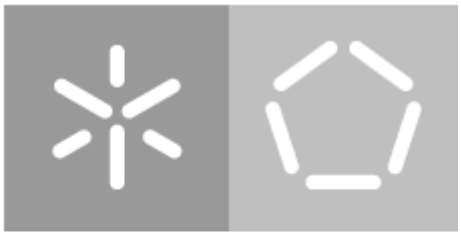
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Sara Catarina Monteiro Pereira

Building a database and development of a Machine Learning algorithm to identify and characterize viral Fusion Peptides

January 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Sara Catarina Monteiro Pereira

Building a database and development of a Machine Learning algorithm to identify and characterize viral Fusion Peptides

Master dissertation
Master's Degree in bioinformatics

Dissertation supervised by:

Diana Andreia Pereira Lousa, Instituto de Tecnologia Química e Biológica António Xavier, Universidade Nova de Lisboa

Isabel Cristina de Almeida Pereira da Rocha

January 2019

ACKNOWLEDGEMENTS / AGRADECIMENTOS

A concretização desta dissertação contou com contributos importantes, de forma direta ou indireta, de um grupo de pessoas às quais gostaria de expressar o meu agradecimento.

Em primeiro lugar, gostaria de agradecer ao Professor Claudio Soares pela oportunidade de desenvolver este projeto no Instituto de Tecnologia Química e Biológica António Xavier (ITQB), da Universidade Nova de Lisboa.

Quero também agradecer a todo o grupo de modelação de proteínas do ITQB, particularmente às minhas orientadoras Diana Lousa e Isabel Rocha, por todos os conselhos e orientações, sobretudo nos momentos com menos rumo.

Aos meus colegas de laboratório, que se tornaram amigos ao longo deste percurso, por me terem acolhido tão bem, especialmente a Mariana, a Joana e o Tiago, pela paciência que tiveram comigo, pela boa disposição, e por todo o apoio nos momentos mais difíceis. Não tenho palavras que cheguem para agradecer tudo o que fizeram por mim!

Aos meus amigos de sempre, por confiarem em mim e no meu trabalho, por me apoiarem nas decisões mais importantes e por me acompanharem em todos os momentos. Obrigada por esta amizade tão bonita e tão valiosa, que continuem presentes até ao fim.

Às minhas *roommates*, Telma e Catarina, por fazerem de Oeiras a minha segunda casa.

Por fim, quero deixar o meu agradecimento mais importante, á minha família: ao meu pai, à minha mãe, ao meu irmão e às minhas avós. Ao meu pai por trabalhar incessantemente para que nunca me faltasse nada, à minha mãe por ser o meu pilar e o meu maior exemplo, ao meu irmão por me mostrar que mesmo sem nada se tem tudo, e às minhas avós por nunca terem deixado de tomar conta de mim. Obrigada por acreditarem em mim e nas minhas capacidades, mesmo quando eu não acreditava, por todos os bons conselhos que me dão e por me ensinarem a saber viver. Obrigada por todo o esforço que fizeram para me trazer até aqui e por me terem dado um dos bens mais preciosos que possuo.

Espero que esta etapa que agora termino possa, de alguma forma, retribuir todo o apoio e carinho que me deram.

RESUMO

Os péptidos de fusão têm um papel importante no mecanismo de fusão viral. Estes péptidos são segmentos de proteínas de fusão que incluem domínios hidrofóbicos conservados absolutamente indispensáveis à atividade fusogénica de glicoproteínas de diversas famílias de vírus. É intrigante que cada vírus tenha um péptido de fusão diferente, o que dificulta a identificação de padrões que os caracterizem, mesmo que o desenvolvimento de novos fármacos dependa do conhecimento detalhado sobre as propriedades dos mesmos. A maioria dos estudos feitos nesta área incidem principalmente sobre os vírus Influenza, VIH e os péptidos de fusão dos retrovírus, contudo não é possível inferir informação para outros vírus devido às diferenças ao nível sequencial.

Tendo em conta todos estes factos, *machine learning* pode ser uma boa ferramenta para revelar padrões que estejam mais impercetíveis à primeira vista que caracterizem péptidos de fusão. Para criar modelos capazes de distinguir claramente um péptido de fusão de uma outra sequência, recorrendo aos aminoácidos presentes na sua sequência, é necessário usar informação que esteja bem anotada e revista. Atualmente, a informação relativa a estes péptidos encontra-se dispersa por várias bases de dados, não existindo assim nenhum local onde a informação esteja centralizada e completa.

No âmbito desta dissertação fez-se uma pesquisa exaustiva sobre péptidos de fusão, resultando em 468 sequências para 207 vírus de um universo de 255 vírus, usando com confiança 111 dessas sequências para efeitos de *machine learning*.

Neste trabalho foram treinados oito modelos de *machine learning* diferentes, e testados usando *5-fold cross-validation*, em diferentes *datasets* de forma a identificar e classificar péptidos de fusão. Para comprovar a utilidade dos modelos, este foram usados em três *datasets* diferentes compostos por sequências retiradas da UniProt e do NCBI. O conjunto de modelos final obteve uma percentagem de exemplos corretamente classificados e *recall* a rondar os 90 %. Estes resultados são promissores na medida em que prevêem corretamente a região mais provável do péptido de fusão, dentro de uma sequência de uma proteína de fusão, resultados que podem ser deveras proveitosos para investigadores desta área científica.

Palavras Chave: Péptidos de Fusão, Fusão Viral, Proteínas de Fusão, *Machine Learning*, Accuracy.

ABSTRACT

Fusion Peptides (FPs) play an important role in viral fusion. They are segments of fusion proteins that include conserved hydrophobic domains absolutely required for the fusogenic activity of glycoproteins from divergent virus families. FPs from different viruses are very different, which is intriguing and makes it difficult to find patterns that could characterize them. However, the development of therapeutics targeting fusion peptides requires a detailed knowledge about their properties.

Most of the studies made in this field were more focused on Influenza, HIV and all retroviruses fusion peptides, but one cannot generalize that information for all viral families, since they are different even at the sequence level. Hence, machine learning can be a good tool to unveil hidden patterns that characterize these peptides. Creating a model capable of separating fusion peptides (positive cases) from non-fusion peptides (negative cases) using their amino acid (AA) sequence as the basis for generating features, requires the usage of well annotated and reviewed proteins. Currently, the information about these peptides is very dispersed and there are no complete databases available to access and use this data.

In the scope of this dissertation, an extensive search on these fusion peptides was performed, which resulted in 468 sequences found for 207 out of 255 viruses. From that universe 111 sequences, with experimentally validated FPs were used in subsequent analysis. Multiple alignments and phylogenetic trees analysis suggested clusters per class and per family, which led to consensus sequences per virus family.

For this work, eight different machine learning models were trained and tested, using a five-fold cross validation process, on different datasets to identify and classify fusion peptides. To prove the value of the developed models, three different datasets composed by well annotated sequences from UniProt and NCBI were used. Ensembles of the created models using one dataset showed good overall performance with scores of accuracy and recall above 90 %. These are promising results on predicting the most plausible regions where the FP is located within an entire fusion protein sequence, which can be very useful in future research.

Keywords: Fusion Peptides, Viral Fusion, Fusion Proteins, Machine Learning, Accuracy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Acronyms

AA – Amino Acid

Ala – Alanine(s)

ANN – Artificial Neural Network

ATR – FTIR – Attenuated total reflectance Fourier-transform Infrared Spectroscopy

AUC – Area under the curve

BLAST – Basic Local Alignment Search Tool

CV – Cross Validation

DB – database

dPABBs – design Peptides Against Bacterial Biofilms

FN – False negative

FPs – Fusion peptides

FP – False positive

FProt – Fusion protein

FTIR – Fourier-transform Infrared Spectroscopy

FRET – Förster resonance energy transfer

Gly – Glycine(s)

HA – Haemagglutinin

HIV - Human immunodeficiency virus

KNN – K -nearest neighbours

LOO – Leave One Out

LR – Logistic Regression

MAD – mean of absolute deviation

MCC – Mathew's Correlation Coefficient

ML – machine learning

NB – Naive Bayes

NMR – nuclear magnetic resonance

RF – Random Forest

RMSE – square root of the mean of SSE

SFV – Semliki forest virus

SSE – sum of square errors

SVM – Support Vector Machine

TBEV – tick-borne encephalitis virus

VDM – Value difference metric

TN – True negative

TP – True positive

Wt – wild type

1

INTRODUCTION

1.1. Motivation

Enveloped viruses, such as influenza and HIV, are coated by an outer membrane and to infect the host cell, these viruses need to fuse the viral and host membranes. This is accomplished through the action of fusion proteins located on the virus surface. The fusion peptide (FP) is one of the most relevant players in the fusion process [1, 2]. This segment of the fusion protein inserts in the host membrane and has an active role in promoting fusion. The FP is a very promising drug target, since it is conserved within a viral species and is vital for the infection process. As an example, antibodies against dengue virus target this region [3].

All FPs share common characteristics, which are determinant for their function: they are hydrophobic, rich in Glycines (Gly) and Alanines (Ala) residues, contain aromatic residues and are usually conserved within a species (mutations frequent lead to a loss of function) [4, 5]. Apart from these general characteristics, FPs from virus belonging to different families can be quite diverse [6]. Some FPs (e.g. Influenza and HIV) are located at the N-terminal tip of the fusion protein, whereas the peptides of other viruses (e.g. dengue and Ebola) are internal fusion loops [6]. The peptides from different families are also quite different at the sequence and structure levels. Although several studies focusing on different FPs have been performed, as far as we know there is no systematic and global analysis of viral fusion peptides and the available information is very dispersed. Relational Databases are a more organized way to collect and access this information.

Additionally, it is not clear how peptides with such distinct characteristics play a common role in membrane fusion. To insert and perturb the host membrane, these peptides need to have specific properties, coded in their sequence and a machine learning algorithm can be used to uncover these patterns. This type of approach was successfully applied to similar problems regarding anti-microbial peptides [7, 8] using physicochemical descriptors such as charge,

hydrophobicity and specific sequence features [8]. The application of a similar approach to viral FPs is very promising, since it is a similar problem. The features identified as relevant by the Machine Learning (ML) algorithm can be used to identify, within the entire fusion protein sequence, the part that corresponds to the FP.

1.1. Document's Structure

This document is structured as follows:

Chapter 2

State of the art

Introduction to the biological problem, description of the previous knowledge on viral fusion peptides and the viral fusion mechanisms as well as machine learning concepts and models. Introduction to ensemble and feature selection methods and model evaluation processes, as well as tools and databases used for fusion peptides analysis.

Chapter 3

Methods

Overview and analysis of the collected data to create the datasets and build the database. Description of the methods used for dataset pre-processing and feature selection, as well as the machine learning models, ensemble methods and statistics used to evaluate the models.

Chapter 4

Development

Description of the code developed in the thesis with associated workflow.

Chapter 5

Results and Discussion

Presentation of the main results generated in the thesis followed by their discussion.

2

STATE OF THE ART

2.1. Fusion Peptides (FPs) – Key players in viral membrane fusion

Viruses are infectious agents that replicate only within the cells of living hosts, mainly bacteria, plants, and animals. They are usually composed of an RNA or DNA core, a nuclear membrane, and, in more complex types, a surrounding envelope (plasma membrane). Enveloped viruses (e.g. Influenza, HIV, Dengue) have viral envelopes covering their genetic material, which typically derive from portions of the host cell membranes. Glycoproteins on the surface of the envelope are responsible for identifying and binding to receptor sites on the host's membrane and inducing membrane fusion, allowing the viral genome to enter and infect the host. This process is collectively known as “viral entry” [3, 4] and one of its key steps is the fusion between the host and viral membranes. Fusion is a complex process, whose main steps are common to all enveloped viruses, although there are differences in the details of the process among different viruses [4]. The fusion process is catalyzed by proteins known as fusion proteins and one particular region of these proteins – the fusion peptide – has a determinant role in this process. This peptide segment inserts into the host membrane during the fusion process and has membrane-perturbing activity [9].

Many fusion proteins are C-terminal fragments of a larger precursor (eg: HA2 fragment of influenza virus hemagglutinin; gp41 fragment of HIV Env) [9] and the mechanism by which fusion proteins mediate membrane fusion is a complex process that involves several segments of these proteins. First, the fusion protein opens up when exposed to a given stimulus (e.g. pH drop, binding of a ligand) and forms a bridge between the two bilayer membranes. Usually a C-terminal transmembrane region holds the fusion protein in the viral membrane. The fusion peptide (located ate the N-terminal fragment of the fusion protein or internally – depending on the virus) connects and interacts with the host membrane [1]. The long central helix breaks, and the fusion protein’s segment between the break and the membrane reconfigures so that it runs back along the central coiled-coil. Ultimately, the fusion peptide and the C-terminal transmembrane anchor are drawn

together – along with the two membranes in which they are connected to, creating a fusion pore [1] (see **Figure 1**).

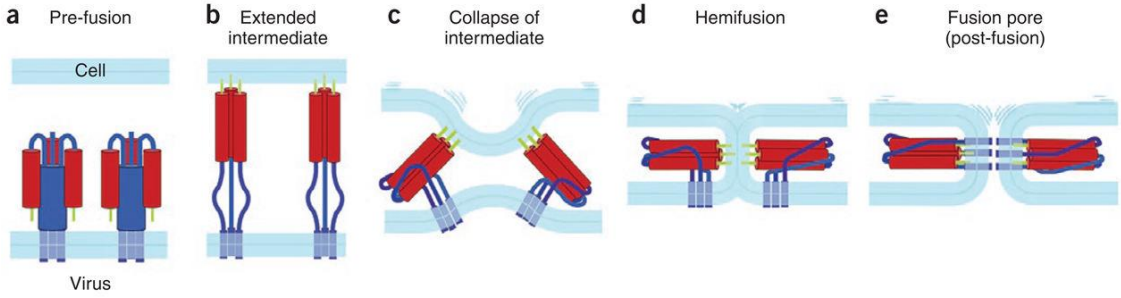


Figure 1 Schematic representation of the sequence of events in membrane fusion promoted by a viral fusion protein.

(a) The protein in the pre-fusion conformation, with its fusion peptide or loop (light green) held. Some features of specific proteins are not represented (e.g.: displacement of the N-terminal fragment of proteins that are cleaved from a precursor or the dimer-to-trimer rearrangement on the surface of flaviviruses).

(b) The protein opens up, extending the fusion peptide or loop to interact with the target bilayer. The part of the protein that bears the fusion peptide forms a trimer cluster.

(c) 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, so that at any point in the process they can extend to different distances along the trimer axis, and the entire trimer can bow outward, away from the deforming membrane.

(d) When collapse of the intermediate has brought the two bilayers into contact, proximal leaflets merge into a hemi-fusion stalk.

(e) As the hemifused bilayers open into a fusion pore, the final zipping up of the C-terminal segments breaks the refolded trimer into its fully symmetric, post-fusion conformation, preventing the pore from resealing [9].

There are at least four distinct mechanisms (pH, binding to another surface protein, temperature or fusion protein cleavage) by which viral fusion proteins can be triggered to undergo fusion inducing conformational changes [1]. Despite this diversity, all characterized viral fusion proteins convert from a fusion-competent state (dimers or trimers, depending on the class) to a membrane-embedded homotrimeric prehairpin, and then to a trimer-of-hairpins. Additionally, all fusion proteins contain a fusion peptide (FP), which inserts into the host membrane during fusion.

Three distinct classes of viral fusion proteins have been identified based on structural criteria. **Class I** fusion proteins, observed in influenza virus, HIV and SARS virus, and other viruses, are characterized by a trimeric assembly of α-helical coiled coil hairpins in the post-fusion state (see **Figure 2**) [4].

These fusion proteins usually require proteolytic processing into two subunits (e.g., influenza HA, paramyxovirus F). For some viruses (e.g., Ebola virus GP) processing into the two subunits occurs for the wild-type (Wt) protein, but is not essential for infection [10]. Some coronavirus S precursors (e.g., MHV) are proteolytically processed during biosynthesis, whereas others (e.g., SARS) are not [11].

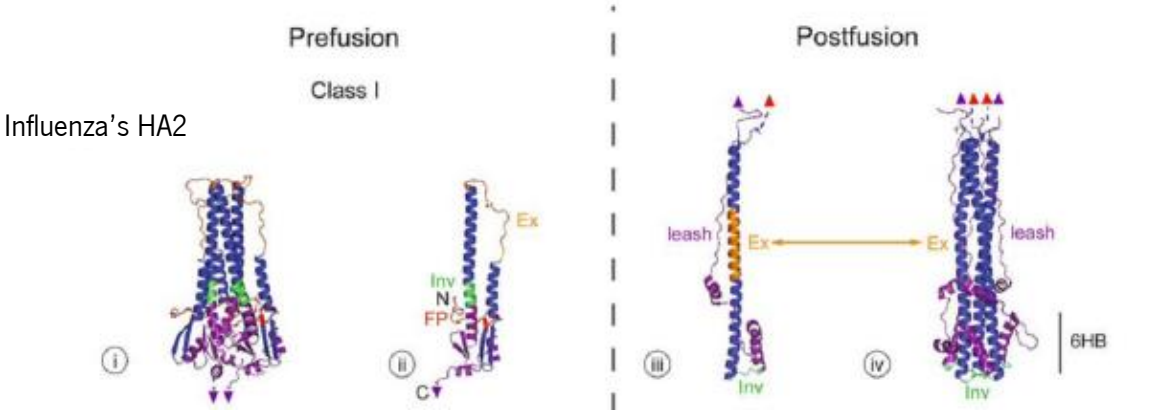


Figure 2 Influenza's and Paramyxovirus' fusion protein structure, in their pre-fusion and post-fusion states. The post-fusion trimer (iv) contains a coiled-coil superficially reminiscent of that in its pre-fusion form. Unlike those in the pre-fusion form, the central helices (dark blue) are longer, packed more tightly along their hydrophobic faces, and are proximal to the fusion peptide (red triangles). The outer helices (purple), which pack against the base of the central coiled-coil, are part of the C-terminal domain connected to the transmembrane domain (purple triangle). This structure, in which the outer C-helices pack against the central N-helical coiled-coil, is referred to as the six-helix bundle (6HB in iv) and is a defining feature of the post-fusion structures of all Class I fusion proteins. A linker C-terminal to the 6HB connects to the transmembrane domain and has been referred to as the leash. Packing of the leash into a groove along the central coiled-coil is required for fusion, as is capping of the coiled-coil.

