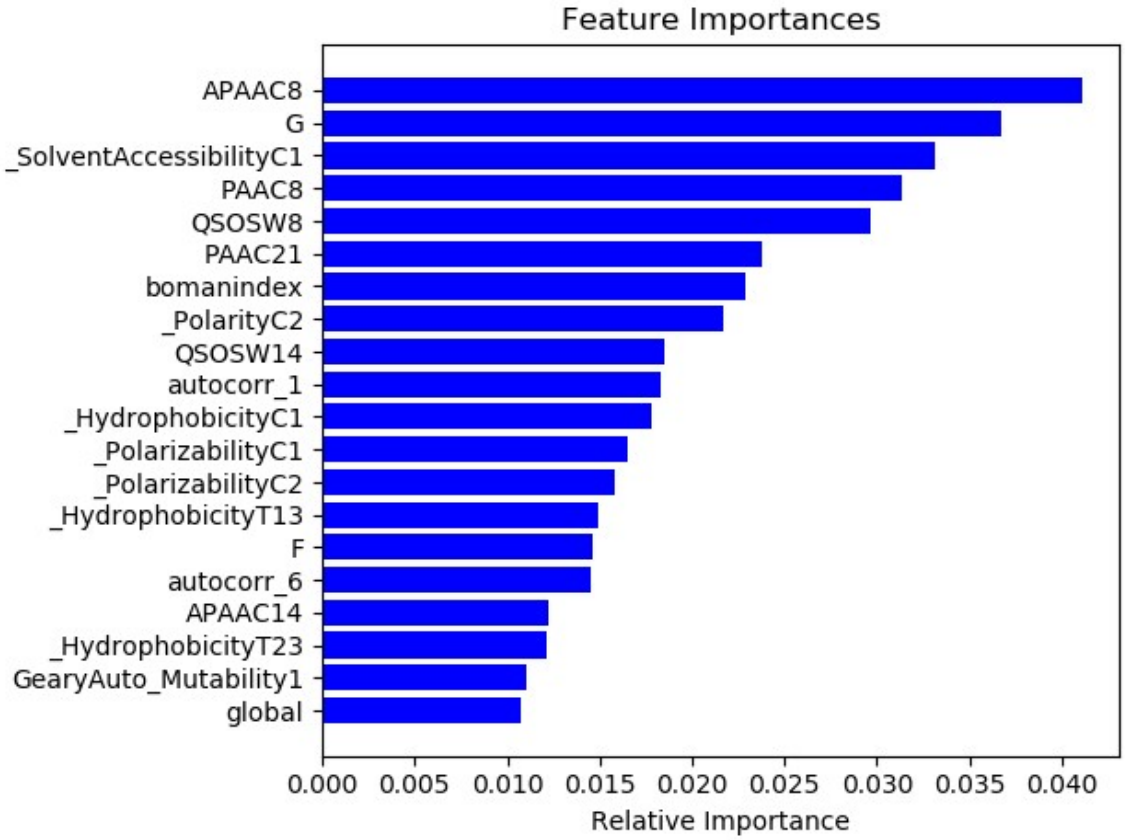


Figure 7: Feature importance of the RF model using the dataset with all features selected recurring to a tree model

PAAC are the initials for pseudo aminoacid composition, and APAAC the initials for amphiphilic pseudo aminoacid composition, the number after these initials represents the amino acid index. So, contributing to the model are PAAC and APAAC gly and APAAC phe, indicating the importance of Glycine and Phenylalanine. In the same way, Glycine and Phenylalanine single aminoacid frequencies appear to have a relevant role in the machine learning model to distinguish fusion peptides, being both hydrophobic residues and often buried in the protein core, phe being aromatic and gly aliphatic.

CTD features' aim is to capture the meaningful biological information from the protein calculating the composition, transition and distribution of aminoacid attributes, such as hydrophobicity, normalized van der Waals volume, polarity, polarizability, charge, secondary

structure and solvent accessibility of the protein sequences. From the CTD features, composition features, that indicates the fraction of aa with a particular property, appears as very relevant. Solvent accessibility C1 indicates the importance of the fraction of aa that are buried inside the core of protein (A,L,F,C,G,I,V,W), polarity C2 the fraction of aa with polarity between 8 and 9.2 (P,A,T,G,S), Hydrophobicity C1 indicates polar aa (R,K,E,D,Q,N), polarizability C1 (G,A,S,D,T) and C2 (C,P,N,V,E,Q,I,L) indicate the frequency of polarized aa between 0-1.08 and 0.128-0.186 respectively. Besides composition, a feature also important is the percentage transition of hydrophobicity between classes 1 and 3, and 2 and 3.

Quasi sequence order using the matrix Schneider-wrede (QSO) from glycine and phenylalanine and the Boman index also achieved high relative importances. Boman index indicates the potential protein interaction. The index, proposed by Boman, is equal to the sum of the solubility values for all residues in a sequence and might give an overall estimate of the potential of a peptide to bind to membranes or other proteins as receptors [77].

Autocorrelation 1 and 6 are descriptive of autocorrelation values taking into consideration the Einsenberg hydrophobicity consensus amino acid scale (Eisenberg hydrophobicity consensus amino acid scale [114]).

In this studied model is latent the weight that glycine and phenylalanine, both hydrophobic aa and their characteristics are having to distinguish viral fusion peptides. This makes sense as viral fusion peptides and TMDs are hydrophobic sequence, which explains the difficult of the model to distinguish them.

Taking a look into other models, that achieve better results, but did not distinguish so well the viral fusion peptides and TMDs, it is possible to observe that the features selected are different. In random forests, the features used are quite similar with the ones described above, with some models benefiting from more autocorrelation descriptors, like MoranAutoMutability 1, Geary auto residue ASA4, Moran auto polarizability4, moranautofreeenergy3, as well as distribution of secondary structures (D component of CTD), like the secondary structure D1025 that represents the distribution of helix in the first quarter of sequence and the secondary structure D3001 representing the distribution of coil aa (gly, *asparagine (asn)*,phe, *serine (ser)*, *aspartic acid (asp)*) on the first residue. SVM models look into other features with more components of CTD and some dipeptide and tripeptide composition being relevant.

Models predicting only with AAC give higher importance to phe and gly aa, which is concordance with the above results, with *alanine (ala)*, *lysine (lys)*, asn and *arginine (arg)* aa being also relevant (lys, asn, arg appear to have negative contributions in SVM predicting the negative class).

Models predicting with only CTD give higher importance to Solvent Accessibility and hydrophobicity C1, Polarity and polarizability C2, charge and secondary structure also in concordance with the results above.