Class I fusion proteins are characterized for being metastable on the virion and perpendicular (projected as a spike) to the viral membrane. The major secondary structure of the native fusion subunit is α-helical, and the oligomeric structure is a trimer, as well as the oligomeric structure of the fusion-active form. However, the structure of the post-fusion form is a trimer-of-hairpins (central α-helical coiled-coil, 6HB, in **Figure 2** IV). In class I native fusion proteins, the fusion peptide is buried in the subunit interface. In the primary sequence this fusion peptide is located at or near the N-terminus [12].

On the other hand, **class II** fusion proteins, found in flaviviruses and alphaviruses, are categorized as trimers of hairpins composed of β-sheets in the post fusion state **Figure 3** [13].

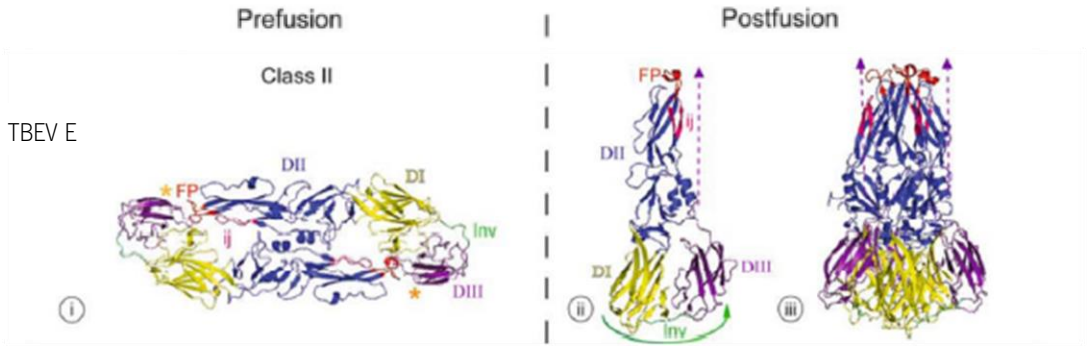


Figure 3 The crystal structures of pre-fusion and post-fusion forms of the TBEV E protein. The pre-fusion E protein is an antiparallel homodimer (i, view from above). It consists of three domains, consisting almost entirely of β-sheet structure. It is primarily contacts between the subunits in Domain II (dark blue) that maintain the homodimer. This domain also contains the fusion peptide loop (FP, red) and the ij loop (ij, pink), which play critical roles in target membrane binding. The fusion loop peptide is not exposed in the native structure, being masked by nearby residues in Domain I (yellow) and Domain III (purple) of the opposite subunit.

These class II fusion proteins consist primarily of β-sheet structure with internal fusion peptides formed as loops at the tips of β-strands. They are associated with a chaperone protein (p62 for SFV E1 and prM for TBEV E), which is cleaved during, or soon after, viral assembly so that the fusion protein generates a competent form. Similarly to class I fusion proteins, class II FProt are also metastable on the virion. However, unlike class I, they are oriented parallel (close to) the viral membrane. These proteins are dimers in the prefusion conformation, whereas the oligomeric structure of the fusion-active form is a trimer, and the structure of the post-fusion form is a trimer of hairpins (mainly β-structure). In the native fusion protein, the FP is masked in the trimer interface, at the tip of the extended β-strands, and its location in the primary sequence is internal [12].

A **class III** fusion protein, found for example in vesicular stomatitis virus and herpes simplex virus, is also characterized by trimers of hairpins, although formed by helical coiled-coil and β-sheets structures [4]. The pre-fusion form of VSV G is a trimer, but the trimer interface is small. In contrast to those in the pre-fusion conformations of all other fusion proteins known to date, the fusion loops (red) are located on the exterior side of the structure, not protected at an interface. Upon acidification, a series of conformational changes occur in VSV G that reposition the fusion loops (red) into the vicinity of the target membrane. A second series of conformational changes then bend the protein back, reorienting the C-terminal portion anti-parallel to the N-terminal segment, thereby bringing the viral and target membranes together. (see **Figure 4**).

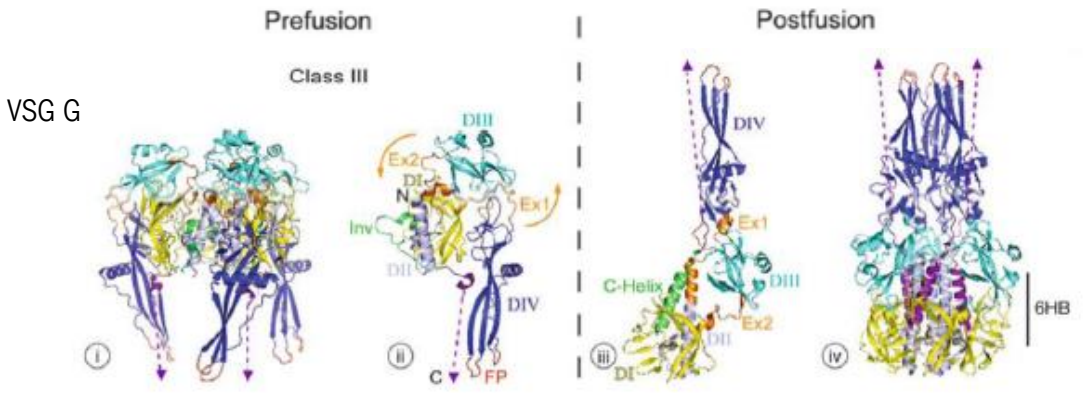


Figure 4 Crystal structures of the neutral (i and ii) and low pH (iii and iv) forms of the VSV G ectodomain.

In the first step, conformational changes occur in two regions, Ex1 and Ex2 (orange in i, ii, and iii). Each region has two parts, one is an unstructured linker, the other has helical structure. During the conformational change the unstructured linker of Ex1 becomes helical and the helical residues become unstructured, resulting in movement of the fusion domain (DIV) approximately 90°. The motion of DIV is completed by changes in Ex2, in which linkers between DII (blue-gray in ii) and DIII (cyan), become helical, extending each of the two DII helices (blue-gray and orange in iii). The result is the rotation of both DIII and DIV such that the fusion loops (red) are now near the target membrane. Finally, inversion of the C-terminal

stem is accomplished by additional structural rearrangements in Domain II. In particular, an unstructured loop (Inv, green in ii) becomes an α-helix, which we refer to in Figure 6D as the "C-helix" (green in iii), that is oriented antiparallel to the core structure. Consequently, the "C-helix" packs against the now elongated helix of Domain II (blue-gray and orange in iii), bringing the C-terminus and viral membrane into proximity with the target membrane, thereby facilitating fusion [12].

Class III G fusion proteins' conformational changes of some rhabdovirus are hypothesized to be reversible. The pre-fusion and post-fusion states are in thermodynamic equilibrium, with the equilibrium shifted towards the post fusion state at low pH [14]. This contrasts with most of the other viral fusion proteins, which are metastable and irreversibly inactivated (lose the capacity to mediate fusion with a subsequently presented target membrane) if triggered in the absence of a target membrane. Also, class III fusion proteins do not require proteolytic processing to generate a fusion competent form and are perpendicularly orientated towards the viral membrane. The oligomeric structure of the fusion protein's native form is a trimer as well as its fusion-active form, while the post-fusion structure is a trimer of hairpins (central α-helical coiled-coil and significant β-structure). Regarding the fusion peptide's location, in the native form it is exposed at the tips of the extended β-strands, whereas in primary sequence it is located internally, containing two loops found at the tips of two neighboring β-strands.

The fusion peptide (FP) is one of the most relevant players in the fusion process [1]. This segment of the fusion protein inserts in the host membrane and has an active role in promoting fusion. Interestingly, synthetic FPs are able to induce fusion of model membranes (vesicles) even in the absence of the rest of the protein and, therefore, they can be used as modes to study fusion *in vitro*.

The FP is a very promising drug target, since it is conserved within a viral species and is vital for the infection process (e.g. antibodies against dengue virus target this region) [5, 15]. All FPs share common characteristics, which are determinant for their function: they are moderately hydrophobic, rich in Gly and Ala residues, contain aromatic residues and are usually conserved within a species (mutations frequent lead to a loss of function) [5, 6]. Apart from these general characteristics, FPs from virus belonging to different families can be quite diverse [5]. Some FPs (e.g. Influenza and HIV) are located at the N-terminal tip of the fusion protein, whereas the peptides of other viruses (e.g. dengue and Ebola) are internal fusion loops [5]. The peptides from different families are also quite different at the sequence and structure levels. The influenza FP is helical in lipidic environments [16, 17], whereas the HIV FP tends to adopt β-sheet structures [18], although it can become helical depending on membrane composition [19]. Other FPs, such as the one from

dengue virus, which only has 14 residues, do not have a defined secondary structure [20]. It is not clear how peptides with such distinct characteristics play a common role in membrane fusion.

The evidence that one would like to know is the conformation of the fusion peptide segment of a viral fusion protein in the presence of membranes and it's behavior in different environments [6]. The road to obtain this specific data can go many ways but it has two main approaches, experimentally or in silico. If one would like to experimentally get this information one can try to crystalize this peptide in the absence of membranes and determine its structure by X-ray crystallography or to study by NMR the conformation of a small synthetic fusion peptide, either in the presence or absence of membranes [6]. However, the secondary structures of many fusion peptides have been determined using less specific spectroscopic techniques, based on the properties of the amide chromophores, such as circular dichroism (CD) – CD spectra are typically recorded from peptides that are bound to small unilamellar vesicles (peptides and vesicles should be well dispersed (non-aggregating) in order to avoid light-scattering artifacts) [6, 9].

In addition, to determine the orientation of a helical peptide in a membrane, one of the most popular techniques is the attenuated total reflectance Fourier transform infrared spectroscopy (ATR-FTIR) [6]. In this technique, planar oriented membranes are deposited on the flat ATR crystals, providing ATR-FTIR spectra for oriented samples as required for orientation studies with linearly polarized light. Results can differ, if membranes with or without peptides are deposited from organic solvents or if they are self-assembled and the peptides added from aqueous solutions, or if the peptides' conformations are measured in bilayers that are submerged in an aqueous buffer, partially hydrated by water vapor, or completely dry [9].

Most of the studies made on the viral entry mechanism are more focused on Influenza virus [21]. For this virus it is possible to induce a fusion event by lowering the pH [22]. Under these circumstances, Fluorescence Resonance Energy Transfer (FRET), assays effectively measure the kinetics of fusion and have been used to understand the effects of mutations and antiviral drugs [22]. In this technique, fluorescent probes are incorporated into the virus membrane and mix with the target cell, becoming diluted into the target cell itself or into the liposome membrane [22]. The resultant change in fluorescence gives a real-time energy measure of the fusion process [22]. Interestingly, a test using a mutant containing one amino acid substitution (T471P) in the fusion peptide of the Murine Leukemia Virus envelope protein gave no signal regarding function. However, even in the absence of its protein, the fusion peptide still bounded to cells normally [22].

The experimental approach can lead to different results when different techniques are applied. The problem is not with the techniques themselves, but rather because of different methods of sample preparation that are commonly employed by practitioners of each technique.

2.2. Machine Learning

As mentioned above, the dependence of enveloped viruses on FP during infection process make FP a promising drug target [4]. However, many fusion peptides sequences are still not available and existent information is dispersed and cannot provide a systematic and global analysis of FP's. Most of the studies made in this field were more focused on Influenza, HIV and all retroviruses fusion peptides, but one cannot generalize that information for all viral families, since they are different even at the sequence level. Hence machine learning can be a good tool to unveil hidden patterns that characterize these peptides.

2.2.1. Concepts and Definitions

Machine Learning (ML) is a field of artificial intelligence/ computer science/ statistics which provides computers the ability of predicting the outcome of an event or situation, and eventually improving its results with time, simulating the gain of experience that is observed in real-life learning processes [23].

In order to help the reader to understand the concept of ML, the next paragraphs contain a summary of some ML concepts and definitions [23, 24].

An **attribute** denotes a feature that describes instances. They can be categorical or continuous. Categorical attributes take values from a set with a finite number of discrete values and can either be nominal, indicating that there is no order between the values (e.g. names and colors), or ordinal (e.g. little, medium, big), where an order can be identified. Continuous attributes take values from a domain which is a subset of real numbers and can take any value within a range (e.g. height, time) [14].

Attribute values **-** For example, if "names" is an attribute, "Mary" is a value of the attribute "names".

Input and output attributes - Input attributes are the attributes that are going to be used to make a prediction of the output attribute.

An **Instance** (or example) is an object composed by a set of input attributes (and possibly the respective output attribute). Instances represent individual cases of the concept to be learned.

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, and the remaining represent the input attributes.

Training Dataset - Group of instances used to learn the best model for a specific data.

Test Dataset - Group of instances used to calculate different statistics on a generated model in order to evaluate the performance of the model.

A **model** can be defined as a function that given an instance´s input attributes predicts its output attribute.

An **algorithm** defines a process that, given a training data set and some pre-chosen criteria, chooses a specific model.

2.2.2. Supervised vs Unsupervised learning

ML algorithms can be supervised or unsupervised. Supervised learning is a method that uses a given input labelled data (training data set) to generate a model, inferring the hidden relationship between that data. Using that model, the result of the class label (out-attribute) can be predicted. This method is really useful when it comes to predicting the class label of a data set with hidden phenomena attached in unfamiliar or unobserved data instances [23]. The errors associated with the prediction can be minimized based on the quality of the used training dataset. Small datasets (20-30 examples) are generally a poor choice for these algorithms. Data sets that contain many similar examples can be a bad choice as well, since they can cause overfitting of the model. The best choice for a data set would be a data set that is a good representation of all possible instances (generalized) so that the model can have an example of each possible outcome [23].

Unsupervised learning algorithms are based on grouping instances without a prespecified dependent attribute. These techniques are designed to discover hidden structures in unlabeled datasets, in which the desired output is unknown. The general approach to learning involves training through probabilistic data models. The goal of ML in this case is to hypothesize representations of the input data for efficient decision making, forecasting, and information filtering and clustering. Clustering (k-means and hierarchical clustering), Principal Component Analysis

(PCA) and Expectation–maximization (EM) are some examples of unsupervised learning algorithms [25].

2.2.3. Development of a ML algorithm

When a ML algorithm is being developed, seven major steps should be considered (**Figure 5**). The first one is to collect the data, where a subset of all available data attributes that might help in resolving the problem are selected. The second step is to process the data, making it comprehensible. Then the data is transformed, by feature scaling, decomposition, or aggregation (combining multiple instances into a single feature). The next step is to train the algorithm, where the training and testing datasets from the data previously transformed are selected. The algorithm is then trained (fourth step) using the training dataset, extracting the knowledge or information on that dataset, which allows one to choose the appropriate parameters to use in the model. This model can then be used to predict the output attribute of other similar data.

The fifth step is where the algorithm, using the test dataset, is evaluated. In this step a model evaluation is performed in terms of effectiveness and performance. Giving the input attributes of the test dataset to the model, and hiding the output attribute, it predicts the output attribute for each instance. Then calculations regarding different statistics about the model's performance are made, comparing the two results, the known and the predicted, and the created model can eventually improve by using a different dataset or improving the old one. The seventh and last step is to apply the validated model, making reliable predictions on new datasets with unknown out-attributes.

Figure 5 Seven step process for developing a machine learning algorithm.

2.2.4. Algorithms

Some of the major models and algorithms used in ML, that is k-nearest neighbours (KNN), Naïve Bayes (NB), linear and logistic regression, decision trees, artificial neural networks (ANN), and SVMs (Support vector machines) are reviewed below.

2.2.4.1. K-nearest neighbours (KNNs)

KNNs is an instance-based learning method, which means that it does not generate a model function. Instead, it stores the training set and uses it when a new prediction needs to be made, being called a lazy learning method for that reason, since it delays the learning process [26]. KNN has two simple ways of making predictions depending on the type of problem (classification or regression). In a classification problem, the algorithm will simply find the k training examples most similar to the new example to be predicted and the final prediction will be the most common class on those k training examples. In a regression problem, the algorithm will predict the final result as the mean of the nearest neighbor values.

To calculate the nearest neighbors, a distance function must be used. For continuous (e.g. height) or linear discrete (e.g. number of children) features, Euclidean distance or Manhattan distance are usually used. For linear symbolic features (e.g. symptoms) the most common way of calculating their distances is to assume the value 1 for a different feature and the value 0 for an equal feature, and then calculate the examples distances using the Euclidean or Manhattan distance [26]. However, the previous method does not consider that some linear symbolic features may have an order, so a different distance function must be used. Value difference metric (VDM) considers two features to be closer if they have more similar classifications, thus ranking the features and providing a better calculation of the neighbors [27].

2.2.4.2. Naïve Bayes (NB)

NB is a simple algorithm that uses relative frequencies to estimate the probability of an example to present a certain result [25], assuming (although naively) the attributes are independent and cannot be differentiated in terms of importance.

This algorithm calculates the L (likelihood) by multiplying the relative frequencies of each of the examples attribute values with the relative frequency of the class. The final prediction will be

the one that presents the highest L value. To get the probability of a given class to be predicted, its L value must be divided by the sum of all L values for all classes.

2.2.4.3. Linear Regression

Linear Regression tries to model the relationship between a dependent variable and a set of independent variables, generating a linear equation that fits the data [26, 27]. Linear regression involves finding a best-fitting set of coefficients minimizing the Sum of Squared Errors (SQE).

2.2.4.4. Logistic Regression (LR)

LR is used in classification problems, where it models the relationship between a set of independent variables and a dependent variable (binary class), predicting the probability of occurrence of the dependent variable [23]. In this process, a logistic function is calculated. With this function, estimating the probabilities of a given class can be performed. To minimize the error function, the minimization of a loss function is also conducted as in linear regression. Logistic regression can also be used in classification problems with more than 2 classes by creating a model for each class.

2.2.4.5. Decision Trees

This method is used for classification problems. It generates classifiers by synthesizing a model based on a tree structure [26]. This tree is composed by n nodes and each node corresponds to an input attribute to be tested. On each node, n possible branches come out corresponding to the values or conditions the attribute can present, leading to n different nodes. If a leaf is reached, the information on that leaf corresponds to the output attribute´s value (or class). To make a prediction on a new example, each attribute is tested on its specific node, starting from the root. After the root´s specific attribute has been tested on the condition, it follows the branch that suits that condition, getting to another node. This process is repeated until a leaf is reached, and the end prediction is obtained [23].

2.2.4.6. Regression Trees

Regression Trees are a variation of decision trees that can be used in regression problems. They are adaptations of decision trees where the leafs instead of class values are composed of a numeric value. M5 is an algorithm that tackles one of the regression trees fundamental problems: the fact that it can only assign a constant value to its leafs. On this algorithm each leaf is composed by a linear model allowing the calculation of the out-attribute as a linear function of the in-attribute´s values [26].

2.2.4.7. Artificial Neural Networks (ANNs)

ANNs represent attempts of simulating the human brain's neurons [23, 26]. These simulated neurons (nodes), like normal neurons, receive inputs and give outputs to other simulated neurons. There are two major types of ANN: Feedforward ANNs (no cycles) and Recurrent ANNs (with cycles) and both can be represented by a graph. In feedforward ANNs, this graph can be divided into an organized disposition with 3 layers: an input layer, a hidden layer, and an output layer. A node (y) has n nodes connected to it (x_n) with different output values ($X_1, \dots, X_n$) and each connecting has a weight associated ($W_{y1}, \dots, W_{yn}$). A node´s output value is calculated by an activation function using the activation value. The activation value of a node "y" can be calculated through the sum of all the "x" nodes output values times the connection weight of the nodes ($Av = \sum X_n \times W_{yn}$). To minimize the cost function for ANN there are many training algorithms, leading to modifications in the weights of the connections of the nodes, namely backpropagation, Rprop, Quickprop, and others [27].

2.2.4.8. Support Vector Machines (SVMs)

SVMs are used for both classification and regression problems. These models are based on creating support vectors from the dataset, which are only a subset of the total dataset calculated by an optimization step that regularizes an objective function by an error term and a constraint [23, 25, 26]. Regarding classification problems, the support vectors are used to calculate a hyperplane that separates the data into two classes, always maximizing the margins of the hyperplane. For regression problems, the data will lie within a "tube" around the hyperplane. However, some data that cannot be divided into the two classes by a linear hyperplane or does not

fit a linear hyperplane tube, so a more complex polynomial function must be applied. In that case kernel methods are used, like polynomial, Gaussian and spline kernels, and can be configured using different parameters.

This type of approach has been applied to antimicrobial peptides with great success. The models predicted regions where antimicrobial peptides were inserted based on their whole amino acid composition, selected residue features and positional preference of the residues. Training datasets showed a maximum accuracy of 95.24%, sensitivity of 92.50%, specificity of 97.73%, and Mathew's Correlation Coefficient (MCC) of 0.91 [8].

2.2.5. Ensemble methods

Ensemble methods consist in developing learning algorithms that generate an ensemble of different models for a given problem [27]. The final result is obtained by combining the individual models' results and returns a single value. To produce a better final prediction than the individual models these methods have to respect two conditions: the individual models have to be precise (they have to present better results than a random model) and be diverse (they have to make errors in different spaces of the test dataset) [27].

The most popular way to create an ensemble model for unstable induction algorithms (those that show considerable changes in the model when faced with changes in the training dataset) is to change the training dataset inputted in the algorithm, hence generating different models that will form the ensemble [27]. In this category, bagging, cross-validation and boosting are the most frequently used. Bagging is based on bootstrap, where the bootstrap sample (training dataset) will be generated by a sampling process with substitution [27]. Cross-validation involves splitting the dataset into subsets of the same size, where each model will be created using different sets of training and test data. Boosting is also based on bootstrap, but in this case after each boosting iteration a weight is applied to each training example, increasing the weight on the incorrectly predicted examples and decreasing the weight in the correctly predicted examples [27].

Another approach to create ensemble methods is to introduce random choices in deterministic models, therefore creating different models in each training. When the algorithm is already stochastic, ensemble models can be created by modifying some of the algorithm´s initial parameters, like changing the number of intermediate nodes in a neural network algorithm [27].

Depending on whether it is a classification or regression problem, the functions that combine the results of the individual models can vary. When facing a classification problem, two approaches

can be taken: a vote function or a winner-takes-all function. The prior, essentially chooses the result that was shown by most of the models, whilst the latter assumes the final result as the one shown by the model with the most confidence (assuming each model is capable of calculating the probability of the result to be correct). Regarding regression problems, a mean function that assumes the final result as the mean of the individual results or a weighted mean function that is similar to the previous but assigns a weight for each model can be used [27].

Additionally, creating hybrid systems that combine two or more learning techniques can obtain a more precise result [27].

2.2.6. Evaluating machine learning models

To evaluate the quality of a model for a given task, different error metrics must be calculated. These metrics will depend on whether it is a classification or a regression problem.

Regarding classification problems, a confusion matrix is usually calculated. For a 2 classes problem, the confusion matrix (**Table 1**) is composed by 2 rows and 2 columns, where the rows represent the desired values (first row - negative values and second row -positive values) and the columns the predicted values (first column – negative values and second column - positive values). From the predicted values, if a value is predicted as negative and its real value is negative it is called a True negative (TN), but if its real value is positive it is called a False negative (FN). Similarly, if a value is predicted as positive and its real value is negative, it is called a False positive (FP), but if its real value is positive it is called a True positive (TP).

Table 1 Example table of a Confusion Matrix.

Confusion Matrix		**Predicted values**	
		Negative	Positive
Desired values	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Giving the table above, error metrics like those below can be easily calculated.

$$\frac{TN + TP}{TN + TP + FP + FN} = Accuracy \quad (1)$$

$$\frac{TP}{FN+TP} = Recall \quad (2)$$

$$\frac{TN}{TN+FP} = Specificity \quad (3)$$

$$\frac{TP}{TP+FP} = Precision \quad (4)$$

$$\frac{TN}{TN+FN} = Negative\ predictive\ value \quad (5)$$

$$2 * \frac{precision * recall}{precision + recall} = F\ score \quad (6)$$

For a classification problem with more than 2 classes, the values are calculated as if a 2x2 confusion matrix existed for each class, whilst the "negative" values are elements of the other classes.

The best model will be the one that presents higher numbers in the Accuracy, Recall, and Specificity values. However, sometimes one must find a balance between the Recall and Specificity value because an increase in the Recall value can cause Specificity to decrease.

2.2.6.1. Receiver Operating Characteristic (ROC) curves

ROC curves are also a good way of evaluating a model, since they show the relationship between Specificity and Recall. Calculating the area under the curve (AUC) of the ROC curve gives us information about the ability of the model to discriminate between the two classes. An AUC of 1 means that the model can distinguish the two classes perfectly and an AUC of 0.5 means that the model has a 50 % chance of distinguishing the two classes correctly. If the classification problem has more than 2 classes, ROC curves must be applied for each class, being the global AUC given by a weighed mean of the frequencies of each class [27].

When a regression problem comes along, the error metrics are calculated based on the error presented by each example (the difference between the predicted value and the real value). Three different error metrics can be considered: SSE (sum of square errors) can be calculated by the summation of the squared subtraction of the desired value (y_i) by the predicted value $(\hat{y}_i)$ as seen in (7), RMSE (square root of the mean of SSE) as seen in (8), and MAD (mean of absolute deviation) as seen in (9) [27].

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = SSE \tag{7}$$

$$\sqrt{\frac{SQE}{N}} = RMSE \tag{8}$$

$$\frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N} = MDA \tag{9}$$

For these error metrics, the model that shows lower values is more precise. A model that presents a value of 0 in this metrics is the ideal model.

For regression problems **a REC (Regression Error Characteristic**) curve can also be used, since sometimes the other metrics are not sufficient to understand the behavior of the model [27].

To accomplish more accurate evaluations of a model´s performance, **a K-fold cross validation** can be performed. The data set will be divided into K subsets, where 1 subset will be used as a test dataset and the others as the training dataset [27]. Then the desired error metrics are calculated by averaging them across all K trials. The advantage of this method is that every subset is going to be used as a test dataset one time, and as a portion of the training dataset k-1 times.

Another way of improving model's accuracy is using **Leave-one-out cross validation**, which is similar to K-fold cross validation. However, in this case K corresponds to the number of examples in the dataset - each run has a training dataset of K-1 examples and a test dataset consisting in the 1 example that was left out [27]. When compared to the K-fold cross validation, this process requires a larger amount of time to compute.

A **Bootstrap** method can also be executed on the dataset, achieving a more accurate evaluation of the model´s performance. The most used form of bootstrap considers a dataset of size n, where a bootstrap sample (training dataset) will be generated from that dataset by a sampling process with substitution [27]. The bootstrap sample created will have the same size as the original dataset but will be composed by some repeated examples. The unused examples of the original dataset will compose the test dataset [2, 7, 28].

2.2.7. Model selection

Model selection consists in choosing a model from a range of different models with different levels of complexity that were trained with the same dataset [27]. This process is accomplished evaluating the models on one or more error metrics and choosing the one with better scores. This can be difficult, since all the scores must be considered (i.e. comparing two models, a model with a bit higher accuracy score does not mean a better model if the F1 score is much lower). The main advantage of this process is to adjust the model to the complexity of the data, therefore reducing overfitting.

Since during the training of the model some randomness is induced, either it being in the division of the dataset into training and test datasets or in the model itself (stochastic models), a single comparison of the error metrics is not enough to conclude which model is performing the best. To conduct a reliable comparison of the models, a high number of simulations most be executed, each using different training and test datasets (variations of the initial dataset). The higher the number of simulations the better, since it will generate a more precise mean of the error metrics, yet requiring a high demand of computational time [27]. Taking this into account, the most common number of simulations goes between 10 and 200.

Besides the mean of each model, calculating the mean's standard deviation and the respective confidence levels, allows making a more reliable choice between the models [27]. If the comparison is between two models, a t-test can be performed. This t-test will access if the two means are significantly different from each other. Given the p-value of the t-test, if it is lower than 0,05 (considering a confidence level of 95%) the means are significantly different, otherwise the means are not significantly different.

2.2.8. Feature selection

Nowadays, due to the large amount of available data, dimensionality is a problem that many models have to face. The more in-attributes the dataset has (x) the more learning examples it needs, growing exponentially in many cases [27]. This problem creates the need to have big datasets which are difficult to save, and often hard to get.

A way to solve this problem is to reduce the number of input attributes. This can be accomplished by extracting features from the dataset (feature extraction), or selecting the most valuable features of the dataset by a process called feature selection [27]. The way of searching

the best feature is a crucial part for feature selection. Concerning ML classification problem, feature selection techniques can be grouped into 3 categories: filter methods, wrapper methods or embedded methods [29]. Filter methods select the features based only on the intrinsic properties of the data. Examples of these are univariate filter methods like Chi-square and Euclidean distance or multivariate filter methods like Correlation-based feature selection (CFS) or Markov Blanket Filter (MBF). Wrapper methods can be divided into deterministic methods and randomized methods. Examples of the first are Forward Selection and Backward Selection and for randomized methods, simulated annealing and genetic algorithms [29].

Forward selection and backward selection are examples of hill-climbing methods. In the first method, the model starts with only one feature and then starts adding more features in each iteration, and in the second method the model starts with all the features and then removes a feature in each iteration [27]. Both are greedy methods, meaning that both will stop if they find a local minimum, which may not be equal to the global minimum (best solution). Examples of Embedded methods are decision trees, weighted naive Bayes and feature selection using the weight vector of SVM [29].

2.3. Sequence analysis algorithms and tools

Sequence similarity searching aims to identify homologous sequences in databases through a process that provides additional and very important information about new sequences.

There are a lot of different homology searching tools, but the most used is the Basic Local Alignment Search Tool (BLAST) [30]. Different BLAST programs can then be chosen: nucleotide blast, protein blast, blastx, tblastn, and tblastx and a "How to BLAST" guide can be found in [31]. Their characteristics are summed up below in **Table 2**.

Table 2 Characteristics of the different BLAST programs.

BLAST program	Description	Searching algorithms
nucleotide blast	Search a nucleotide database using a nucleotide query.	- BLASTn (slightly similar sequences) - mega BLAST (highly similar sequences) - discontiguous mega BLAST (more dissimilar sequences)

BLAST program	Description	Searching algorithms
protein blast	Search a protein database using a protein query.	- BLASTp (protein-protein BLAST) - psi-BLAST (Position-Specific Iterated BLAST) - phi- BLAST (Pattern Hit Initiated BLAST) - delta-BLAST (Domain Enhanced Lookup Time Accelerated BLAST)
blastx	Search a protein database using a translated nucleotide query.	
tblastn	Search a translated nucleotide database using a protein query.	
tblastx	Search a translated nucleotide database using a translated nucleotide query.	

Many aspects of the search can be defined in BLAST, such as the database that is going to be used for the query homology search, being the Non-redundant protein sequences (nr) the most frequently used. Also, this search can be limited to a specific organism or taxonomic group or using a keyword or query size using Entrez Query. Many parameters of the algorithms selected can also be changed by the user.

The HMMER tool is a homology searching tool that can build a profile Hidden Markov model using a multiple sequence alignment given by the user using the HMMER3 tool [32]. The HMM profile can then be used to search databases of protein sequences. This tool has been successfully applied in several studies regarding fusion peptides [2, 7].

The WebLogo tool provides an easily perceivable description of sequence similarities based on multiple aligned sequences. Each logo consists of stacks of letters, one stack for each position in the sequence. The size of each stack indicates the sequence conservation at that position (measured in bits), whereas the height of symbols within the stack reflects the relative frequency of the corresponding amino or nucleic acid at that position [33]. This tool has been enhanced with additional features and options, to provide a convenient and highly configurable sequence logo generator [34].

Motifs are widespread patterns within sequences of nucleotides or amino acids that usually have biological significance, making them useful for inferring a protein's function or even to identify sequence homology. Finding motifs can be achieved by different methods including simple heuristic algorithms, expectation-maximization algorithms (E-M) like the one used in the MEME tool, Gibbs sampling and HMMs like the ones used in HMMER.

2.4. Relevant bioinformatics tools and databases

Depending on what kind of information one wants, one could choose the more appropriate database to retrieve it from. The Universal Protein Resource (Uniprot) is a database that contains reviewed annotations (Swiss-Prot) and automatically generated annotations (TrEMBL) of protein data [35].

Developed by the European Bioinformatics Institute (EMBL-EBI), the SIB Swiss Institute of Bioinformatics and the Protein Information Resource (PIR), Uniprot, and more specifically Swiss-Prot is an important source of reviewed protein data.

2.5. Relevant development environments

2.5.1. Biopython library

Biopython (web site: http://biopython.org/) has several functionalities that facilitate working with protein and nucleic acid sequences. The sequence object "Seq" belongs to this library and contains the sequence´s string and the sequence´s alphabet. It supports different methods such as finding a sequence´s complement or reverse complement.

Biopython can do sequence alignments using the module "AlignIO", extract information from different biological databases (Entrez, PubMed, SwissProt, Prosite). Its libraries also have a good set of parsers that can parse different files in different formats including FASTA, GenBank, PubMed, SwissProt, Unigene and SCOP. These parsers allow access to the information contained in the records of the file [36].

2.5.2. Computing libraries in python

Scikit-learn is a python module for solving machine learning problems that makes use of numpy, scipy and matplotib libraries (available at http://www.scipy.org/) among other useful resources (such as KNNs, decision trees, naïve bayes, linear and logistic regression and SVMs) [37].

In order to be used in the scikit-learn package for supervised learning (e.g.: SVC ANN, KNN, NB etc.), a dataset must be composed by input attribute values (a numpy array with n examples and m in-attributes) and output attribute values (a numpy array with 1 dimension of size n), then the dataset file can be easily loaded using the "genfromtxt" function from the "numpy" library.

Datasets can be divided into training and test with the “cross_validation” function, where a parameter defines the proportion of instances that is used for each category. The model is trained by fitting it to the using the “fit”. Regarding unsupervised learning problems, the scikit-learn libraries also have a variety of models, including clustering, principal component analysis (PCA), etc.

Cross validation leave-one-out or K fold cross validation can also be used with this package to evaluate the model’s performance. The scoring parameter is the error metric used by the validation function. If the error metric is omitted, it considers the default method´s estimator as the error metric, but other error metrics like the f1 error metric can be chosen. For regression problems, the scoring parameter can be replaced by the R2 error metric, mean squared error and mean absolute error.

Ensemble methods like bagging, random forests and boosting can be applied. Methods for feature selection, variance filters, univariate filters using Chi squared and linear regression are also available.

This package has many other resources, including dataset transformations and dataset loading utilities [37].