Models predicting only with PAAC and APAAC give higher importances to gly and phe (PAAC and APAAC 8 and 14), and also to PAAC21 , 27 and PAAC and APAAC 6 and 3 *glutamine (gln)* and asn, reinforcing the importance of phe and gly.

7.2.2 *Dataset2*

The second dataset was used to evaluate the models' ability to distinguish fusion peptides from transmembrane domains. Models with this dataset achieved performances very close or even 1. Models with only AAC, CTD or PAAC features performed well with MCC in the order of the 90%. The models that used a more developed set of features, however, achieved scores higher than 90%. Feature selection was not a differentiating factor. In the table below (Table 15), it is possible to observe a selection of the best models for each set of features.

Table 15: Best performing models in dataset 2 for all set of features

Feature Selection	**Number Features**	**ML Model**	**Scores**				
			MCC	**Acc**	**AUC-ROC**	**Sens.**	**Specif.**
svc	67	SVM	1	1	1	1	1
svc	67	SGD	1	1	1	1	1
svc	67	RF	0.97	0.99	0.99	0.97	1
svc	67	ANN	1	1	1	1	1
AAC	20	RF	0.91	0.96	0.96	0.92	1
CTD	135	KNN	0.94	0.97	0.97	1	0.94
PAAC	60	SVM	0.91	0.96	0.95	0.97	0.94

When fed with a viral fusion protein of Dengue virus, in order to forecast the location of viral fusion peptide, the models did not obtained good results predicting several other zones not related either with fusion peptide nor TMD region. These models were not able to distinguish the fusion peptide from other zones of the sequence.

Although the models using this dataset did not identify the viral fusion peptide, the features used could provide clues to differentiate FPep from TMDs.

CTD features specially linked to hydrophobicity, polarity and charge, autocorrelation descriptors linked to flexibility, hydrophobicity and polarizability appear as relevant features. Secondary structure as helix, length, aliphatic index (relative volume occupied by aliphatic side chains as ala, *valine (val)*, *isoleucine (ile)* and *leucine (leu)*), PAAC and some aminoacids as leu and gly are relevant as well. This set of features is a very good indicator of the differences between this two zones of viral fusion proteins. They are in concordance with the literature that points out as differentiating characteristics the aa, hydrophobicity and structural properties [115, 116].

Both FPep and TMDs have similar properties such as high hydrophobicity, which implies high content of aa such as ala, ile, leu, *methionine (met)*, phe, val, *proline (pro)* and gly. However, the relative frequencies of each aa residues may vary between FPeps and TMDs. Fusion peptides are hydrophobic sequences composed of 20-30 aminoacids mostly apolar residues, enriched with alanines and glycines, whereas TMDs consist of around 20 aa hydrophobic residues with also a greater content of glycine [115].

Other important feature is the structural flexibility, where regarding the FPep is not a universal criteria differing between classes. The transmembrane domain, on contrary, adopt mainly a helix structure [116, 115].

Besides this, FPeps are usually located further upstream in the sequence than TMDs (located at the C terminus) [116]. FPeps are very conserved among viruses of the same family and, in pre-fusion state, are buried in the fusion protein, whereas TMDs are exposed (connect to the membrane). To refine the models ability to distinguish FPep and TMDs, aligments, structural (especially in pre-fusion state) and location information could be added.

The exploration of these models could also reveal different features not explored yet to distinguish TMDs from FPeps.

7.2.3 *Dataset3*

The third dataset was used to evaluate the models' capability to distinguish fusion peptides from a negative set, having both random sequences from the fusion protein and transmembrane domains.

As expected, the models obtained did not achieved MCC scores as higher as the previous datasets. Models with only AAC, CTD or only PAAC features performed poorly not achieving MCC scores higher than 60%. Models using more features had better results, specially SVM and SGD. The RF, KNN and GNB performed worst. With this dataset only the models based on feature selection with SVC got scores higher than 0.70, representing a significative difference to the models using none or with feature selection using tree models.

Furthermore, and as had succeed with above two models, using a combination set of features instead of individual sets of features gives better performances. This is also shown in several studies such as [84].

In the table below (Table 16) it is possible to observe the models with MCC scores ≥ 0.72.

SVM models have optimized parameters of C value of 0.01 and a linear kernel. SGD models were obtained with loss function of hinge and alpha of 1 to feature selection only with SVC and loss function log and alpha of 10 for both univariate and SVC feature selection. Both models used L2 regularization. ANN was achieved using a identity activation function with batch size of 5 and the GNB had a var smoothing parameter of 0.0001.

Table 16: Best performing models in dataset 3 for all set of features

Feature Selection	Number Features	ML Model	Scores				
			MCC	Accuracy	AUC-ROC	Sens.	Specif.
svc	122	SVM	0.8	0.9	0.9	0.83	0.97
mutual_svc	105	SVM	0.76	0.88	0.88	0.86	0.9
svc	122	SGD	0.8	0.9	0.9	0.83	0.97
mutual_svc	105	SGD	0.73	0.85	0.86	0.75	0.97
svc	122	ANN	0.72	0.85	0.86	0.78	0.94
svc	122	GNB	0.72	0.84	0.85	0.69	1

The learning curves plotted, showed high training scores and low validation scores suggesting that the model could benefit from more training samples.

When fed with viral fusion protein of Dengue virus, to predict the location of the viral fusion peptide, the models achieved good results predicting the fusion peptide location relatively accurately. Figures 8, 9 and 10 represent the regions predicted with a confidence of ≥ 0.99 in yellow, regions with confidence of $0.95 - 0.99$ in blue and the actual location of FPep in a red box for the three best performing models. Figure 8 is obtained with a SVM model and feature selection using SVC. Figure 9 is obtained with a SVM model using univariate followed by SVC feature selection. Figure 10 represents the prediction with a SGD model using univariate followed by SVC feature selection.