3

METHODS

3.1. Data collection and creation of a FP database

Creating a model capable of separating fusion peptides (positive cases) from non-fusion peptides (negative cases) using their amino acid sequence as the basis for generating features, requires the usage of well annotated and reviewed proteins.

The positive cases (fusion peptides) were harder to obtain, since this information is widely dispersed and there are no complete databases available to access and use this data.
Since the field of interest was enveloped viruses (as only them can have fusion peptides) a search on ExPASy was made to select the fusion peptide from all these viruses. All sequences were retrieved first from UniProt using the advanced search with the following query: the field "Family and domains > Region" was filtered using the term "fusion peptide" and at the same time the field "organism" was filtered using the term "virus". The retrieved results contained Uniprot entry, status, organism, region and protein length and sequence. Each entry was explored individually to search for the fusion peptide sequence and its residues and annotation method (eg. Sequence analysis, Similarity, Curated, etc).

A database containing the information gathered for each fusion peptide was created on csv format and includes the following:

- **Virus:** species, family and host;
- **Fusion Protein:** UniProt ID, protein name, class, description, residues, sequence and status;
- **Fusion Peptide:** Residues, Sequence, Annotation Method, Putative;
- **Comments:** where we specify virus's strains and other relevant information.

After a preliminary analysis, we found 24 different viruses with FP sequences (excluding the different strains of the same virus,). However, we detected that not all known fusion peptides were

annotated at UniProt (e.g. Influenza A virus did not present any matches and it is the most studied virus). Hence, another search was made at NCBI PubMed with the keywords “fusion peptide” and “virus species” for all species of interest, collecting all resultant references. Each reference was explored to retrieve fusion protein sequences, description and class, and FP’s sequences and their activation mechanism (eg. pH, binding to receptors, etc)

Fusion protein classes identified by references were generalized for proteins with the same description, for example, all E1 glycoproteins are class II, all G glycoproteins and gH are class III, etc [1].

The negative cases (non-FP’s) were divided in two groups: the transmembrane domains (since the TMD sequence has physicochemical features similar to the FP sequence) and amino acid sequences randomly retrieved from the fusion proteins sequence (excluding the FP part). Therefore it is important to have these two negative datasets. The transmembrane domains were obtained by filtering the UniProt database for “Transmembrane” on “Subcellular Location” and “virus” on “Organism” getting a total of 2288 sequences. The second group of sequences were retrieved from all fusion proteins, using a script that returns all possible subsequences of a given length (e.g. 20 aa) within a fusion protein sequence, excluding the fusion peptide region.

Overall, there were 467 positive cases, from 216 different species - the appendix A summarizes the information on the collected sequences.

3.2. Developing a machine learning algorithm to classify fusion peptides

As detailed above, the dependence of enveloped viruses on FP during infection process makes the FP a promising drug target [4]. However, many fusion peptide sequences are still not available and existent information is widely dispersed and cannot provide a systematic and global analysis of FP’s. Most of the studies made in this field were more focused on Influenza, HIV and all retroviruses fusion peptides, but one cannot generalize that information for all viral families, since they are different even at the sequence level. Hence machine learning can be a good tool to unveil hidden patterns that characterize these peptides.

3.3. Input attributes

To develop a good model, a good set of input attributes (features) is necessary. In this thesis, for each peptide in the training set, we generated an ensemble of **9507** physicochemical descriptors falling into the five categories listed in **Table 3** [7]. A detailed description of each descriptor is provided below. All descriptors were efficiently generated using the freely available propy Python package [38].

Table 3 Groups and categories of the Physicochemical descriptors used for the machine learning.

Feature Group	**Features**	**Nº of Descriptors**	**Nº of Descriptor Values**
Amino Acid Composition	Amino Acid Composition	1	20
	Dipeptide Composition	1	400
	Tripeptide Composition	1	8000
Autocorrelation	Normalized Moreau-Broto autocorrelation	15	240 (8*δ_{max})
	Moran autocorrelation	15	240 (8*δ_{max})
	Geary autocorrelation	15	240 (8*δ_{max})
Physicochemical composition	Composition	7	21
	Transition	7	21
	Distribution	7	105
Sequence Order features	Sequence order coupling number	2	60(2*δ_{max})
	Quasi-sequence order descriptors		100 (2*$(20 + \delta_{max})$)
Pseudo Amino Acid Composition	Pseudo Amino Acid Composition	1	30 (20 +λ_{max})
	Amphiphilic Pseudo Amino Acid Composition	1	30 (20+λ_{max})

1. **Residue composition**.

   a. Amino acid composition measures the fraction of each of the 20 natural amino acid types in the sequence,

   $$f(X) = \frac{N_X}{N}$$

   , where N_X is the number of residues of type X in the sequence of length N and $X \in \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ for a total of 20 descriptor values.

   b. The dipeptide composition measures the relative fractions of the 400 possible contiguous dipeptides observed in the sequence,

   $$f(X, Y) = \frac{N_{XY}}{(N-1)}$$

   , where the indices X and Y run over the 20 natural amino acids.

   c. The tripeptide composition measures the relative fractions of the 8000 possible contiguous tripeptides observed in the sequence,

   $$f(X, Y, Z) = \frac{N_{XYZ}}{(N-2)}$$

   , where the indices X, Y and Z run over the 20 natural amino acids.

2. **Autocorrelation.** The autocorrelation measures the similarity in the properties of any pair of amino acid residues along the peptide chain, revealing any correlated distribution of amino acid properties as a function of the separation between residues along the peptide backbone.

   a. The normalized Moreau-Broto autocorrelation, defined as a function of the separation between any pair of amino acid positions, δ, as

   $$AC_{nMB}(\delta) = \sum_{i=1}^{N-\delta} \frac{P_i P_{i+\delta}}{(N-\delta)}$$

   , where $\delta = 1 \ldots \delta_{max}$, P_i is the physicochemical property of amino acid at position i, and we make the standard choice of $\delta_{max} = 30$.

We considered 8 different amino acid physicochemical properties: the Kyte-Doolittle hydropathy index, the average flexibility index, the polarizability parameter, the free energy of solution in water, the solvent accessible surface area, the residue volume, the steric

parameter related to the van der Waals volume of the side chain and the relative mutability [39–41].

3. **Physicochemical composition.** These descriptors seek to provide a coarse-grained representation of the distribution of specific physicochemical properties along the peptide backbone. Seven physicochemical features are considered – hydrophobicity, van der Waals volume, polarity, polarizability, charge, secondary structure, and solvent accessibility, each of which is coarse grained into the three categories below.

a. The composition descriptor, C, measures the fraction of amino acids in the peptide that fall into each of the three categories for each of the seven descriptors, making a total of 21 descriptor values.

b. The transition descriptor, T, measures the fraction of pairs of contiguous residues that belong to each of the three possible combinations of different categories (i.e., category 1 and 2, 1 and 3, or 2 and 3, where the order is immaterial). Applied to each of the seven descriptors, this also yields 21 descriptor values.

c. The distribution descriptor, D, furnishes for each of the three categories five values: the fractional distance along the peptide sequence that must be traveled to encounter (i) the first residue belonging to the category, (ii) 25%, (iii) 50%, (iv) 75%, and (v) 100% of the residues belonging to the category.

Applied to each of the three categories for each of the seven descriptors, this yields a total of 105 descriptor values [39, 42–44].

4. **Sequence order features**. Similar to the autocorrelation descriptors, this class of features seeks to characterize patterns in physicochemical properties along the peptide backbone.

a. The sequence order coupling number as a function of the separation between any pair of amino acid positions, δ, normalized by sequence length is defined as

$$\tau_\delta = \frac{1}{(N-\delta)} \sum_{i=1}^{N-\delta} \left(d_{i,i+\delta}\right)^2$$

, where $\delta = 1 \ldots \delta_{max}, d_{i,i+\delta}$ is the physicochemical "distance" between the amino acid at position i and that at position $(i+\delta)$, and we make the standard

choice of $\delta_{max} = 30$ [39, 40, 45]. We consider 2 different definitions of the physicochemical distance between a pair of amino acids defined by the Schneider-Wrede context matrix and Grantham chemical distance matrix.

b. In the quasi sequence order we compute physicochemical distances according to the Schneider-Wrede context matrix and Grantham chemical distance matrix. Values of p_{k+20} in short peptides for which $k > (N-1)$ are assigned a value of zero.

5. Pseudo Amino Acid Composition. The pseudo amino acid composition $(PseAAC)$ introduced by Chou is a descriptor that characterizes the amino acid composition of a peptide while simultaneously maintaining information on sequence and length [45, 46]. The original $PseAAC$ defined by Chou associates to a peptide $PseAAC = [p_1, p_2, \dots p_{20}, p_{20+1}, \dots, p_{20+\lambda_{max}}]$, which is an ensemble of $(20 + \lambda_{max})$ numbers defined as:

$$p_i = \begin{cases} \dfrac{f_i}{\sum_{k=1}^{20} f_k + w \sum_{k=1}^{\lambda max} (N-k)_{\tau_k}}, & 1 \le i \le 20 \\ \dfrac{w\tau_{i-20}}{\sum_{k=1}^{20} f_k + w \sum_{k=1}^{\lambda max} (N-k)_{\tau_k}}, & 20+1 \le i \le 20 + \lambda_{max} \end{cases}$$

where $f_i = \frac{N_i}{N}$ is the fraction of residues of type i in the sequence of length N, and i indexes over the $i = 1 \dots 20$ natural amino acids, $\tau_\delta = \frac{1}{(N-\delta)} \sum_{i=1}^{N-\delta} (d_{i,i+\delta})^2$, and w is a weight factor.

Following Chou, we chose $w = 0.05$ [45]. The first 20 features are associated with the amino acid composition of the sequence, whereas the next λ_{max} contain sequence order information describing the prevalence of pairwise physicochemical correlations between amino acids separated by $k = 1 \dots \lambda_{max}$ positions that Chou refers to as "tiers" [45]. In Chou's original formulation of the $PseAAC$, $\lambda_{max} \le (\mathrm{N} - 1)$ reflecting the fact that tiers corresponding to residue separations exceeding the length of the peptide are undefined, limiting λ_{max} according to the shortest peptide in the ensemble. In this work $\lambda_{max} = 10$. Chou defines the physicochemical distances appearing in the factors of τ_δ as the weighted average of the Z- scored residue hydrophobicities H, hydrophilicities L,

and side chain masses M assigned by $\left(d_{i,i+\delta}\right)^2 = \frac{1}{3}((H_i - H_{i+\delta})^2 + (L_i - L_{i+\delta})^2 + (M_i - M_{i+\delta})^2)$.

3.4. Data sets

Three different datasets were developed to create and evaluate different models (see **Table 4**). The first dataset was composed by 222 instances, half of this instances correspond to fusion peptide sequences. For each fusion peptide sequence there is experimental evidence showing that they are fusion active (positive samples). The other half of the dataset (negative samples) correspond to a set of 111 sequences randomly generated from the fusion protein sequence, having the same length as the corresponding fusion peptide. The positive samples were labeled as "Fusion Peptide" whereas the negative samples were given the label "Non Fusion Peptide". For each instance, we generated 9507 features as described above. This dataset is a matrix with 9510 columns (9507 features, 1 class, 2 columns of metadata) and 222 rows (111 instances of positive cases and 111 instances of negative cases).

Since the FP and TMD are usually the most hydrophobic regions of fusion proteins, we thought that it would be important to evaluate if our models were able to distinguish the two types of peptides. Therefore, the second dataset was equal to the first dataset only differing in the negative instances which, in this case, are transmembrane domains randomly extracted from the negative cases generated using UniProt.

The third dataset is a mixture of the other two datasets, i.e. it contains 111 positive instances (known fusion peptides) and 111 negative instances (half are randomly generated sequences from fusion proteins and the other half are TMDs).

Table 4 Datasets example of content.

Dataset1 (222x9510)	**Features (x9507)**	**Label (x2)**
Positive Cases (111)	(...)	Fps
Negative Cases (111)	(...)	Nonfps

3.5. Dataset pre-processing

The datasets used to train machine learning models are sometimes not optimal, either containing missing values (NaN), not being standardized, or not being scaled. Undeniably, the dataset should be pre-processed and transformed before developing a machine learning model. To preform pre-processing of the datasets the "scikitlearn" pre-processing features were used. The Standardization of all datasets was done by removing the mean and scaling to unit variance using the "StandardScaler" feature.

3.6. Feature selection

Sometimes, there are features that can lower a model's performance, and the larger the number of features, the longer will take a model to fit the data. Sklearn implements a variety of algorithms for feature selection. However, when these algorithms were tested, the result was not the expected since they were not reducing the dataset enough. Since we have a relatively small dataset, our aim was to have no more than 25 features.

To achieve that, we developed a small function where we used a support vector machine estimator (with a linear kernel), which is then trained using the initial set of features. The estimator then assigns weights to each feature and eliminates the ones with the smallest weights according to the user's threshold, repeating the process recursively, until the desired number of features are achieved.

3.7. Models

Seven different models were evaluated using the training dataset: K-nearest neighbours (KNN) model using sklearn "KNeighborsClassifier" function; Logistic regression (LR) model using sklearn "linear_model.LogisticRegression" function; Stochastic Gradient Descent (SGD) using sklearn "SGDClassifier" function; Random Forest (RF) using sklearn "RandomForestClassifier" ; Naïve Bayes (NB) model using sklearn "GaussianNB" function; and Support Vector Machine (SVM) model using sklearn "svm" function.

3.8. Ensemble Methods

The objective of the ensemble methods is to improve the model´s performance. Therefore, some ensemble methods were implemented. A bagging classifier was used in each model using the sklearn "BaggingClassifier" function. A voting classifier was implemented using the sklearn´s "VotingClassifier" function, which uses a majority voting, meaning that the final prediction for each peptide was the class most often predicted by all the classifiers **(Figure 6).**

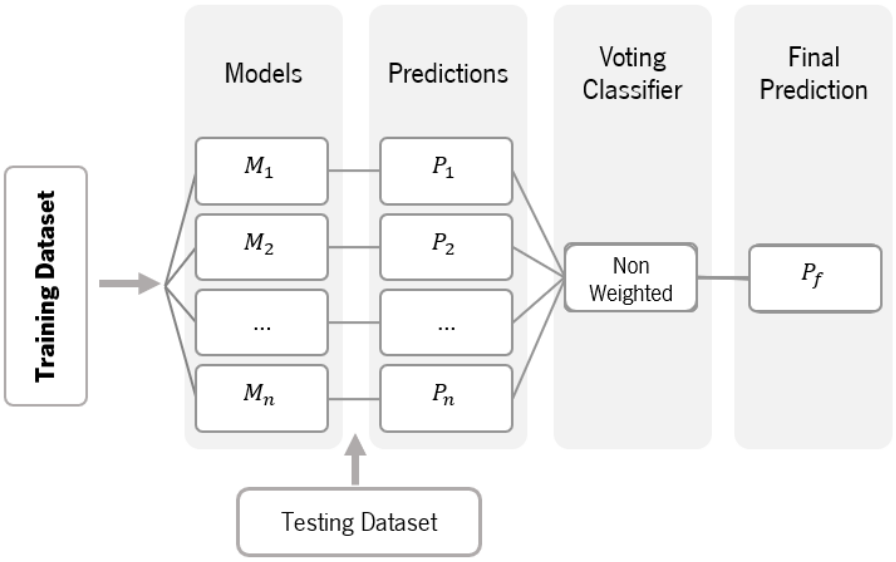


Figure 6 Explanatory schema of the workflow.

3.9. Cross-validation and model performance evaluation

To calculate a model's performance, the training dataset must be different from the dataset used for validation. The sklearn cross validation utilities, has a "cross_val_score" function, to perform a five-fold cross validation test for each model, guaranteeing that the dataset will be divided into 5 equal parts, where 4 parts of the dataset will be used to train the model and the other one for evaluating the model.

This process is used to evaluate the performance of the models. The results of the performance evaluation were calculated by (1) the mean of the Accuracy and Recall scores for each fold of the cross-validation process and (2) leave-on-out. Confusion matrix scores were calculated through the division of the original dataset into training and testing datasets with a proportion of 0.70, meaning that the training dataset will be composed by 70% of the data while the test dataset will be composed by the last 30%.

4

DEVELOPMENT

4.1. Code Developed

The code developed for the purpose of this thesis was written in python 3.6 using Spyder as a Python IDE combined with Jupyter Notebook. The data was collected by downloading the files containing the positive and negative cases as specified in 3.1. After that, the curation was manually done, and the pre- processing was implemented in the notebook using the "StandardScaler" function from the scikit learn package.

4.2. Workflow

The process of creating the datasets to the development of the final models used to make the predictions followed the workflow presented in Figure 19.

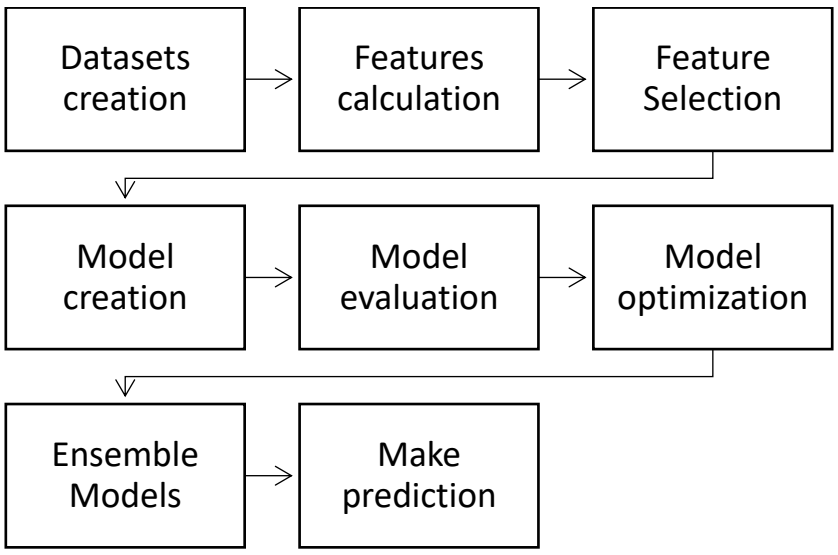


Figure 7 Workflow of the developed algorithm.

4.3. Dataset generation

As explained previously, we tested three different negative datasets in this work, one containing random sequences from fusion proteins (excluding the fusion peptide), another one containing transmembrane domains and a third one combining the first two datasets. In the first step, the data was gathered in two *.csv* files, one of them having the fusion proteins and the respective fusion peptide sequences as well as the virus name in the same line and the second one containing the transmembrane domain (TMD) sequences and the virus they belonged to.

In order to generate de datasets automatically and randomly, the script *Datasets.py* was created. This script generates the three types of datasets according to the user preference and saves them in the working directory with the name given also by the user.

Regarding the first dataset, it is generated by reading a *.csv* file containing the virus name, the fusion protein sequence and the fusion peptide sequence in the first, second and third columns, respectively. For each line in that file (except the header) the function checks if it is a line with only a fusion protein sequence, only a fusion peptide sequence, or both. If there is only a fusion protein sequence (meaning there is no FP annotated for it) the code generates all, same sized, possible fragments (the size is a random number between 15 and 20 – the usual FP's size). If there is only a fusion peptide it simply appends the sequence to the list. If both sequences exist, the FP's sequence is retrieved from the Fusion protein sequence, and then with the remaining part of it, fragments with the same size as the FP are generated. To build the final dataset, the user choses how many negative and positive training examples should be used, and the function retrieves a random dataset containing the chosen number of positive and negative examples, which are outputted to another .csv file called ***dataset1.csv***.

The second dataset is generated by reading two *.csv* files. The first file contains the virus name and the sequence of the fusion peptide in the first and in the second columns, respectively (positive examples). The second file contains the virus name and the sequence of the transmembrane domain in the first and in the second columns, respectively (negative examples). To build the final dataset, the sequences of both files are shuffled and a new .csv file called ***dataset2.csv*** is created containing the number of positive and negative examples chosen by the user.

The third dataset takes in the same .csv file from the first dataset, and the .csv file with the TMD sequences from the second dataset. The mechanism to generate the positive and the negative examples for the first .csv file is the same as the first dataset. All sequences are shuffled and the

user choses how many positive and negative (TMDs and random fragments from the Fusion Protein sequence) examples are written to the output .csv file called ***dataset3.csv***.

4.4. Feature generation

The features used in the machine learning algorithms were generated using the script *features.py*. This script takes one *dataset.csv* file and for each sequence it generates all the features mentioned in the **section 3.3**, using the **Propy** package [38]. The name of the virus, the sequence, and all the 9507 features generated are stored in a *.csv* file in the working directory with the name chosen by the user.

4.5. Data processing

In order to label the dataset, before starting developing the machine learning models, an array with the labels "fp" and "non fp" is appended to the dataset according to its size, in order to identify the negative and the positive cases. Afterwards a preliminary analysis is performed and several metrics are calculated: the dataset dimensions (*dataset.shape*), the type of data included in the dataset *(dataset.dtypes)*, a brief statistic description about every feature (*dataset.describe*) and the data distribution among labels (*dataset.groupby('labels').size*). Subsequently the presence of null values is checked, and all features are standardized, using *sklearn* preprocessing package functions *StandardScaler* to have the properties of a standard normal distribution. The scaling is usually a requirement for many machine learning algorithms but it is also important when comparing measurements that have different units and scales, which is the case.

4.6. Feature selection

Since we had 9507 features, before applying ML it was necessary to reduce the number of features used and for that purpose some built-in functions from *sklearn* package were tested. However, we were not satisfied with results, so we designed a function, similar to one used in a previous work focusing on antimicrobial peptides [7]. This function receives as arguments a dataset and a threshold. It uses a support vector classifier with a linear kernel to fit the dataset and then removes the features whose coefficients absolute values are below the threshold. It is recommended to start

with small threshold values (0,0001 – 0,01) and gradually increase its value. (Ex.: 0. 0001, 0. 0002, 0. 0003, ... ,0.001, ..., 0.002, ... , 0.01, ..., n). Thus, we used a recursive approach, in which this function was called several times, with gradually larger thresholds until only a small number of features remained.

4.7. Model creation, evaluation and optimization

Regarding the ML models, in a first moment several algorithms were evaluated on the test set, namely KNN, Logistic Regression, SGD Classifier, Decision Trees, Naive Bayes, SVCs and Neural Networks. To evaluate reliably the performance of the models, the score of each prediction for each model was calculated using a 5-fold cross validation (accuracy as score function) and leave one out.

The parameters used in most of the models described above were the default ones. However, for some algorithms (neural networks for instance) different parameter sets had to be tested in order to optimize the models.

After the evaluation and optimization of the models, the goal is to use one or several of these models to make predictions about new test cases. We decided to build our prediction function from an ensemble of models (with its parameters duly optimized). To this end, we used the Bagging method where 50 would be the number of estimators, 70% would be the maximum number of used samples and the maximum number of features to train the models. With the optimized models given by the Bagging method, a voting classifier (hard voting) was implemented using KNN, SVC, Logistic Regression, Neural Networks and Naïve Bayes bagged models.

5

Results and Discussion

Overall, in the built database, 107 PDB entries and 640 UniProt entries were collected: 598 of them reviewed and 42 unreviewed. From a total of 255 virus species, which are known enveloped viruses that target humans, the fusion protein's sequences were found for 207 of them, amounting to a total of 617 sequences (for some species several sequences corresponding to different subtypes were retrieved). Fusion peptide sequences were found for 207 virus species, amounting to a total of 468 fusion peptide sequences, of which 341 have annotated methods (i.e. the method of assessment is known) and 186 have activation triggers. These statistics include duplicate sequences for different virus strains (see **attachment A**).

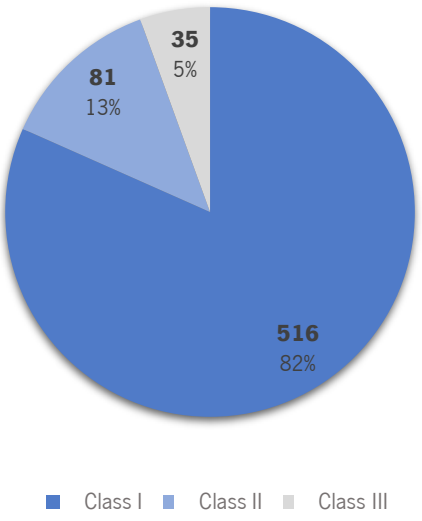


Figure 8 Distribution of FP's sequences per class.

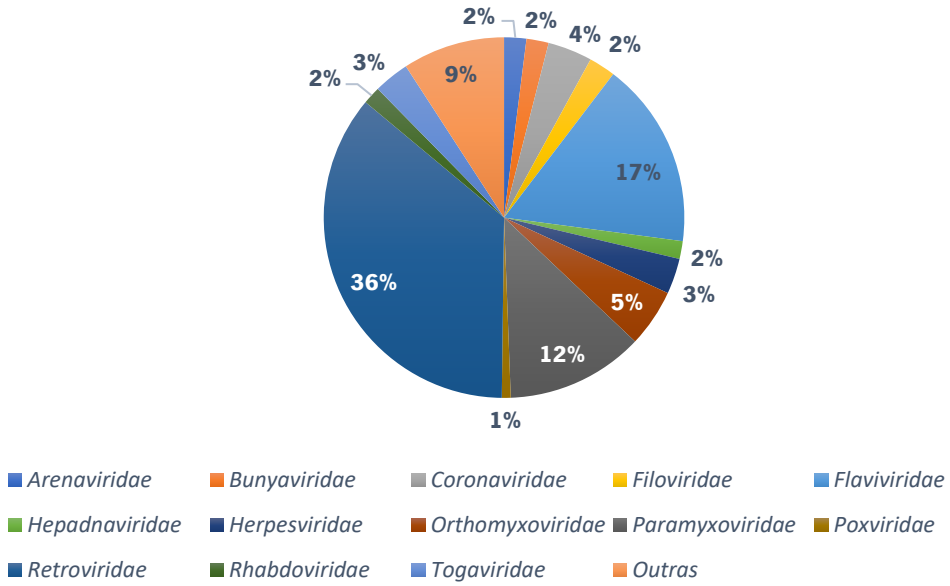


Figure 9 Distribution of FP's sequences per family. Most of the annotated or studied viruses belong to Retrovirus, Flavivirus and paramyxovirus' families, note that for some families were found more than one fusion peptide per specie.

Fusion peptides sequences were found for 3 out of 5 different species of *Arenaviridae's* family, 5 out of 15 species of *Bunyaviridae's* family, 12 out of 13 species of *Coronaviridae's* family, 12 out of 15 species of *Flaviviridae's* family, 8 out of 10 species of *Herpesviridae's* family, 33 out of 236 species of *Orthomyxoviridae's* family, 24 out of 38 species of *Paramyxoviridae's* family, 100 out of 101 different species of *Retroviridae's* family, 2 out of 9 species of *Rhabdoviridae's* family and 14 out of 18 species of *Togaviridae's* family. Also, FPs sequences were found for all three species of *Filoviridae's* family

Most of the sequences found belong to class I fusion proteins (82%), followed by class II (15%) and only 5% belong to class III (**Figure 8**). 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) (**Figure 9**).

5.1. Protein Alignments

As mentioned in *Methods*, multiple alignments of all fusion protein and fusion peptide sequences were made. At the phylogenetic tree, it was possible to generally observe that sequences were being grouped by classes, and inside classes grouped by families. However, some sequences were dispersed throughout the tree, such as *Togaviridae* sequences (**Figure 10**). For example, the

Retroviridae family, shows up largely grouped at the bottom of the tree and some sequences show up at the top of it and at the middle.

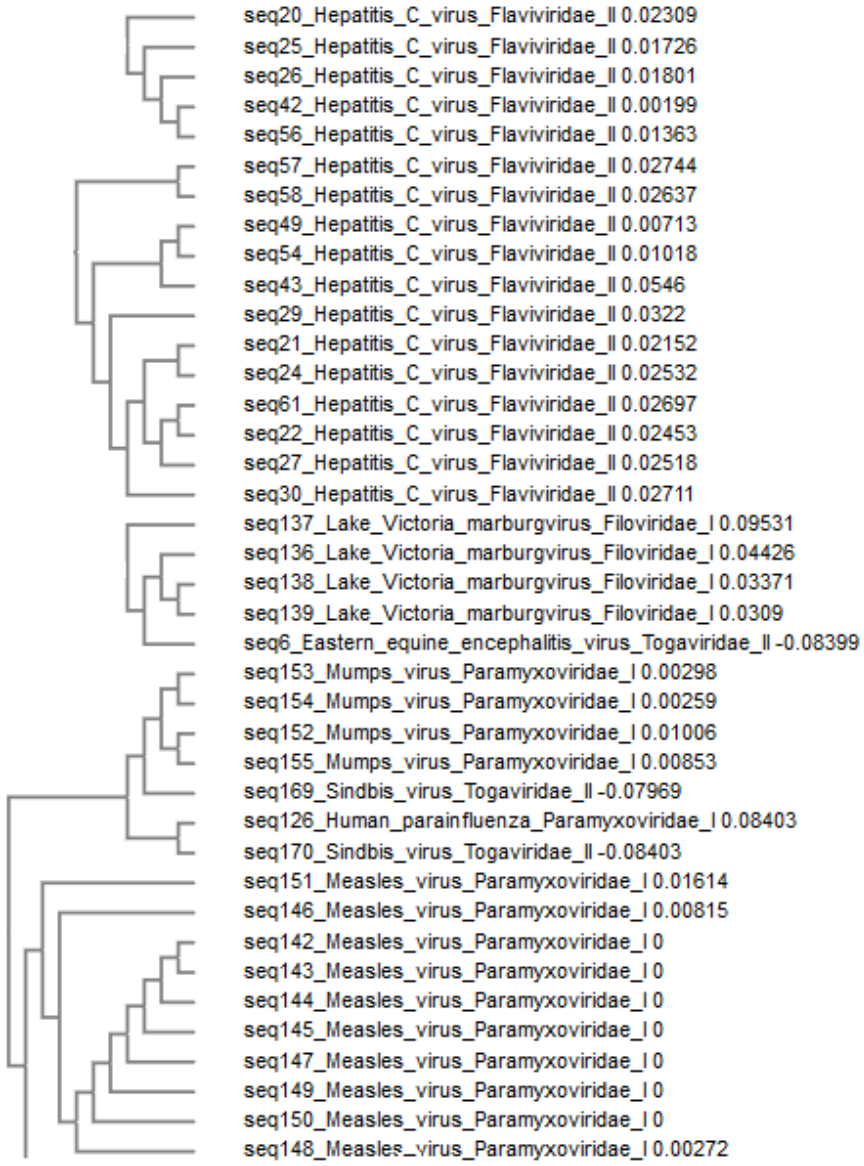


Figure 10 Fragment of the Phylogenetic tree where a group of Class II fusion protein sequences of the same family can be observed, and another group of Class I fusion protein sequences grouped by families. Here is also apparent that some species from Togaviridae's family are not correctly clustered.

Although the species of all families were known, the correspondent class was not identified a priori for all of them. Analyzing the phylogenetic tree, one could predict the classes for those species. As an example, Hendra virus could possibly belong to Class I (**Figure 11**), and Mayaro Virus, Semliki Forest Virus and Sagiyama virus, from *Togaviridae* family, could fit into Class II (**Figure 12**), since they are clustered with other members of these classes. [47, 48]

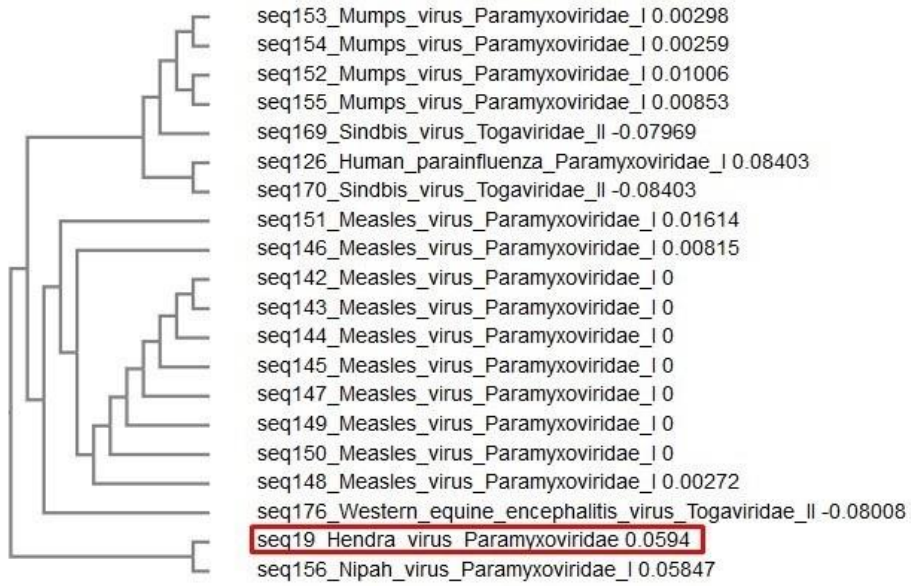


Figure 11 . Phylogenetic tree's fragment. Hendra Virus belong to Paramyxovirus's family, it's closest neighbor is a Class I Paramyxovirus, and their clustered to a group of Class I paramyxoviruses, hence, Hendra virus's protein probably belong to Class I.

Figure 12 . Phylogenetic tree's fragment. Mayaro Virus, Semliki Forest Virus and Sagiyama virus belong to Togavirus' family, their closest neighbors are a group of Class II Togarirus (Chikungunya and O'nyong-nyong viruses), clustered to another group of Class II

Some regions of the fusion protein (e.g. the fusion peptide) are usually conserved within a given viral family [4]. Hence, alignments of the fusion proteins for each family were individually made. Since there was no (or few) information available for some families, the alignment between families was only performed for the ones having a significant number of sequences (*Filoviridae, Flaviviridae, Paramyxoviridae, Retroviridae and Togaviridae*).

Filoviridae family alignment resulted in two conserved regions (at the beginning and at the end of the sequence). The second region mentioned corresponds to the fusion peptide region found for

this family [10] at UniProt as seen in **Figure 13** below, which indicates that intrafamily alignments can be used to provide hints into the location of the fusion peptide.