MRCIGISNRDFVEGVSGGSWVDIVLEHGSCVTTMAKNKPTLDFELIETEAKQPATLRKYCIEAKLTNTTTDSRCPTQ
GEPSLNEEQDKRFVCKHSMVDRGWGNGCGLFGKGGIVTCAMFTCKKNMKGKVVQPENLEYTIVITPHSGEEHAV
GNDTGKHGKEIKITPQSSITEAELTGYGTVTMECSPRTGLDFNEMVLLQMENKAWLVHRQWFLDLPLPWLPGAD
TQGSNWIQKETLVTFKNPHAKKQDVVVLGSQEGAMHTALTGATEIQMSSGNLLFTGHLKCRLRMDKLQLKGMSY
SMCTGKFKVVKEIAETQHGTIVIRVQYEGDGSPCKIPFEIMDLEKRHVLGRLITVNPIVTEKDSPVNIEAEPPFGDSYIII
GVEPGQLKLNWFKKGSSIGQMIETTMRGAKRMAILGDTAWDFGSLGGVFTSIGKALHQVFGAIYGAAFSGVSWI
MKILIGVIITWIGMNSRSTSLSVSLVLVGVVTLYLGVMVQA

Figure 8: Dengue's fusion protein. Regions predicted by SVM model (dataset 3 and feature selection SVC) as containing fusion peptides with confidence ≥ 0.99 highlighted in yellow, 0.95-0.99 in blue. Actual fusion peptide displayed in the red box

Figure 8, SVM model and feature selection using SVC, predicted with relative high accuracy the location of the FPep. The two first aa of the viral FPep only are revealed when taking in consideration probabilities between 0.95 and 0.99.

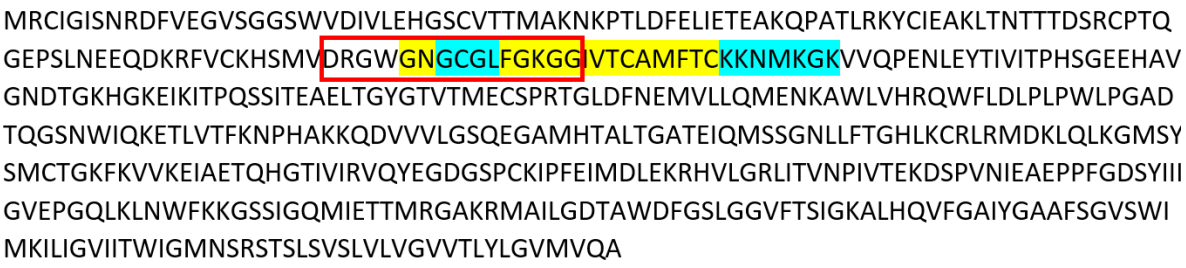


Figure 9: Dengue's fusion protein. Regions predicted by SVM model (dataset 3 and feature selection univariate and SVC) as containing fusion peptides with confidence $\geq$ 0.99 highlighted in yellow, 0.95-0.99 in blue. Actual fusion peptide displayed in the red box.

The SVM model using univariate followed by SVC feature selection, Figure 9, also predicted with relative high accuracy the location of the FPep. Although it only locates the FPep 4 aa downstream of the real one. Considering the prevision higher than 0.99 it predicts the location in a more constrained way than the other models.

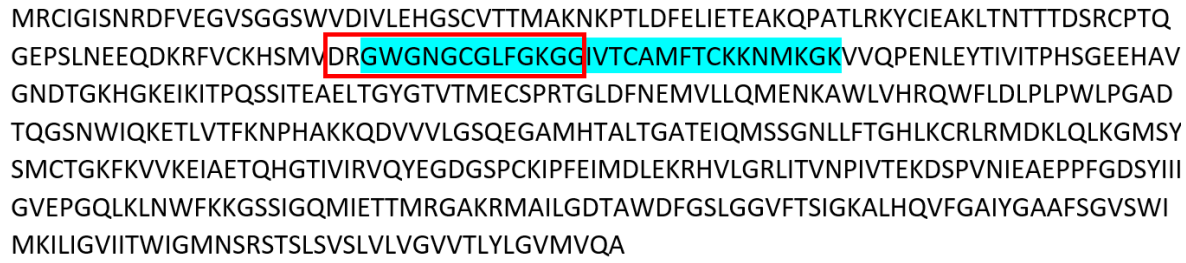


Figure 10: Dengue's fusion protein. Regions predicted by SGD model (dataset 3 and feature selection univariate and SVC) as containing fusion peptides with confidence $\geq$ 0.99 highlighted in yellow, 0.95-0.99 in blue. Actual fusion peptide displayed in the red box.

Figure 10 represents the prediction with a SGD model using univariate followed by SVC feature selection. This model did not predict any subsequence with a confidence higher than 0.99. With probabilities between 0.95-0.99, it locates the FPep relatively close to the real one, starting 2 aa downstream and ending with 16 aa downstream of the real one.

All three models performed well, predicting the location of the FPep of Dengue's virus fusion protein without predicting the TMD sequence. However, any of them recognize with confidence the first 2 aa of FPep and all predicted several aa right after the real FPep as being part of the FPep. This is not surprising, given that the actual limits of viral fusion peptides are somewhat arbitrary and not clearly defined. Besides this, another possible correlated reason may be the fact that the algorithm search for subsequences with 20 aa. Nonetheless, it is necessary to experiment the models in more known sequences to validate them.

Table 17 displays the highest score features (positive and negative) with their *features importances (FI)* in all 4 models tested. The features common in all 4 models are highlighted

(the positive ones in red and the negatives in blue). Tripeptide and dipeptide composition, secondary structure, autocorrelation and CTD descriptors appear as relevant features.

TPC and DPC appeared as some of the relevant features, specially involving the phe and gly aa. It is worth noting that while phe appears on the positive hyperplane space, the gly is associated with the negative space. Tripeptides with ala in composition are also important.

In the first residue (on positive side of hyperplane) aa related to helix secondary structure (*glutamic acid (glu)*, ala, leu, met, gln, lys, arg, *histidine (his)*) are expected. When considering the first 75% of residues of the sequence (as features on the negative side of the hyperplane) are expected aa linked to random coil secondary structure (gly, asn, pro, ser, asp).

Autocorrelation Mutability calculated by Geary, that describes the probability of each aminoacid change in a given small evolutionary interval [117], is relevant in the positive space of the SVM hyperplane.

Boman index, that indicates the potential to protein interaction [77] with membranes or other receptors is also common in all 4 models in the negative space. This result is highly expected in viral fusion peptides.

CTD features involving solvent accessibility, polarity and hydrophobicity as autocorrelation descriptors, PAAC and APAAC are important as well. As said before, although there is no precise definition of the fusion peptide, it generally is a hydrophobic or amphipathic segment rich in gly and ala residues, located at the N terminus of viral protein sequence and with reduced to moderate polarity [118]. Another criterion is that mutation in the fusion peptide segment of the fusion protein often leads to loss of activity because of the essential nature of the fusion peptide segment for membrane fusion [2]. These characteristics are very patent in the models with the importance of aminoacids gly, ala and phe.

Another structural characteristic of viral fusion peptides is their secondary structure. They often are helices or partially a-helical although some families have other structure [2]. The appearance of relevant features linked to distribution of secondary structure is therefore expectable. Besides this, the features related with Boman index, solvent accessibility, polarity and hydrophobicity, autocorrelation and PAAC also agree with the FPep properties described in literature, make sense in the problem context and could provide clues to a clearer insight of viral fusion peptides.

Despite the scarcity of data and the lack of case studies, the models provided good results and the features extracted make biological sense, suggesting that the machine learning approach followed, although it could be improved, is correct. With no doubt, the best dataset to work to predict a fusion peptide is the dataset 3, although it yields lower MCC scores, it produce models that clearly identify the FPep not predicting TMDs or other parts of fusion protein. Taking this dataset into account, feature selection revealed to be very important with only feature selection based on SVC models producing good results. In the same line,

SGD, SVM and ANN models yielded satisfactory results whereas tree based models as RF demonstrate not to be as good for this kind of problem.

One should take into consideration that the dataset is composed primarily of sequences belonging to class I fusion proteins, which may lead to a high degree of similarity between sequences in the dataset. Nonetheless, it showed to perform well when fed with the Dengue fusion protein, a class II protein.

Adding more known case studies and extending with confidence the database of fusion peptides could lead to a significant improvement and provide important information on viral fusion peptides. Refining the models could also lead to improve the location prediction of viral fusion peptides.

Besides this, the models could also benefit from features, such as alignments, motif searches or location of the viral fusion peptide in sequence since these are important features in real situations but not described computationally in this study. Structural information of fusion proteins in pre-fusion state could also be a good feature to add, for the model be able to distinguish between FPep and TMD as one is buried in the fusion protein and the other is exposed to connect to the membrane.

All the discussed results give insights on the possible location of the FPep on the tested fusion protein sequences, however they lack experimental evidence. Hence, the next steps of this work should also include the experimentally validation of the results obtained in this dissertation.

This study and the features described by the models could provide unknown biological insight on sequence and function of viral fusion peptides. Here, it is demonstrated that secondary structure, hydrophobicity, polarity, flexibility and charge, aa composition (single, di and tripeptide composition) involving gly and phe, Boman index, autocorrelation descriptors, CTD and PAAC are very good features to distinguish fusion peptides. To test if these models are truly accurate more secure case studies and wet lab experiments would be needed.

Table 17: Table discriminating the highest FI from the 4 models divided by the hyperplane space they occupy. Features that are common in all four models are highlighted, in red the positive ones and in blue the negatives ones.

SVM (svc sel.)	FI	SGD (svc sel.)	FI	SVM (mutual_svc sel.)	FI	SGD (mutual_svc sel.)	FI
KGG	0.155	PolarityC1	0.114	HydrophobicityC3	0.138	HydrophobicityC3	0.297
PolarityC1	0.122	FCS	0.100	SecondaryStr D1001	0.133	FCS	0.279
RSA	0.110	KGG	0.100	global	0.133	global	0.279
FIG	0.110	Solvent AccessibilityC1	0.099	FCS	0.130	FIG	0.274
Solvent AccessibilityC1	0.110	MoreauBrotoAuto ResidueVol13	0.098	FIG	0.129	IAG	0.265
VIA	0.093	AFC	0.094	IAG	0.108	SecondaryStr D1001	0.238
FCS	0.092	SecondaryStr D1001	0.088	PAAC28	0.108	PAAC28	0.227
MoreauBrotoAuto ResidueVol13	0.091	autocorr_3	0.078	PAAC22	0.107	GearyAuto Mutability5	0.218
autocorr_3	0.090	GearyAuto Mutability5	0.078	MoranAuto AvFlexibility19	0.099	MoranAuto AvFlexibility19	0.217
AFC	0.089	LTS	0.076	GearyAuto Mutability5	0.096	NormalizedVDWV D3075	0.207
SecondaryStr D1001	0.089	VIA	0.074	PAAC13	0.085	PAAC22	0.201
GearyAuto Mutability5	0.077	PAAC27	0.072	YST	0.084	F	0.192
F	0.076	RSA	0.070	F	0.082	QC	0.192
PAAC27	0.074	FIG	0.070	SLS	0.080	PAAC30	0.189
AAT	0.069	MF	0.068	NormalizedVDWV D3075	0.080	LI	0.186
LTS	0.066	M	0.068	SAA	0.078	PAAC13	0.185
GS	-0.069	GIG	-0.068	GearyAuto Steric22	-0.077	Solvent AccessibilityT23	-0.164
GIG	-0.070	GST	-0.068	APAAC16	-0.079	taugrant3	-0.165
GST	-0.070	MoreauBrotoAuto AvFlexibility24	-0.068	NR	-0.079	Solvent AccessibilityD3050	-0.170
AVT	-0.071	GS	-0.071	IAL	-0.079	HydrophobicityT13	-0.171
AVP	-0.072	VSV	-0.075	MoranAuto ResidueASA5	-0.080	MoranAuto ResidueASA5	-0.179
GearyAuto Mutability20	-0.083	AVT	-0.075	APAAC17	-0.083	GWT	-0.193
VSV	-0.087	GWT	-0.075	APAAC20	-0.089	GIG	-0.196
SecondaryStr D3100	-0.089	MoranAuto AvFlexibility20	-0.078	GIG	-0.090	APAAC20	-0.199
MoranAuto AvFlexibility20	-0.100	SecondaryStrD3100	-0.079	ME	-0.090	GS	-0.208
bomanindex	-0.103	LVD	-0.091	Solvent AccessibilityD3050	-0.114	IAL	-0.212
SecondaryStr D3075	-0.106	GearyAuto Mutability20	-0.093	GWT	-0.115	NR	-0.224
GearyAuto Hydrophobicity20	-0.114	GearyAuto Hydrophobicity20	-0.102	bomanindex	-0.117	APAAC17	-0.235
LVD	-0.121	bomanindex	-0.104	GS	-0.142	bomanindex	-0.285
GWT	-0.121	SecondaryStrD3075	-0.107	SecondaryStr D3075	-0.172	SecondaryStr D3075	-0.414

8

CONCLUSION AND PROSPECTS FOR FUTURE WORK

The first aim of this project was to build a generic automated platform for the classification of peptides/proteins based on their physicochemical properties and making use of different machine learning models. The package developed facilitates the major tasks of machine learning and it includes modules to read and alter sequences, to calculate protein features, do dataset preprocessing, do feature reduction and selection, perform clustering and to build machine learning models and make predictions. As it is built in a modular way, the user retains the power to manipulate and use others functions outside of the package, having control over the different steps and adapting/extending the code to fit their specific needs. The package is directed to handle protein related problems, but its modular construction allows users to use it in other problems.