Ebolavirus	Filoviridae	Q05320	Envelope glycoprotein	I	GP2	1–676	MGVTGILQLPRDRF	524–539	GAAIGLAWIPYFGPAA	Curated
Ebolavirus	Filoviridae	Q66799	Envelope glycoprotein	I	GP2	1–676	MGSGYQLLQLPREF	525–540	GAAVGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	Q66814	Envelope glycoprotein	I	GP2	1–676	MEGLSLLQLPRDKF	524–539	HNAAGIAWIPYFGPGA	Similarity
Ebolavirus	Filoviridae	Q91DD8	Envelope glycoprotein	I	GP2	503–677		525–540	GAAAGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	Q7T9D9	Envelope glycoprotein	I	GP2	1–676	MGGLSLLQLPRDK	524–539	HNAAGIAWIPYFGPGA	Similarity
Ebolavirus	Filoviridae	P87671	Envelope glycoprotein	I	GP2	1–676	MGVTGILQLPRDR	524–539	GAAIGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	O11457	Envelope glycoprotein	I	GP2	1–676	MGVTGILQLPRDRF	524–539	GAAIGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	Q89853	Envelope glycoprotein	I	GP2	503–677		525–540	GAAVGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	Q66798	Envelope glycoprotein	I	GP2	1–676	MEGLSLLQLPRDKF	524–539	HNAAGIAWIPYFGPGA	Similarity
Ebolavirus	Filoviridae	P87666	Envelope glycoprotein	I	GP2	502–676		524–539	GAAIGLAWIPYFGPAA	Similarity
Ebolavirus	Filoviridae	Q66810	Envelope glycoprotein	I	GP2	502–676		524–539	GAAIGLAWIPYFGPAA	Similarity
Lake Victoria marburgvirus	Filoviridae	Q6UY66	Envelope glycoprotein	I	GP2 - by similarity	1–681	MRTTCLFISLILIQGI	529–549	GLSWIPFFGPGIEGLYTAGLI	Similarity
Lake Victoria marburgvirus	Filoviridae	Q1PDC7	Envelope glycoprotein	I	GP2 - by similarity	1–681	MKTIYFLISLILIQSIK	529–549	GLSWIPFFGPGIEGLYTAGLI	Similarity
Lake Victoria marburgvirus	Filoviridae	P35254	Envelope glycoprotein	I	GP2 - by similarity	1–681	MKTTCLFISLILIQGI	529–549	GLSWIPFFGPGIEGLYTAGLI	Similarity
Lake Victoria marburgvirus	Filoviridae	P35253	Envelope glycoprotein	I	GP2 - by similarity	1–681	MKTTCFLISLILIQGT	529–549	GLSWIPFFGPGIEGLYTAVLI	Similarity

Figure 13 Filoviridae's family information found at UniProt agree with the multiple alignment's conserved region.

Paramyxoviridae family alignment also came back with a good result, showing a conserved region in the beginning of the alignment, agreeing with the annotated fusion peptide region for this family (see **Figure 14**) [49, 50].

Human parainfluenza	Paramyxoviridae	P25467	Fusion glycoprotein F0	I	F1	1–551	MHHLHPMIVCIFVI	107–131	FAGVVVGLAALGVATAAQITAA	Similarity
Human parainfluenza	Paramyxoviridae	P06828	Fusion glycoprotein F0	I	F1	1–539	MPTSILLIITTMIMA	110–134	FFGGVIGTIALGVATSAQITAAV	Similarity
Human parainfluenza	Paramyxoviridae	P27286	Fusion glycoprotein F0	I	F1	1–551	MHHLHPMIVCIFVI	107–131	FAGVVVGLAALGVATAAQITAA	Similarity
Human parainfluenza	Paramyxoviridae	P26629	Fusion glycoprotein F0	I	F1	1–551	MHHLHPMIVCIFVI	107–131	FAGVVVGLAALGVATAAQITAA	Similarity
Human parainfluenza	Paramyxoviridae	P12605	Fusion glycoprotein F0	I	F1	1–555	MQKSEILFLIYSSLLL	113–137	FFGAVIGTIALGVATAAQITAGIA	Similarity
Human respiratory syncytial virus	Paramyxoviridae		Fusion glycoprotein F0	I	F1					
Human respiratory syncytial virus	Paramyxoviridae		Fusion glycoprotein F0	I	F1					
Human respiratory syncytial virus	Paramyxoviridae	P12568	Fusion glycoprotein F0	I	F1	1–574	MELPILKANAITTILA	137–157	FLGFLLGVGSAIASGTAVSKV	Similarity
Human respiratory syncytial virus	Paramyxoviridae	O36624	Fusion glycoprotein F0	I	F1	1–574	MELLIHRLSAIFLTLA	137–157	FLGFLLGVGSAIASGIAVSKV	Similarity
Human respiratory syncytial virus	Paramyxoviridae	P11209	Fusion glycoprotein F0	I	F1	1–574	MELPILKTNAITAILA	137–157	FLGFLLGVGSAIASGIAVSKV	Similarity
Human respiratory syncytial virus	Paramyxoviridae	P13843	Fusion glycoprotein F0	I	F1	1–574	MELLIHRSSAIFLTLA	137–157	FLGFLLGVGSAIASGIAVSKV	Similarity
Human respiratory syncytial virus	Paramyxoviridae	P03420	Fusion glycoprotein F0	I	F1	1–574	MELLILKANAITTILT	137–157	FLGFLLGVGSAIASGVAVSKV	Similarity
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1					
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1				FAGVVLAGAALGVATAAQITAGIA**LHQSNLN**	
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1				FAGVVLAGAALGVATAAQITAGIAL	
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1				FAGVVLAGAALGVATAAQITAGIAHQSMLNSQ	
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1					
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1					
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1					
Measles virus	Paramyxoviridae		Fusion glycoprotein F0	I	Fusion glycoprotein F1					
Measles virus	Paramyxoviridae	P69355	Fusion glycoprotein F0	I	F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	P69357	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	P69354	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	Q786F3	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	P26031	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–529	MSIMGLKVNVSAIF	116–140	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	P69358	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity
Measles virus	Paramyxoviridae	P35973	Fusion glycoprotein F0	I	Fusion glycoprotein F1	1–550	MGLKVNVSAIFMAV	113–137	FAGVVLAGAALGVATAAQITAG	Similarity

Figure 14 Paramyxoviridae's family information found at UniProt agree with the multiple alignment's conserved region.

Retroviruses family alignment showed a conserved region in the middle of the fusion protein, agreeing with the collected data as well [4, 51, 52]. However, *Flaviviridae* viruses' alignment did not present significant conserved regions and information showed that protein sequence length was inconstant, probably due to incorrect annotations. In order to improve this alignment, in the future it will be necessary to verify all the protein annotations of this family.

5.2. Fusion Peptide Alignments

The global alignment showed the same results as for the fusion proteins. At the phylogenetic tree, generally, FPs sequences were grouped by classes, and inside classes grouped by families. However, *Retroviridae* family appeared to be dispersed throughout the tree as happened with fusion protein sequences.

Hepatitis B virus, belongs to *Hepadnaviridae's* family, and only one FP sequence was found. At the global phylogenetic tree, it was grouped with Hepatitis C *Flavivirus* (Class II fusion protein) suggesting that *Hepadnavirus* family fusion proteins could belong to a Class II (**Figure 15**).

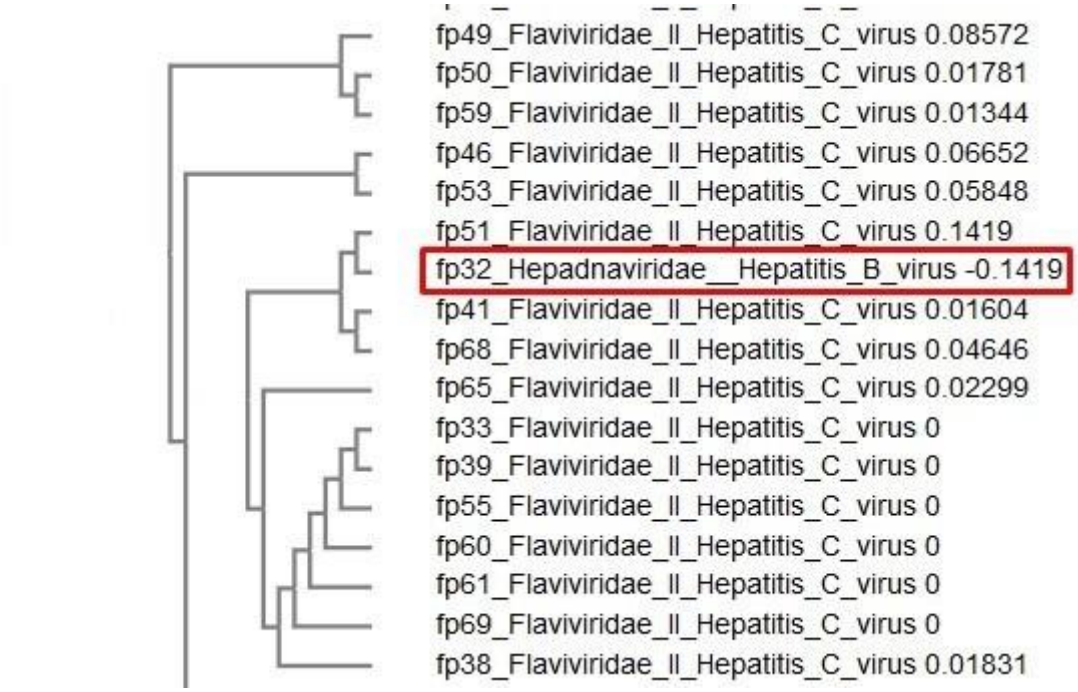


Figure 15 Fragment of FP's Phylogenetic tree that includes Hepatitis B FP's sequence, suggesting that it's fusion protein could belong to class II.

Hepatitis G virus belongs to the *Flaviviridae* family, which has class II fusion proteins [1, 3], but at the phylogenetic tree, this species is grouped with *Rhabdovirus* family and herpesvirus family. The last two have class III fusion proteins [1, 3], suggesting a misclassification (see **Figure 16**). However, it is also possible that in spite of clustering together in the phylogenetic analysis, these proteins belong to different classes. It is worth noting that the classification into classes is not based on phylogeny but rather on empirical structural properties.

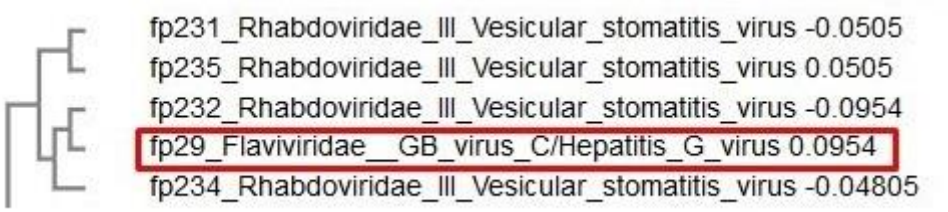


Figure 16 Fragment of FP's Phylogenetic tree that includes Hepatitis G FP's sequence, suggesting a misclassification.

Since the FP global alignment suggested groups of families, all sequences were divided by families, and aligned individually to create sequence logos of the fusion peptide for each family. Results were more significant for 6 families, the ones with more FP sequences (*Filoviridae*, *Retroviridae*, *Flaviviridae*, *Paramyxoviridae*, *Togaviridae* and *Orthomyxoviridae*), resulting in more significant conserved domains (**Figure 17**).

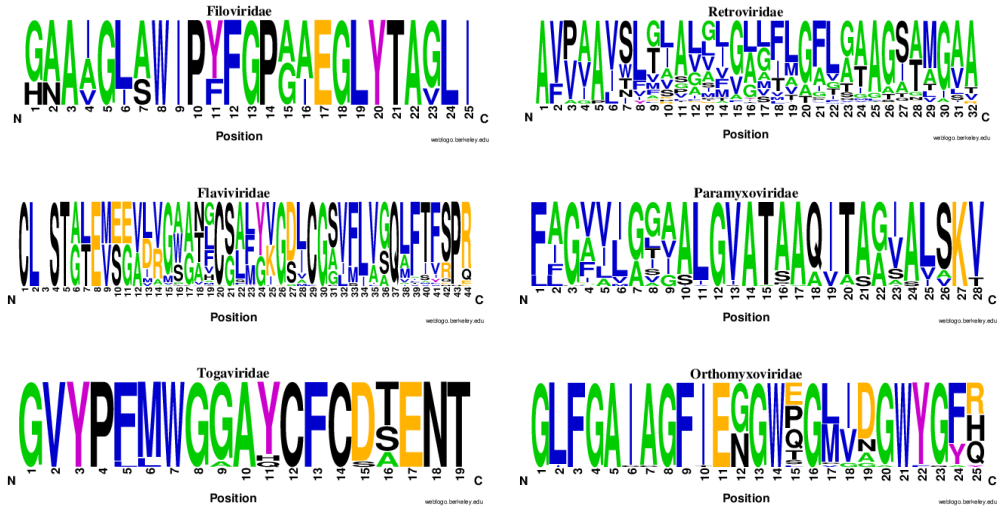


Figure 17 FP's sequence logos created with Weblogo version 3.6.0 and probability as unit. Abscissa axis indicate residues' positions. Colors for this WebLogos do not correspond to default ones.Gree: Alanine (A) and Glycine (G), because they are the most present residues in FPs sequences; Black: Histidine (H), Asparagine (N), Serine (S), Proline (P), Threonine (T), Cysteine (C) and Glutamine (G) for being considered polar residues; Blue: Isoleucine (I), Valine (V), Leucine (L), Tryptophan (W), Phenylalanine (F) and Methionine (M) for being considered Hydrophobic residues; Yellow: Glutamic Acid (E), Arginine (R), Lysine (K) and Aspartic Acid (D) for being considered charged residues; Pink: Tyrosine (Y) for being hydrophobically neutral.

Dhori virus belongs to *Orthomyxoviridae* family, however no fusion protein nor fusion peptide sequence was found, hence from the generated sequence logo for this family, it could be suggested that Dhori's fusion peptide conserved domain could include or be included in the GLFGAIAGFEGGWEGLIDGWYGFR motif.

Rubella virus belongs to *Togaviridae* family, however no fusion peptide or fusion protein sequences were found for this virus. Hence, out of the logo created for *Togaviruses*, one can suggest that Rubellas's fusion peptide conserved domain could include the VYPFMWGGAYCFCDTENT motif.

Once again, these results indicate that intrafamily fusion peptide alignments can be useful for locating fusion peptides and predicting their sequences for the viruses in which they are unknown.

5.3. Machine Learning Scores

Three different sets of data were used (Dataset1, Dataset2 and Dataset3) to evaluate the developed models. The first dataset was used to evaluate the models' capability to distinguish fusion peptides from other random sequences belonging to the fusion protein. The second dataset was used to evaluate if the models were able to distinguish fusion peptides from transmembrane domains. 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.

The number features was always 25 or below and these features were automatically selected using a support vector classifier, as explained in the Chapter 4.4 - Feature generation.

5.3.1. Generated features

Overall, the most common group of features among the three datasets were single amino acid composition, dipeptide composition, the distribution descriptors – hydrophobicity, polarity, secondary structure, charge and solvent accessibility, and pseudo amino acid composition (**Table 5**).

According to the literature, fusion peptides are mostly hydrophobic or amphipathic, containing, in some cases, charged groups, and have a reduced to moderate polarity. Hence, the features that were selected by the SVM machine learning classifier agree with the FP properties described in the literature.

Table 5 Features generate by the SVC, for the three used datasets. Group of features are aggregated by color: light blue is the single amino acid composition, green the dipeptide composition, orange is pseudo amino acid composition, and dark blue is the CTD group of features.

Features	Dataset 1	Dataset 2	Dataset 3
A	✓	✓	
C		✓	✓
E	✓		
F	✓		✓
G	✓		
K	✓		
L		✓	
T			✓
AV	✓		
FC			✓
FF			✓
FG			✓
FH			✓
FI			✓
GF			✓
GL			✓
HL			✓
IG			✓
II			✓
IT		✓	
LA	✓		✓
LG		✓	
LL			✓
PL	✓		
RD	✓		
TM			✓
TS			✓
VD			✓

Features	Dataset 1	Dataset 2	Dataset 3
APAAC1	✓		
APAAC3	✓		✓
APAAC10	✓		
APAAC12	✓		
APAAC14			✓
APAAC15	✓		
APAAC17			✓
APAAC20	✓		
PAAC1	✓	✓	
PAAC5			✓
PAAC8	✓		
PAAC14	✓		
PAAC32		✓	
PAAC38		✓	
CHARGED2075		✓	
HYDROPHOBICITYD1001		✓	
HYDROPHOBICITYD2025			✓
HYDROPHOBICITYD3025	✓		
HYDROPHOBICITYD3050		✓	
POLARITYD1001	✓		
POLARITYD1050		✓	
POLARITYD2001	✓		
POLARITYD2050	✓		
POLARITYD2075		✓	
SECONDARYSTRD1025	✓		
SECONDARYSTRD1100	✓	✓	
SECONDARYSTRD2050		✓	
SECONDARYSTRD2075		✓	
SECONDARYSTRD3050		✓	
SOLVENTACCESSIBILITYD2001		✓	

Besides single amino acid composition and dipeptides, which are self-explanatory, PAAC are the initials for pseudo amino acid composition, and APAAC the initials for amphiphilic pseudo amino acid composition, the number after these initials represents the amino acid index: 1 corresponds to A (Alanine), 2 corresponds to R (Arginine) and so on (see **Table 6**).

Regarding the remaining features (Hydrophobicity, Charge, Polarity, Secondary Structure and Solvent Accessibility), they are all part of the CTD features group, specifically, the Distribution group (hence the letter D on CTD and at the end of the feature, e.g.: HYDROPHOBICITY**D**2025). This group of features represents the distribution of certain amino acids (depending on the property) throughout the analyzed sequence. On the example given above, the number **2** after the letter **D** represents neutral amino acids (G, A, S, T, P, H and Y), and the three numbers after that represent in which quarter of the sequence this distribution is confined, see **attachment B.**

It is interesting to note that the features that were generated by the feature selection protocol varied according to the dataset used. This is can be explained by the fact that the properties that distinguish FPs from other random sequences belonging to the fusion protein are different form the properties that distinguish them from transmembrane domains.

Table 6 Amino Acid (AA) indexes mapping, and correspondent chemical properties.