This package was validated using two membrane-interacting peptides studies involving antimicrobial and anticancer peptides. The comparative analysis made evident the performance and validated the package here described.

Although it meets all the tasks it was built for, the package could be improved. More features could be included (structural and evolutionary), the code could be improved to enhance efficiency, and other traits could be added such a deep learning pipeline, more ML functions and availability as a web server.

During the development of this thesis, several web servers and stand alone packages were published and released. Nonetheless, the developed package maintains its usability and interest. It is advantageous because:

- It offers the option to change the sequences and obtain subsequences (such as N and C terminals or scanning windows)
- It can extract/calculate a high number and type of descriptors for protein sequences;
- It is designed to conduct all main steps to construct a predictor, facilitating the machine learning process in all of its stages (which is not common in the available packages);
- It is, compared to others, more user friendly with more variety of plots and schemes to help cluster, features and machine learning analyses;

- It is built in modular ways, which is also a major advantage, as the user can, easily integrate other features or use other methods of different tools to complete their work

The second major goal of this work was, using the tool developed, explore the viral fusion peptides, one of the most relevant players in viral fusion processes and therefore, very promising drug targets. The use of ML to identify and further understand viral fusion peptides is a promising and unexplored approach, and, as far as we know, there are no ML studies focused on them.

Here, a ML pipeline was applied to understand the major differences between TMDs and FPeps and from FPeps from the rest of the viral fusion protein sequence. The models developed accurately predict the location of the FPep inside the fusion protein and could be used not only, to predict the location, but to understand and unravel the distinctive characteristics of these peptides.

A complex set of physicochemical features was compared to the use of individual sets of features. Different feature techniques and ML algorithms were tested. The models using a combination set of features instead of individual sets achieved better performances. Feature selection revealed to be an important asset, with the use of a SVC L1 penalized approach performing better than approaches involving trees. ML models SVM and SGD also outperformed other models.

The feature analysis revealed key differentiating characteristics of viral fusion peptides. Secondary structure, hydrophobicity, polarity, flexibility and charge, aa composition (single, di and tripeptidecomposition) involving gly and phe, Boman index, autocorrelation descriptors, CTD and PAAC revealed to be important features to distinguish the fusion peptides. These characteristics were in concordance with bibliography and make biological sense. Furthermore, this approach could be used to deeply understand and find characteristics not yet described.

To further improve this study, future steps would be the extension of the fusion peptide database. Besides this, the models could also benefit from features such as alignments, motif searches or location in sequence, as they are important features in real situations but not described computationally in this study. Structural information of fusion proteins in pre-fusion state could also be a good feature for the model to be able to distinguish between FPep and TMDs as one is buried in the fusion protein and the other is exposed to connect to the membrane. To test if these models are truly accurate more secure case studies and wet lab experiments would be needed.

Overall, the models built provide good results and the features extracted make biological sense, suggesting that the machine learning approach followed, although not perfect, is correct. These models could be used to understand and find precious data to understand the viral fusion process.

BIBLIOGRAPHY

[1] D. Gaspar, A. Salomé Veiga, and M. A. R. B. Castanho, "From antimicrobial to anticancer peptides. A review," *Frontiers in Microbiology*, vol. 4, no. OCT, pp. 1–16, 2013.

[2] R. M. Epand, "Fusion peptides and the mechanism of viral fusion," 2003.

[3] D. S. Cao, Q. S. Xu, and Y. Z. Liang, "Propy: A tool to generate various modes of Chou's PseAAC," *Bioinformatics*, vol. 29, no. 7, pp. 960–962, 2013.

[4] Z. He, J. Zhang, X. H. Shi, L. L. Hu, X. Kong, Y. D. Cai, and K. C. Chou, "Predicting drug-target interaction networks based on functional groups and biological features," *PLoS ONE*, vol. 5, no. 3, pp. 1–8, 2010.

[5] A. Sharma, P. Gupta, R. Kumar, and A. Bhardwaj, "DPABBs: A Novel in silico Approach for Predicting and Designing Anti-biofilm Peptides," *Scientific Reports*, vol. 6, no. July 2015, pp. 1–13, 2016.

[6] E. Y. Lee, B. M. Fulan, G. C. L. Wong, and A. L. Ferguson, *Mapping membrane activity in undiscovered peptide sequence space using machine learning*, vol. 113. 2016.

[7] L. Xu, G. Liang, L. Wang, and C. Liao, "A novel hybrid sequence-based model for identifying anticancer peptides," *Genes*, vol. 9, no. 3, 2018.

[8] S. Akbar, M. Hayat, M. Iqbal, and M. A. Jan, "iACP-GAEnsC: Evolutionary genetic algorithm based ensemble classification of anticancer peptides by utilizing hybrid feature space," *Artificial Intelligence in Medicine*, vol. 79, pp. 62–70, 2017.

[9] W. Chen, H. Ding, P. Feng, H. Lin, and K.-c. Chou, "iACP: a sequence-based tool for identifying anticancer peptides," *Oncotarget*, vol. 7, no. 13, pp. 16895–16909, 2016.

[10] S. Kumar and H. Li, "In Silico Design of Anticancer Peptides," vol. 1647, 2017.

[11] Z. Hajisharifi, M. Piryaiee, M. Mohammad Beigi, M. Behbahani, and H. Mohabatkar, "Predicting anticancer peptides with Chou's pseudo amino acid composition and investigating their mutagenicity via Ames test," *Journal of Theoretical Biology*, vol. 341, pp. 34–40, 2014.

[12] B. Manavalan, S. Basith, T. Hwan Shin, S. Choi, M. Ok Kim, and G. Lee, "MLACP: machine-learning-based prediction of anticancer peptides," *Oncotarget*, vol. 8, no. 44, pp. 77121–77136, 2017.

[13] F. Grisoni, C. S. Neuhaus, G. Gabernet, A. T. Müller, J. A. Hiss, and G. Schneider, "Designing Anticancer Peptides by Constructive Machine Learning," *ChemMedChem*, vol. 13, no. 13, pp. 1300–1302, 2018.

[14] B. Liu, "BioSeq-Analysis: a platform for DNA, RNA and protein sequence analysis based on machine learning approaches," *Briefings in Bioinformatics*, no. January, pp. 1–15, 2017.