AA Index	AA	Description	Hydrophobic	Charged	Polar	Amphipatic
1	A	Alanine	✓			
2	R	Arginine		✓		
3	N	Asparagine			✓	
4	D	Aspartic acid		✓		
5	C	Cysteine			✓	
6	Q	Glutamine			✓	
7	E	Glutamic acid		✓		
8	G	Glycine	✓			
9	H	Histidine			✓	
10	I	Isoleucine	✓			
11	L	Leucine	✓			
12	K	Lysine		✓		
13	M	Methionine				✓
14	F	Phenylalanine	✓			

AA Index	AA	Description	Hydrophobic	Charged	Polar	Amphipatic
15	P	Proline	✓			
16	S	Serine			✓	
17	T	Threonine			✓	
18	W	Tryptophan				✓
19	Y	Tyrosine				✓
20	V	Valine	✓			

5.4. ML models: Default parameters

To evaluate and improve the models performance, several different tests were performed, using different feature filters and pre-processing methods on Dataset1. First, the pre-processing of the dataset, only the "StandardScaler" function was used and no set of parameters was given to the algorithms (i.e. the default parameters were used). The results of the five-fold cross validation and leave-one-out tests are shown in **Table 7**, in which the **SVC** model was the lowest performing model (bellow 0.80) in both Accuracy and F1 scores while all the others were above. Overall, all models had a good performance, however the Logistic Regression model outperformed all the other ones. This algorithm is one of the simplest ML algorithm and yet provides great efficiency, plus its variance is low.

Table 7 Mean of accuracy scores after a 5-fold cross validation and leave-one-out processes using Dataset 1. Coloured in red are the models with lower score and coloured in green are the models with higher scores.

Models	F1 Scores	Accuracy (5-fold CV)	Accuracy (Loo)
KNN	0.84	0.87	0.89
Logistic Regression	0.93	0.92	0.92
SGDC	0.93	0.88	0.89
Decision Trees	0.90	0.89	0.88
Naïve Bayes	0.93	0.85	0.82
SVC	0.80	0.78	0.81
NN	0.91	0.94	0.93

Regarding Dataset 2, cross-validation and Leave-one-out tests are shown in **Table 8**, in which the **SGDC** model was the lowest performing model in both Accuracy (loo) and F1 scores. Overall, all models had a good performance, however the **Neural Networks** model outperformed all the

other ones. Since this dataset is only composed by TMDs and FPs, which have similar properties, making this problem more complex, we had to use more sophisticated algorithms to find the patterns to distinguish them.

Table 8 Mean of accuracy scores after a 5-fold cross validation and leave-one-out processes using Dataset 2. Coloured in red are the models with lower score and coloured in green are the models with higher scores.

Models	F1 Scores	Accuracy (5-fold CV)	Accuracy (Loo)
KNN	0.84	0.80	0.83
Logistic Regression	0.87	0.88	0.91
SGDC	0.41	0.74	0.75
Decision Trees	0.90	0.85	0.91
Naïve Bayes	0.82	0.84	0.84
SVC	0.72	0.64	0.78
NN	0.91	0.91	0.91

Dataset 3, cross-validation and leave-one-out tests are shown in **Table 9**, in which the **SVC** model was the lowest performing model in all measures. Overall, all models had a poor performance. However, the **Logistic Regression** model outperformed all the other ones. This dataset is composed by TMDs, FPs and random fusion protein's subsequences. Since FPs and TMDs are both hydrophobic sequences and similar at the sequence level, it is possible that the algorithms are joining TDMs and FPs in the same class, as being all Fusion Peptides. A work around for this dataset could be a multiclass classification approach, where the algorithms could distinguish FPs from TMDs from other fusion protein sequences, having not two, but three classes instead.

Table 9 Mean of accuracy scores after a 5-fold cross validation and leave-one-out processes using Dataset 3. Coloured in red are the models with lower score and coloured in green are the models with higher scores.

Models	F1 Scores	Accuracy (5-fold CV)	Accuracy (Loo)
KNN	0.65	0.61	0.68
Logistic Regression	0.76	0.72	0.75
SGDC	0.50	0.63	0.64
Decision Trees	0.65	0.65	0.69
Naïve Bayes	0.83	0.68	0.69
SVC	0.34	0.46	0.61
NN	0.64	0.67	0.73

5.5. ML models: Optimized parameters

The results described above were obtained with the parameters defined by default for each ML model. However, in some cases, for convergence purposes these parameters had to be manually changed, as the case of the neural networks algorithm, in which the solver and the alpha had to be added to the parameters set.

Nevertheless, there are methodologies to optimize these parameters, which were also tested (the results are shown in the tables below). The final value is the bagged result of different optimized models. For the bagging method, the defined number of estimators was 20, and the maximum number of samples to train each estimator was 70% of the original set of data.

Overall, using the optimized parameters, all models had improvements on their performances. As shown in **Table 10**, the bagged models which performed best were **SVCs** for datasets one and two, and Logistic Regression for dataset three. All models had their lowest performance when using KNN models.

Table 10 Mean of accuracy scores after a 5-fold cross validation process. Coloured in red are the models with lower score and coloured in green are the models with higher scores.

Models	Dataset 1	Dataset 2	Dataset 3
KNN	0.88	0.81	0.63
Logistic Regression	0.92	0.88	0.72
Naïve Bayes	0.90	0.86	0.69
SVC	0.94	0.90	0.66
NN	0.93	0.88	0.67

Again, dataset 1 showed the best results when compared to other datasets and since SVCs were the models showing best performance, below (**Table 11**) are the optimized parameters for this model.

Table 11 Optimized parameters for SVM models using Dataset 1. Description of parameters retrieved from sklearn's package information.

Parameter	Description	Value
C	Penalty parameter C of the error term.	1
cache_size	Specify the size of the kernel cache (in MB).	200
class_weight	Set the parameter C of class i to class_weight[i]*C.	None

Parameter	Description	Value
coef0	Independent term in kernel function.	0.0
decision_function_shape	Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). However, one-vs-one ('ovo') is always used as multi-class strategy.	ovr
degree	Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.	3
gamma	Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. Current default is 'auto' which uses 1 / n_features, if gamma='scale' is passed then it uses 1 / (n_features * X.std()) as value of gamma.	0.001
kernel	Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).	rbf
max_iter	Hard limit on iterations within solver, or -1 for no limit.	-1
probability	Whether to enable probability estimates.	False
random_state	The seed of the pseudo random number generator used when shuffling the data for probability estimates. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.	None
shrinking	Whether to use the shrinking heuristic.	True
tol	Tolerance for stopping criterion.	0.001
verbose	Enable verbose output.	False

On dataset 1 the C parameter was low which means gamma parameter must be small (=0.001), meaning that the used data have higher bias and low variance. These parameters make sense since the models were trained using a very small set of samples (222 instances). Also, since the number of features is small compared to an intermediate number of samples, rbf (gaussian) is a plausible choice of kernel.

Regarding dataset 2 the parameters gamma (=auto) and kernel (=linear) differed from the results on dataset 1, adjusting the model to the set of training data. These parameters make sense, since this dataset is composed exclusively with FPs and TMDs, hence the choice of a linear kernel

Also, on dataset 3 the parameters C (=100), gamma (=auto) and kernel (=linear) were different. The Penalty parameter C of the error term is higher on this dataset when compared to the other two, because this dataset has a larger variance on the samples (50% FPs, 25% TMDs and 25% other random fusion protein subsequences).

After obtaining the best results for each model, an ensemble method was implemented on them. The purpose of implementing an ensemble of the models was to combine their prediction capabilities into a better suited model. Three hard voting classifiers and three weighted voting classifiers were tested, and the results are shown in **Table 12**.

Regarding the hard-voting classifier, KNN, SVC, NN, Logistic Regression and NB models were used, while the weighted-voting classifier used the same models with weights equal to 3, 4, 2, 2 and 3 respectively.

Table 12 Accuracy and F1 score for both voting classifiers, for the three datasets. Coloured in green are the models with higher scores.

Metrics	**Voting**	**Dataset 1**	**Dataset 2**	**Dataset 3**
Accuracy	**Hard**	0.96	0.90	0.73
	Weighted	0.94	0.99	0.81
F1 Score	**Hard**	0.95	0.87	0.74
	Weighted	0.93	0.98	0.83

Overall, the results obtained using this ensemble method were quite promising, especially for dataset 1, with ***Accuracy*** and *F1* scores of around 0,95. The results on dataset 2 are also very good, although not as good as those obtained for dataset 1 (scores around 0,9). This is quite impressive since FPs and TMDs have similar features such as high hydrophobicity, which means Alanine (A), Isoleucine (I), Leucine (L), Methionine (M), Phenylalanine (F), Valine (V), Proline (P),

Glycine (G) are predominant in both FPs and TMDs sequences. However, the relative frequencies of each aa residues may vary between FPs and TMDs, e.g. FPs contain a larger number of G and A residues. The fact that our models were able to distinguish the two classes means that our protocol was able to select the features that differ between FPs and TMDs, which *a priori* did not appear to be trivial.

Regarding dataset 3, as discussed above, it is possible that the algorithms are misclassifying TMDs as being FPs, when compared against other sequences from a fusion protein, hence this dataset performance is always the worst. Nevertheless, tuning the parameters and using an ensemble method improved the results.

Overall, using dataset 1, algorithms perform better when compared to dataset 2 and especially dataset 3. It is possible that dataset 1 also misclassifies TMDs as fusion peptides and more tests are needed to clarify this. In any case, this is the dataset that better represents the problem that we want to treat, i.e., given a fusion protein sequence predict the location of the fusion peptide. For this reason, we decided to use the models trained using dataset 1 to create the prediction function discussed below

5.6. Case Studies

5.6.1. Control Group - Dengue

To test if the trained models were predicting the correct fusion peptide sequence, a known fusion protein was given as an input – the fusion protein from Dengue virus.

The models correctly predicted the fusion peptide sequence. However, the model also predicted as fusion peptide, some sequences that were immediately before the beginning of the FP and some sequences immediately after the end of the FP, which is reasonable, since these sequences contain parts of the actual FP. Also, Dengue's TMD is located at the end of the fusion protein sequence, and the algorithm predicted it to be a FP, as well as some sequences that were immediately before the beginning of the TMD (**Figure 18** and **Figure 19**). As discussed above, this was expected, since we used the models trained with dataset 1 (which contained few TMD sequences) to make these predictions. These results indicate that we need to further refine our datasets and use additional information to distinguish FPs from TMDs. FPs are usually located further upstream in the sequence than TMDs. Additionally, FPs are very conserved among viruses

of the same family and this information can help to distinguish them from TMDs. Additionally, our algorithm can be refined to output a probability value, instead of a binary output (FP or non FP). We also note, that in ML there is always a compromise between precision and recall. In this case, we prioritize recall, since our aim is to predict putative FPs that can be tested in the lab.

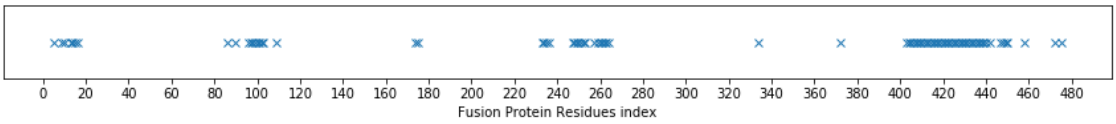


Figure 18 FP indexes predicted by the ensemble of ML models, for Dengue's Fusion Protein.

Given Dengue's fusion protein **Figure 19** and the indexes of the predicted sequences **Figure 18**, it is possible to observe predominant regions of prediction.

Figure 19 Dengue's fusion protein. Regions predicted as containing fusion peptides highlighted in yellow. Actual fusion peptide highlighted in red.

MRCIGISNRDFVEGVSGGSWVDIVLEHGSCVTTMAKNKPTLDFELIETEAKQPATLRKYCIEAKLTN
TTTDSRCPTQGEPSLNEEQDKRFVCKHSMVDRGWGNGCGLFGKGGIVTCAMFTCKKNMKGKV
VQPENLEYTIVITPHSGEEHAVGNDTGKHGKEIKITPQSSITEAELTGYGTVTMECSPRTGLDFNEM
VLLQMENKAWLVHRQWFLDLPLPWLPGADTQGSNWIQKETLVTFKNPHAKKQDVVVLGSQEG
AMHTALTGATEIQMSSGNLLFTGHLKCRLRMDKLQLKGMSYSMCTGKFKVVKEIAETQHGTIVIR
VQYEGDGSPCKIPFEIMDLEKRHVLGRLITVNPIVTEKDSPVNIEAEPPFGDSYIIIGVEPGQLKLNW
FKKGSSIGQMIETTMRGAKRMAILGDTAWDFGSLGGVFTSIGKALHQVFGAIYGAAFSGVSWIM
KILIGVIITIGMNSRSTSLSVSLVLVGVVTLYLGVMVQA

5.6.2. Rubella virus

Rubella virus is single stranded RNA virus of the *Togaviridae* family (genus *Rubivirus*), which also includes Chikungunya virus. Rubella virus is a spherical, 40- to 80-nm, positive-sense, single-stranded RNA virus with spike-like, hemagglutinin-containing surface proteins [53].

In **Figure 20,** we can observe a predicted FP region in the beginning of the fusion protein up until the 140th residue, and another one in the end. This virus belongs to the *Togavridae* family, and this family's FP region is in the beginning of the fusion protein. Hence, it is very likely that one of the predicted sequences from the beginning of the fusion protein corresponds to the real FP.

The region of predicted sequences at the end of the fusion protein could correspond to the TMD of the fusion protein.

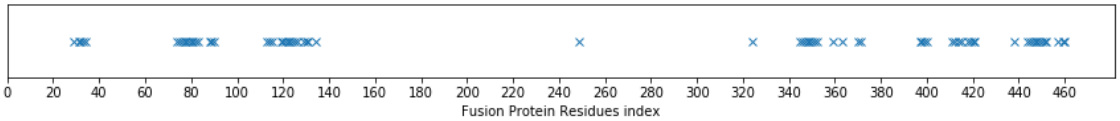


Figure 20 FP indexes predicted by the ensemble of ML models, for Rubella Virus Fusion Protein.

5.6.3. Classical swine fever virus (Hog cholera virus)

This virus was previously called hog cholera virus. It belongs to the *Pestivirus* genus in the *Flaviviridaes* family. CSFV is closely related to the ruminant *pestiviruses* that cause bovine viral diarrhea and border disease.

In **Figure 21**, we can observe a predicted FP in the first 170 residues of the fusion protein, and another one at the end of it. Since this virus belongs to the *Flavivridae* family, and this family's FP region is located, around the 100th residue of the fusion protein it's highly likely that the real CSFV fusion peptide is located up until the 165th residue. It is also likely that the sequences predicted at the end of the fusion protein correspond to the TMD.

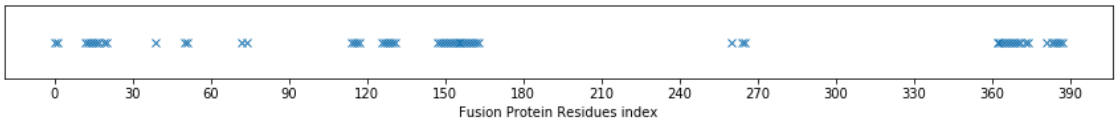


Figure 21 FP indexes predicted by the ensemble of ML models, for Classical swine fever virus Fusion Protein.

5.6.4. Eastern equine encephalitis virus

This virus is a mosquito transmitted disease that can cause severe inflammation of the brain (encephalitis) in horses and humans.

In **Figure 22**, we can observe a prominent predicted FP region around the 80th residue of the fusion protein, and another one at the end. Since this virus belongs to the *Togaviridae* family, and this family's FP region is in the beginning of the fusion protein, it is highly likely that this virus FP is located around that first prominent region. It is also likely that the sequences predicted at the end of the fusion protein correspond to TMD.

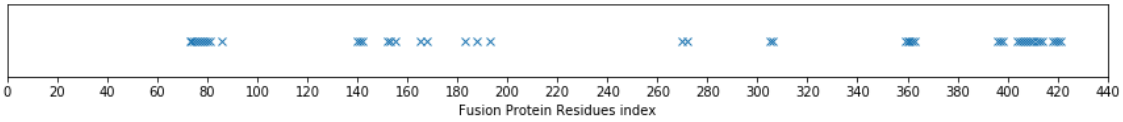


Figure 22 FP indexes predicted by the ensemble of ML models, for Eastern equine encephalitis virus Fusion Protein.

5.6.5. Human Coronavirus

This virus family was a case of deep inconsistencies between databases. Few of them had reference to the fusion peptide, and the ones who had, were completely different from one another.

Coronaviruses are named after the crown-like spikes on their surface. There are four main sub-groups of coronaviruses, known as alpha (e.g.: 229E and NL63), beta (e.g.: OC43 and HKU1), gamma, and delta. Coronaviruses are, as all viruses in this dissertation, enveloped viruses. They contain a positive-sense single-stranded RNA genome and a helical nucleocapsid. The genomic size of coronaviruses ranges from approximately 26 to 32 kilobases, the largest for an RNA virus. The spike (S), envelope (E), membrane (M) and nucleocapsid (N) proteins contribute to the overall structure of all coronaviruses, and this virus FP is usually located in the spike glycoprotein

In **Figure 23** , we can observe three predicted regions up until the 180th residue, another region between 380th and 400th residues, and a final region at the end. Usually this last region corresponds to a TMD, however since the information about this family FP is so uneven, only with experimental evidence could possible to predict which region contains the FP.

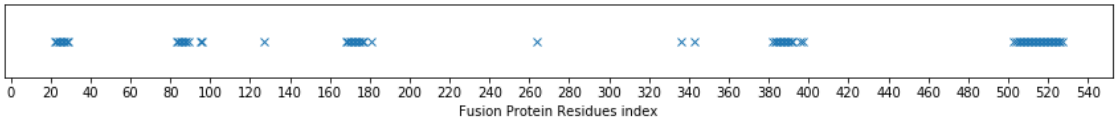


Figure 23 FP indexes predicted by the ensemble of ML models, for Human Coronavirus Fusion Protein.