[15] F. G. Avci, B. S. Akbulut, and E. Ozkirimli, "Membrane active peptides and their biophysical characterization," *Biomolecules*, vol. 8, no. 3, pp. 1–43, 2018.

[16] N. M. O'Brien-Simpson, R. Hoffmann, C. S. B. Chia, and J. D. Wade, "Editorial: Antimicrobial and Anticancer Peptides," *Frontiers in Chemistry*, vol. 6, no. February, pp. 1–2, 2018.

[17] World Health Organization (WHO), "The top ten leading causes of death by broad income group," *Fact sheet*, 2011.

[18] M. Mahlapuu, J. Håkansson, L. Ringstad, and C. Björn, "Antimicrobial Peptides: An Emerging Category of Therapeutic Agents," *Frontiers in Cellular and Infection Microbiology*, vol. 6, no. December, pp. 1–12, 2016.

[19] D. W. Hoskin and A. Ramamoorthy, "Studies on Anticancer Activities of Antimicrobial Peptides," vol. 64, no. 12, pp. 2391–2404, 2008.

[20] G. Wang, X. Li, and Z. Wang, "APD3: The antimicrobial peptide database as a tool for research and education," *Nucleic Acids Research*, vol. 44, no. D1, pp. D1087–D1093, 2016.

[21] W. Shoombuatong, N. Schaduangrat, and C. Nantasenamat, "Unraveling the bioactivity of anticancer peptides as deduced from machine learning," *EXCLI Journal*, vol. 17, no. Thundimadathil 2012, pp. 734–752, 2018.

[22] M. Malmsten, "Interactions of Antimicrobial Peptides with Bacterial Membranes and Membrane Components," *Current Topics in Medicinal Chemistry*, vol. 16, no. 1, pp. 16–24, 2015.

[23] B. Bechinger and S. U. Gorr, "Antimicrobial Peptides: Mechanisms of Action and Resistance," *Journal of Dental Research*, vol. 96, no. 3, pp. 254–260, 2017.

[24] P. Bhadra, J. Yan, J. Li, S. Fong, and S. W. Siu, "AmPEP: Sequence-based prediction of antimicrobial peptides using distribution patterns of amino acid properties and random forest," *Scientific Reports*, vol. 8, no. 1, pp. 1–10, 2018.

[25] A. Tyagi, P. Kapoor, R. Kumar, K. Chaudhary, A. Gautam, and G. P. Raghava, "In silico models for designing and discovering novel anticancer peptides," *Scientific Reports*, vol. 3, pp. 1–8, 2013.

[26] B. Apellániz, N. Huarte, E. Largo, and J. L. Nieva, "The three lives of viral fusion peptides," *Chemistry and Physics of Lipids*, vol. 181, pp. 40–55, 2014.

[27] G. P. Pattnaik, G. Meher, and H. Chakraborty, "Exploring the Mechanism of Viral Peptide-Induced Membrane Fusion," pp. 69–78, 2018.

[28] B. Podbilewicz, "Virus and Cell Fusion Mechanisms," *Annual Review of Cell and Developmental Biology*, vol. 30, no. 1, pp. 111–139, 2014.

[29] S. C. Harrison, "Viral membrane fusionHarrison, S. C. (2009). Viral membrane fusion, 15(7), 690–698.," *National Structure Molecular Biology*, vol. 15, no. 7, pp. 690–698, 2009.

[30] L. K. Tamm and X. Han, "Viral fusion peptides: A tool set to disrupt and connect biological membranes," *Bioscience Reports*, vol. 20, no. 6, pp. 501–518, 2000.

[31] B. L. Victor, D. Lousa, J. M. Antunes, and C. M. Soares, "Self-assembly molecular dynamics simulations shed light into the interaction of the influenza fusion peptide with a membrane bilayer," *Journal of Chemical Information and Modeling*, vol. 55, no. 4, pp. 795–805, 2015.

[32] D. Lousa, A. R. Pinto, B. L. Victor, A. Laio, A. S. Veiga, M. A. Castanho, and C. M. Soares, "Fusing simulation and experiment: The effect of mutations on the structure and activity of the influenza fusion peptide," *Scientific Reports*, vol. 6, no. May, pp. 1–14, 2016.

[33] M. Awad and R. Khanna, *Efficient Learning Machines*. Apress Media, 2015.

[34] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 1, pp. 1–17, 2017.

[35] P. Baldi and S. Brunak, *Bioinformatics: The machine learning approach*. second ed., 2001.

[36] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, 2000.

[37] A. Smola and S. Vishwanathan, *Introduction to machine learning*. Cambridge University Press 2008, 2008.

[38] C. S. Greene, J. Tan, M. Ung, J. H. Moore, and C. Cheng, "Big data bioinformatics," *Journal of Cellular Physiology*, vol. 229, no. 12, pp. 1896–1900, 2014.

[39] A. C. Muller and S. Guido, *Introduction to Machine Learning with Python: A guide for data scientists*. O'Reilly Media, 2017.

[40] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[41] G. Schneider and U. Fechner, "Advances in the prediction of protein targeting signals," *Proteomics*, vol. 4, no. 6, pp. 1571–1580, 2004.

[42] E. Frank, L. Trigg, G. Holmes, and I. H. Witten, "Technical note: Naive Bayes for regression," *Machine Learning*, vol. 41, no. 1, pp. 5–25, 2000.

[43] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature Biotechnology*, 2008.

[44] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057. 2013.

[45] E. Veronese, U. Castellani, D. Peruzzo, M. Bellani, and P. Brambilla, "Machine learning approaches: From theory to application in schizophrenia," *Computational and Mathematical Methods in Medicine*, vol. 2013, 2013.

[46] Chih-Wei Hsu, Chih-Chung Chang, C.-J. Lin, Chih-Wei Hsu, Chih-Chung Chang, C.-J. Lin, Chih-Wei Hsu, Chih-Chung Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification," *BJU international*, 2008.

[47] B. Dasgupta, D. Liu, and H. T. Siegelmann, "Neural networks," in *Handbook of Approximation Algorithms and Metaheuristics*, 2007.

[48] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," 2015.

[49] L. Zhang, J. Tan, D. Han, and H. Zhu, "From machine learning to deep learning: progress in machine intelligence for rational drug discovery," *Drug Discovery Today*, vol. 22, no. 11, pp. 1680–1685, 2017.

[50] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, 2016.

[51] M. Bowles, *Machine Learning in python: essential techniques for predictive analysis*. 2015.

[52] T. Klikauer, "Scikit-learn: Machine Learning in Python," *TripleC*, 2016.

[53] J. Dean, *Big data, data mining, and machine learning: Value creation for business leaders and practitioners*. Wiley, 2014.

[54] Z.-H. Zhou, *Ensemble methods: Foundations and Algorithms*. 2012.

[55] I. Guyon and A. Elisseef, "An Introduction to Feature Extraction," pp. 1–24, 2006.

[56] J. Li, S. Fong, S. Mohammed, and J. Fiaidhi, "Improving the classification performance of biological imbalanced datasets by swarm optimization algorithms," *Journal of Supercomputing*, vol. 72, no. 10, pp. 3708–3728, 2016.

[57] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[58] S. Raschka, *Python Machine learning*. Packt Publishing, 2015.

[59] S. Piramuthu, "Evaluating Feature Selection Methods for Learning in Data Mining Applications," vol. 00, no. c, 1998.

[60] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[61] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in Bioinformatics*, vol. 2015, no. 1, 2015.

[62] L. Wang, Y. Wang, and Q. Chang, "Feature selection methods for big data bioinformatics: A survey from the search perspective," *Methods*, vol. 111, pp. 21–31, 2016.

[63] R. Y. Zheng, "Peptide Bioinformatics," *Artificial Neural Network: Methods and Applications2*, pp. 155–179, 2008.

[64] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," pp. 233–240, 2016.

[65] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric," *PLoS ONE*, vol. 12, no. 6, pp. 1–17, 2017.

[66] T. M. Mitchell, *Machine learning*. McGraw Hill Higher Education, 1997.

[67] I. Idris, *NumPy Beginner's Guide*. 2013.

[68] W. McKinney, "pandas: a Foundational Python Library for Data Analysis and Statistics," in *PyHPC*, 2011.

[69] S. Tosi, *Matplotlib for Python Developers*. 2009.

[70] J. Eric, O. Travis, P. Pearu, and et Al., "SciPy : Open source scientific tools for Python," *Computing in Science and Engineering*, 2001.

[71] F. Chollet and E. all., "Keras," 2015.

[72] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, Y. J. Michael Isard, Rafal Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, J. Chris Olah, O. Shlens, B. Steiner, I. Sutskever, P. T. Kunal Talwar, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[73] S. Vinga, "Biological sequence analysis by vector-valued functions: revisiting alignment-free methodologies for DNA and protein classification," pp. 1–36, 2006.

[74] K.-C. Chou, "Pseudo Amino Acid Composition and its Applications in Bioinformatics, Proteomics and System Biology," *Current Proteomics*, vol. 6, no. 4, pp. 262–274, 2009.

[75] F. M. Li and X. Q. Wang, "Identifying anticancer peptides by using improved hybrid compositions," *Scientific Reports*, vol. 6, pp. 1–6, 2016.

[76] Z. R. Li, H. H. Lin, L. Y. Han, L. Jiang, X. Chen, and Y. Z. Chen, "PROFEAT: A web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence," *Nucleic Acids Research*, vol. 34, no. WEB. SERV. ISS., pp. 32–37, 2006.

[77] H. G. Boman, "Antibacterial peptides: Basic facts and emerging concepts," *Journal of Internal Medicine*, vol. 254, no. 3, pp. 197–215, 2003.

[78] E. Y. Lee, B. M. Fulan, G. C. L. Wong, and A. L. Ferguson, *Mapping membrane activity in undiscovered peptide sequence space using machine learning*, vol. 113. 2016.

[79] S. Roy, D. Martinez, H. Platero, T. Lane, and M. Werner-Washburne, "Exploiting amino acid composition for predicting protein-protein interactions," *PLoS ONE*, vol. 4, no. 11, 2009.

[80] A. Basit, "Identification of Anticancer Peptides Using Optimal Feature Space of Chou ' s Split Amino Acid Composition and Support Vector Machine," 2017.

[81] D. S. Cao, Y. Z. Liang, J. Yan, G. S. Tan, Q. S. Xu, and S. Liu, "PyDPI: Freely available python package for chemoinformatics, bioinformatics, and chemogenomics studies," *Journal of Chemical Information and Modeling*, 2013.

[82] H. Ding, P. M. Feng, W. Chen, and H. Lin, "Identification of bacteriophage virion proteins by the ANOVA feature selection and analysis," *Molecular BioSystems*, vol. 10, no. 8, pp. 2229–2235, 2014.

[83] J. Dong, Z. J. Yao, L. Zhang, F. Luo, Q. Lin, A. P. Lu, A. F. Chen, and D. S. Cao, "PyBioMed: a python library for various molecular representations of chemicals, proteins and DNAs and their interactions," *Journal of Cheminformatics*, 2018.

[84] S. A. Ong, H. H. Lin, Y. Z. Chen, Z. R. Li, and Z. Cao, "Efficacy of different protein descriptors in predicting protein functional families," *BMC Bioinformatics*, vol. 8, pp. 1–14, 2007.

[85] D. S. Horne, "Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities," *Biopolymers*, vol. 27, no. 3, pp. 451–477, 1988.

[86] Y. Liang, "Prediction of Protein Structural Class Based on Different Autocorrelation Descriptors of Position – Specific Scoring Matrix," vol. 73, pp. 765–784, 2015.

[87] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang, "Predicting protein-protein interactions based only on sequences information," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 11, pp. 4337–4341, 2007.

[88] Y.-C. Wang, X.-B. Wang, Z.-X. Yang, and N.-Y. Deng, "Prediction of Enzyme Subfamily Class via Pseudo Amino Acid Composition by Incorporating the Conjoint Triad Feature," *Protein & Peptide Letters*, vol. 17, no. 11, pp. 1441–1449, 2012.

[89] Y. C. Wang, Y. Wang, Z. X. Yang, and N. Y. Deng, "Support vector machine prediction of enzyme function with conjoint triad feature and hierarchical context," *BMC Systems Biology*, vol. 5, no. SUPPL. 1, p. S6, 2011.

[90] K. C. Chou, "Prediction of protein subcellular locations by incorporating quasi-sequence-order effect," *Biochemical and Biophysical Research Communications*, vol. 278, no. 2, pp. 477–483, 2000.

[91] W. A. Abbasi, F. U. Hassan, A. Yaseen, and F. U. A. A. Minhas, "ISLAND: In-Silico Prediction of Proteins Binding Affinity Using Sequence Descriptors," pp. 1–14, 2017.

[92] K. C. Chou, "Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes," *Bioinformatics*, vol. 21, no. 1, pp. 10–19, 2005.