5.6.6. Omsk hemorrhagic fever virus

Omsk hemorrhagic fever virus (OHFV) is a member of the virus family *Flaviviridae*, and it causes Omsk hemorrhagic fever (OHF). It was first described between 1945 and 1947 in Omsk, Russia from patients with hemorrhagic fever, hence the name.

In **Figure 24**, we can observe a prominent region of prediction at the end of the fusion protein and some dispersed predictions up until the 260th residue. Since this virus belongs to the *Flavivridae* family, and this family's FP region is in the beginning of the fusion protein, one cannot draw conclusions from this prediction.

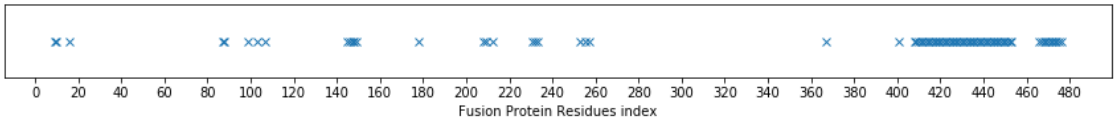


Figure 24 FP indexes predicted by the ensemble of ML models, for Omsk hemorrhagic fever virus Fusion Protein.

5.6.7. Punta toro phlebovirus (PTV)

The Punta Toro virus is a member of the *Phlebovirus* genus of the *Bunyaviridae* family. It was initially isolated from patients in Colombia and two key patients in Panama.

PTV has a helical nucleocapsid as well as an outer envelope. On the viral envelope PTV has two major glycoproteins, Gn and Gc – fusion proteins, that function in host-cell binding and entry. Within the *Phlebovirus* genus, these glycoproteins form a characteristic icosahedral lattice. Due to this structure, PTV appears as a relatively spherical particle when viewed in an electron micrograph with a diameter from 80 to 120 nm [54].

In **Figure 25**, we can observe a prominent region of prediction at the end of the fusion protein, and two other regions between the 120th and the 220th residues. Since this virus belongs to the *Bunyaviridae* family, and this family's FP region is located at the end of the fusion protein, it is likely that this virus FP is contained from the 420th residue forward.

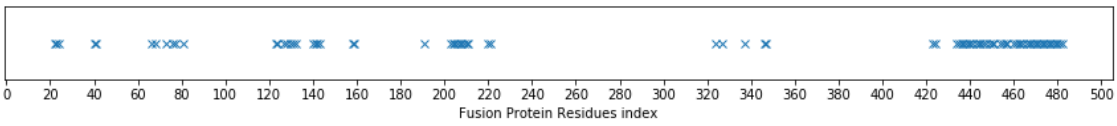


Figure 25 FP indexes predicted by the ensemble of ML models, for Punta toro phlebovirus Fusion Protein.

6

CONCLUSIONS

Enveloped viruses are coated by an outer membrane and, to infect the host cell, these viruses need to fuse the viral and host membranes. The fusion peptide (FP) is one of the most relevant players in this fusion process [1, 2], therefore it is a very promising drug target (as an example, antibodies against dengue virus target this region) [3].

Although several studies focusing on different FPs have been performed, as far as we know there is no systematic and global analysis of viral fusion peptides and the available information was very dispersed. Therefore, a more systematic way to collect and access this information, such as a database, was needed. This was the first fulfilled goal this work provided – dispersed types of information related to fusion peptides (Fusion protein sequence, class, FP sequence, activation mechanism, virus host, structure, etc) was gathered into a single database (more than 800 rows of information).

Additionally, it was not clear how peptides with such distinct characteristics play a common role in membrane fusion. Since machine learning was successfully applied to similar problems regarding anti-microbial peptides, using physicochemical descriptors such as charge, hydrophobicity and specific sequence features, this approach could work on identifying FPs within the entire fusion protein sequence [7, 8]. Indeed, ML was a powerful tool, not only to unveil the most likely region where the FP is located, but also to reveal which were the most significant features in distinguishing this region from other regions within the fusion protein sequence. The results of this work results showed models with scores of accuracy and recall around 90%, and plausible FP predictions on fusion proteins belonging to least studied viruses. The second fulfilled goal of this work promises great impact on the approach of this type of problems, influencing the way FPs, which are possible drug targets, are experimented and tested.

Although all goals proposed in this thesis were accomplished, there is always room for improvement. It was established to prioritize a better recall, so there were many sequences predicted as FP that are not actual FPs. Hence, the algorithms precision should be enhanced. A

change of proportion of the training and cross validation sets from 70/30 to 80/20 may work, even though the number of training and testing samples could not be enough to achieve a robust model. The best solution would be to increase the number of samples of all datasets. However, fusion proteins and fusion peptides still have few conclusive studies about their sequences, structure and activation mechanism and most of the information found on different databases is not consistent, making this a difficult task. Other methods that could also identify FPs location in a fusion protein sequence, such as motif searches, can be a work around for this matter. A different approach for feature selection could also enhance the algorithms precision, especially if structural features such as fusion protein class, or virus family are added to the set of used features to train the model. Since most incorrectly predicted FPs are TMDs, to improve the ability to distinguish the two of them one could use structural information of fusion proteins pre-fusion state. In this state FPs are buried in the fusion protein, and TMDs are exposed since they connect to the membrane.

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

REFERENCES

1. Harrison SC (2015) Viral membrane fusion. Virology 479–480:498–507. https://doi.org/10.1016/j.virol.2015.03.043

2. Wu S, Han J, Liu R, et al (2016) A computational model for predicting fusion peptide of retroviruses. Comput Biol Chem 61:245–250. https://doi.org/10.1016/j.compbiolchem.2016.02.013

3. Harrison SC (2008) Viral membrane fusion. Virology 479–480:498–507

4. Apellániz B, Huarte N, Largo E, Nieva JL (2014) The three lives of viral fusion peptides. Chem. Phys. Lipids 40–55

5. Kam Y, Lee CY, Teo T, et al (2017) Cross-reactive dengue human monoclonal antibody prevents severe pathologies and death from Zika virus infections. JCI Insight 2:1–10. https://doi.org/10.1172/jci.insight.92428

6. Epand RM (2003) Fusion peptides and the mechanism of viral fusion. Biochim. Biophys. Acta - Biomembr. 116–121

7. Lee EY, Fulan BM, Wong GCL, et al (2016) Mapping membrane activity in undiscovered peptide sequence space using machine learning. 113:13588–13593. https://doi.org/10.1073/pnas.1609893113

8. Sharma A, Gupta P, Kumar R, Bhardwaj A (2016) dPABBs: A Novel in silico Approach for Predicting and Designing Anti-biofilm Peptides. New Delhi

9. Tamm LK, Han X (2000) Viral fusion peptides: A tool set to disrupt and connect biological membranes. Charlottesville

10. Ito H, Watanabe S, Sanchez A, Whitt MA (1999) Mutational Analysis of the Putative Fusion Domain of Ebola Virus Glycoprotein. 73:8907–8912

11. Burkard C, Verheije MH, Wicht O, et al (2014) Coronavirus Cell Entry Occurs through the Endo-/Lysosomal Pathway in a Proteolysis-Dependent Manner. PLoS Pathog 10:. https://doi.org/10.1371/journal.ppat.1004502

12. White JM, Delos SE, Brecher M, Schornberg K (2008) Structures and Mechanisms of Viral Membrane Fusion Proteins. Crit Rev Biochem Mol Biol 43:189–219. https://doi.org/10.1080/10409230802058320.Structures

13. Gibbons DL, Vaney MC, Roussel A, et al (2004) Conformational change and protein-protein interactions of the fusion protein of Semliki Forest virus. Nature 427:320–325. https://doi.org/10.1038/nature02239

14. Gaudin Y (2000) Rabies Virus-induced Membrane Fusion Pathway. J Cell Biol 150:601–611

15. Gilbert-Ross M, Konen J, Koo J, et al (2017) Targeting adhesion signaling in KRAS, LKB1 mutant lung adenocarcinoma. JCI Insight 2:1–11. https://doi.org/10.1172/jci.insight.90487

16. Han X, Bushweller JH, Cafiso DS, Tamm LK (2001) Membrane structure and fusion-triggering conformational change of the fusion domain from influenza hemagglutinin. Nat Struct Biol 8:715–720. https://doi.org/10.1038/90434

17. Lorieau JL, Louis JM, Bax A (2010) The complete influenza hemagglutinin fusion domain adopts a tight helical hairpin arrangement at the lipid:water interface. Proc Natl Acad Sci 107:11341–11346. https://doi.org/10.1073/pnas.1006142107

18. Haque ME, Koppaka V, Axelsen PH, Lentz BR (2005) Properties and structures of the influenza and HIV fusion peptides on lipid membranes: Implications for a role in fusion. Biophys J 89:3183–3194. https://doi.org/10.1529/biophysj.105.063032

19. Lai AL, Moorthy AE, Li Y, Tamm LK (2012) Fusion activity of HIV gp41 fusion domain is related to its secondary structure and depth of membrane insertion in a cholesterol-dependent fashion. J Mol Biol 418:3–15. https://doi.org/10.1016/j.jmb.2012.02.010

20. Melo MN, Sousa FJR, Carneiro FA, et al (2009) Interaction of the Dengue Virus Fusion Peptide with Membranes Assessed by NMR : The Essential Role of the Envelope Protein Trp101 for Membrane Fusion. J Mol Biol 392:736–746. https://doi.org/10.1016/j.jmb.2009.07.035

21. Rowse M, Qiu S, Tsao J, et al (2015) Characterization of potent fusion inhibitors of influenza virus. PLoS One 10:1–15. https://doi.org/10.1371/journal.pone.0122536

22. Kolokoltsov AA, Davey RA (2004) Rapid and Sensitive Detection of Retrovirus Entry by Using a Novel Luciferase-Based Content-Mixing Assay. J Virol 78:5124–5132. https://doi.org/10.1128/jvi.78.10.5124-5132.2004|issn

23. Awad M, Khanna R (2015) Efficient learning machines: Theories, concepts, and applications for engineers and system designers

24. Mitchell T (1997) Machine Learning. McGraw-Hill

25. Smola A, Vishwanathan SVN (2014) Introduction to machine learning. Methods Mol Biol 1107:. https://doi.org/10.1007/978-1-62703-748-8-7

26. Rocha M, Ferreira PG (2017) Análise e Exploração de Dados com R, 03-2017th ed

27. Rocha M, Neves JM, Cortez P (2008) Análise Inteligente de Dados

28. Seligman SJ (2008) Constancy and diversity in the flavivirus fusion peptide. Virol J 5:1–10. https://doi.org/10.1186/1743-422X-5-27

29. Saeys Y, Inza I, Larranaga P (2007) A review of feature selection techniques in bioinformatics. 19 23:2507–2517

30. NCBI. https://blast.ncbi.nlm.nih.gov/Blast.cgi. Accessed 17 Jan 2018

31. BLAST Homepage and Selected Search Pages. ftp://ftp.ncbi.nlm.nih.gov/pub/factsheets/HowTo_BLASTGuide.pdf. Accessed 17 Jan 2018

32. Finn RD, Clements J, Arndt W, et al (2015) HMMER web server: 2015 Update. Nucleic Acids Res 43:W30–W38. https://doi.org/10.1093/nar/gkv397

33. Crooks G, Hon G, Chandonia J, Brenner S (2004) NCBI GenBank FTP Site\nWebLogo: a sequence logo generator. Genome Res 14:1188–1190. https://doi.org/10.1101/gr.849004.1

34. Crooks GE, Hon G, Chandonia J-M, Brenner SE WebLogo. http://weblogo.berkeley.edu/. Accessed 14 Jan 2018

35. Bateman A, Martin MJ, O'Donovan C, et al (2017) UniProt: The universal protein knowledgebase. Nucleic Acids Res 45:D158–D169.

https://doi.org/10.1093/nar/gkw1099

36. Chang J, Chapman B, Friedberg I, et al (2008) Biopython Tutorial and Cookbook

37. Pedregosa F, Varoquaux G, Gramfort A, et al (2012) Scikit-learn: Machine Learning in Python. 12:2825–2830. https://doi.org/10.1007/s13398-014-0173-7.2

38. Cao D, Xu Q, Liang Y (2013) Systems biology propy : a tool to generate various modes of Chou ' s PseAAC. 29:960–962. https://doi.org/10.1093/bioinformatics/btt072

39. Yee LC, Wei YC (2012) Current Modeling Methods Used in QSAR/QSPR. Stat Model Mol Descriptors QSAR/QSPR 2:1–31. https://doi.org/10.1002/9783527645121.ch1

40. Mitchell B.O. JBO (2014) Machine learning methods in chemoinformatics. Wiley Interdiscip Rev Comput Mol Sci 4:468–481. https://doi.org/10.1002/wcms.1183

41. Schneider G, Baringhaus KH (2013) De novo design: From models to molecules. novo Mol Des 1–55. https://doi.org/10.1002/9783527677016.ch1

42. Arora J (2011) Introduction to Optimum Design, Third Edition

43. Shoval O, Sheftel H, Shinar G, et al (2012) Evolutionary trade-offs, pareto optimality, and the geometry of phenotype space. Science (80) 336:1157–1160. https://doi.org/10.1126/science.1217405

44. Schmidt NW, Lis M, Zhao K, et al (2012) Molecular basis for nanoscopic membrane curvature generation from quantum mechanical models and synthetic transporter sequences. J Am Chem Soc 134:19207–19216. https://doi.org/10.1021/ja308459j

45. Park CB, Kim HS, Kim SC (1998) Mechanism of action of the antimicrobial peptide buforin II: Buforin II kills microorganisms by penetrating the cell membrane and inhibiting cellular functions. Biochem Biophys Res Commun 244:253–257. https://doi.org/10.1006/bbrc.1998.8159

46. Markov P, Monte C (1992) Gelman, A 7:457–472. https://doi.org/10.1214/ss/1177011136

47. Mohanram H, Nip A, Domadia PN, et al (2012) NMR structure, localization, and vesicle fusion of chikungunya virus fusion peptide. Biochemistry 51:7863–7872. https://doi.org/10.1021/bi300901f

48. Agopian A, Quetin M, Castano S (2016) Structure and interaction with lipid membrane models of Semliki Forest Virus Fusion Peptide. BBA - Biomembranes 1858:2671–2680. https://doi.org/10.1016/j.bbamem.2016.07.003

49. Varsanyi TM, Kovamees J, Norrby E (1991) Molecular cloning and sequence analysis of human parainfluenza type 2 virus mRNA encoding the fusion glycoprotein. J Gen Virol 72:89–95. https://doi.org/10.1099/0022-1317-72-1-89

50. Bagai S, Lamb RA (1997) A glycine to alanine substitution in the paramyxovirus SV5 fusion peptide increases the initial rate of fusion. Virology 238:283–290. https://doi.org/10.1006/viro.1997.8858

51. Lai AL, Freed JH (2015) The Interaction between Influenza HA Fusion Peptide and Transmembrane Domain Affects Membrane Structure. Biophys J 109:2523–2536. https://doi.org/10.1016/j.bpj.2015.10.044

52. Duan L, Du J, Wang X, et al (2016) Structural and functional characterization of EIAV gp45 fusion peptide proximal region and asparagine-rich layer. Virology 491:64–72. https://doi.org/10.1016/j.virol.2016.01.010

53. Parkman PD (1996) Togaviruses: Rubella Virus. In: Baron S (ed) Medical Microbiology, 4th ed. Galveston (TX): University of Texas Medical Branch at Galveston

54. Guu TSY, Zheng W, Tao YJ (2012) Bunyavirus: structure and replication. pp 245–266

ATTACHMENTS

Attachment A - Data's summary

Protein Sequences (188 Total)	Class	I	104
		II	65
		III	0
	Family	*Arenaviridae*	0
		Bunyaviridae	0
		Coronaviridae	3
		Filoviridae	11
		Flaviviridae	55
		Hepadnaviridae	0
		Herpesviridae	1
		Orthomyxoviridae	0
		Paramyxoviridae	27
		Poxviridae	0
		Retroviridae	64
		Rhabdoviridae	0
		Togaviridae	27
Peptide Sequences (238 total)	Class	I	134
		II	65
		III	12
	Family	*Arenaviridae*	5
		Bunyaviridae	5
		Coronaviridae	3
		Filoviridae	15
		Flaviviridae	54
		Hepadnaviridae	1

		Herpesviridae	8
		Orthomyxoviridae	19
		Paramyxoviridae	34
		Poxviridae	3
		Retroviridae	64
		Rhabdoviridae	6
		Togaviridae	21

Table M Data's summary referred to Fusion Proteins and Peptides sequences divided by class and family.

Attachment B - Mapping of the used features

HYDROPHOBICITY = {1 : RKEDQN, 2 : GASTPHY, 3 : CLVIMFW}

1 - STAND FOR POLAR;

2 - STAND FOR NEUTRAL;

3 - STAND FOR HYDROPHOBICITY.

POLARITY = {1 : LIFWCMVY, 2 : CPNVEQIL , 3 : KMHFRYW}

1 - STAND FOR (4.9-6.2);

2 - STAND FOR (8.0-9.2);

3 - STAND FOR (10.4-13.0).

CHARGE={1:KR , 2 : ANCQGHILMFPSTWYV , 3 : DE }

1 - STAND FOR POSITIVE;

2 - STAND FOR NEUTRAL;

3 - STAND FOR NEGATIVE.

SECONDARYSTR={ 1 : EALMQKRH , 2 : VIYCWFT , 3 : GNPSD }

1 - STAND FOR HELIX;

2 - STAND FOR STRAND;

3 - STAND FOR COIL.

SOLVENTACCESSIBILITY={ 1 : ALFCGIVW , 2 : RKQEND , 3 : MPSTHY }

1 - STAND FOR BURIED;

2 - STAND FOR EXPOSED;

3 - STAND FOR INTERMEDIATE.