[93] A. T. Müller, G. Gabernet, J. A. Hiss, and G. Schneider, "modlAMP: Python for antimicrobial peptides," *Bioinformatics (Oxford, England)*, vol. 33, no. 17, pp. 2753–2755, 2017.

[94] A. Pande, S. Patiyal, A. Lathwal, C. Arora, D. Kaur, A. Dhall, G. Mishra, H. Kaur, N. Sharma, S. Jain, S. S. Usmani, P. Agrawal, R. Kumar, V. Kumar, and G. P. Raghava, "Computing wide range of protein/peptide features from their sequence and structure," *bioRxiv*, p. 599126, 2019.

[95] B. Panwar, S. Gupta, and G. P. Raghava, "Prediction of vitamin interacting residues in a vitamin binding protein using evolutionary information," *BMC Bioinformatics*, vol. 14, no. 1, 2013.

[96] P. Agrawal, S. Bhalla, K. Chaudhary, R. Kumar, M. Sharma, and G. P. Raghava, "In silico approach for prediction of antifungal peptides," *Frontiers in Microbiology*, vol. 9, no. FEB, pp. 1–13, 2018.

[97] H. B. Rao, F. Zhu, G. B. Yang, Z. R. Li, and Y. Z. Chen, "Update of PROFEAT: A web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence," *Nucleic Acids Research*, vol. 39, no. SUPPL. 2, pp. 385–390, 2011.

[98] D. Ofer and M. Linial, "ProFET: Feature engineering captures high-level protein functions," *Bioinformatics*, vol. 31, no. 21, pp. 3429–3436, 2015.

[99] R. Muhammod, S. Ahmed, D. Md Farid, S. Shatabda, A. Sharma, and A. Dehzangi, "PyFeat: a Python-based effective feature generation tool for DNA, RNA and protein sequences," *Bioinformatics*, pp. 2–3, 2019.

[100] Z. Chen, P. Zhao, F. Li, A. Leier, T. T. Marquez-Lago, Y. Wang, G. I. Webb, A. I. Smith, R. J. Daly, K. C. Chou, and J. Song, "IFeature: A Python package and web server for features extraction and selection from protein and peptide sequences," *Bioinformatics*, 2018.

[101] Z. Chen, P. Zhao, F. Li, T. T. Marquez-Lago, A. Leier, J. Revote, Y. Zhu, D. R. Powell, T. Akutsu, G. I. Webb, K.-C. Chou, A. I. Smith, R. J. Daly, J. Li, and J. Song, "iLearn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data," *Briefings in Bioinformatics*, vol. 00, no. January, pp. 1–11, 2019.

[102] H. B. Rao, F. Zhu, G. B. Yang, Z. R. Li, and Y. Z. Chen, "Update of PROFEAT: A web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence," *Nucleic Acids Research*, vol. 39, no. SUPPL. 2, pp. 385–390, 2011.

[103] S. Liu, L. Fan, J. Sun, X. Lao, and H. Zheng, "Computational resources and tools for antimicrobial peptides," *Journal of Peptide Science*, vol. 23, no. 1, pp. 4–12, 2017.

[104] S. Lata, N. K. Mishra, and G. P. S. Raghava, "AntiBP2: Improved version of antibacterial peptide prediction," *BMC Bioinformatics*, vol. 11, no. SUPPLL.1, pp. 1–7, 2010.

[105] X. Xiao, P. Wang, W. Z. Lin, J. H. Jia, and K. C. Chou, "IAMP-2L: A two-level multi-label classifier for identifying antimicrobial peptides and their functional types," *Analytical Biochemistry*, vol. 436, no. 2, pp. 168–177, 2013.

[106] F. H. Waghu, R. S. Barai, and S. Idicula-Thomas, "Leveraging family-specific signatures for AMP discovery and high-throughput annotation," *Scientific Reports*, vol. 6, pp. 1–7, 2016.

[107] P. K. Meher, T. K. Sahu, V. Saini, and A. R. Rao, "Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC," *Scientific Reports*, vol. 7, no. January, pp. 1–12, 2017.

[108] D. Veltri, U. Kamath, and A. Shehu, "Deep learning improves antimicrobial peptide recognition," *Bioinformatics*, vol. 34, no. 16, pp. 2740–2747, 2018.

[109] A. C. Kaushik and D.-Q. Wei, "An Accurate Bioinformatics Tool For Anti-Cancer Peptide Generation Through Deep Learning Omics," *bioRxiv*, p. 654277, 2019.

[110] Y. Xu, S. Yu, J. W. Zou, G. Hu, N. A. Rahman, R. B. Othman, X. Tao, and M. Huang, "Identification of peptide inhibitors of enveloped viruses using support vector machine," *PLoS ONE*, vol. 10, no. 12, pp. 1–15, 2015.

[111] A. Pande, S. Patiyal, A. Lathwal, C. Arora, D. Kaur, A. Dhall, G. Mishra, H. Kaur, N. Sharma, S. Jain, S. S. Usmani, P. Agrawal, R. Kumar, V. Kumar, and G. P. Raghava, "Computing wide range of protein/peptide features from their sequence and structure," *bioRxiv*, p. 599126, 2019.

[112] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. De Hoon, "Biopython: Freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[113] S. C. M. Pereira, *Building a database and development of a Machine Learning algorithm to identify and characterize viral Fusion Peptides*. Master dissertation, Minho University, 2019.

[114] D. Eisenberg, E. Schwarz, M. Komaromy, and R. Wall, "Analysis of membrane and surface protein sequences with the hydrophobic moment plot," *Journal of Molecular Biology*, vol. 179, no. 1, pp. 125–142, 1984.

[115] K. Weise and J. Reed, "Fusion peptides and transmembrane domains of fusion proteins are characterized by different but specific structural properties," *ChemBioChem*, vol. 9, no. 6, pp. 934–943, 2008.

[116] H. Yao, M. W. Lee, A. J. Waring, G. C. Wong, and M. Hong, "Viral fusion protein transmembrane domain adopts β-strand structure to facilitate membrane topological changes for virus-cell fusion," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 35, pp. 10926–10931, 2015.

[117] M. O. Dayoff, R. M. Schartz, and B. C. Orcutt, "A Model of Evolutionary Change in Proteins," *ATLAS OF PROTEIN SEQUENCE AND STRUCTURE*, pp. 345–352, 1978.

[118] A. Ashkenazi and Y. Shai, "Viral Fusion Peptides," *Handbook of Biologically Active Peptides*, no. chapter260, p. 1904, 2013.