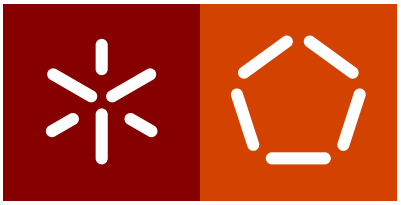
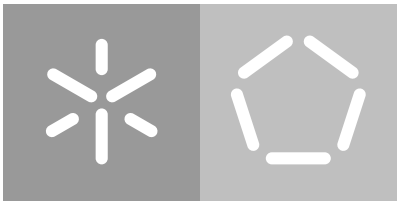


Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Pedro Porto Dias

PhagePro: Prophage finding tool

July 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

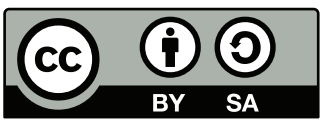
João Pedro Porto Dias

PhagePro: Prophage finding tool

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Oscar Manuel Lima Dias
Luís Daniel Rodrigues de Melo

July 2021



ACKNOWLEDGMENTS

Ever Tried. Ever Failed. No matter. Try again. Fail again. Fail better.
Samuel Beckett

Começo por agradecer aos meus orientadores, Oscar Dias e Luís Melo, pelo interessante projecto proposto e pela disponibilidade e prontidão demonstrada na resolução de dúvidas que foram surgindo no decorrer da dissertação. Um obrigado também ao professor Miguel Rocha pela ajuda na transferência de mestrado e por me orientar na busca de projecto de tese.

Aos colegas do laboratorio agradeço a paciência e o tempo dispendidos para me explicar e ajudar na resolução de problemas mais técnicos que por vezes se demonstravam uma 'dor de cabeça'.

Aos 'colegas de vila real' que, embora longe fisicamente, sempre se mostraram disponveis para ajudar no que fosse preciso e para me dar apoio quando as coisas corriam menos bem.

Quero continuar por agradecer aos colegas da sala 0.09 e aos colegas de mestrado pela companhia e o tempo dispendido na partilha de momentos, sejam eles momentos elucidativos ou momentos de lazer. Reparando bem, todos os membros integrantes da sala tinhas diferenças significativas e tinham para oferecer diferentes prespectivas. Isso fez com que a vossa companhia me tornasse uma pessoa mais completa e certamente me melhorou o meu trabalho. Obrigado.

Quero agradecer à minha familia por me fazerem ser quem sou e por me terem dado os recursos e apoio necessários para concretizar os meus objectivos de vida.

Este estudo contou com o apoio da Fundação para a Ciência e Tecnologia (FCT) portuguesa no âmbito do financiamento estratégico da unidade UIDB/04469/2020. A

obra também foi parcialmente financiada pelo Projeto PTDC/SAU-PUB/29182/2017 [POCI-01-0145-FEDER-029182].

STATEMENT OF INTEGRITY

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

ABSTRACT

Bacteriophages are viruses that infect bacteria and use them to reproduce. Their reproductive cycle can be lytic or lysogenic. The lytic cycle leads to the bacteria death, given that the bacteriophage hijacks hosts machinery to produce phage parts necessary to assemble a new complete bacteriophage, until cell wall lyse occurs. On the other hand, the lysogenic reproductive cycle comprises the bacteriophage genetic material in the bacterial genome, becoming a prophage. Sometimes, due to external stimuli, these prophages can be induced to perform a lytic cycle. Moreover, the lysogenic cycle can lead to significant modifications in bacteria, for example, antibiotic resistance.

To that end, PhagePro was created. This tool finds and characterises prophages inserted in the bacterial genome. Using 42 features, three datasets were created and five machine learning algorithms were tested.

All models were evaluated in two phases, during testing and with real bacterial cases. During testing, all three datasets reached the 98 % F1 score mark in their best result. In the second phase, the results of the models were used to predict real bacterial cases and the results compared to the results of two tools, Prophage Hunter and PHASTER. The best model found 110 zones out of 154 and the model with the best result in dataset 3 had 94 in common.

As a final test, *Agrobacterium fabrum strC68* was extensively analysed. The results show that PhagePro was capable of detecting more regions with proteins associated with phages than the other two tools.

In the ligth of the results obtained, PhagePro has shown great potential in the discovery and characterisation of bacterial alterations caused by prophages.

Keywords: Virus, Bacteriophages, Machine learning, Galaxy, PhagePro

RESUMO

Bacteriófagos são vírus que infetam bactérias usando-as para garantir a manutenção do seu genoma. Este processo pode ser realizado por ciclo lítico ou lisogénico. O ciclo lítico consiste em usar a célula para seu proveito, criar bacteriófagos e lisar a célula. Por outro lado, no ciclo lisogénico o bacteriófago insere o seu código genético no genoma da bactéria, o que pode levar à transferência de genes de interesse, tornando-se importante uma monitorização dos profagos.

Assim foi desenvolvido o PhagePro, uma ferramenta capaz de encontrar e caracterizar bacteriofagos em genomas bactérias. Foram criadas features para distinguir profagos de bactérias, criando três datasets e usando algoritmos de aprendizagem de maquina.

Os modelos foram avaliados durante duas fases, a fase de teste e a fase de casos reais. Na primeira fase de testes, o melhor modelo do dataset 1 teve 98% de F1 score, dataset 2 teve 98% e do dataset 3 também teve 98%. Todos os modelos, para teste em casos reais, foram comparados com previsões de duas ferramentas Prophage Hunter e PHASTER. O modelo com os melhores resultados obteve 110 de 154 zonas em comum com as duas ferramentas e o modelo do dataset 3 teve 94 zonas.

Por fim, foi feita a análise dos resultados da bactéria *Agrobacterium fabrum strC68*. Os resultados obtidos mostram resultados diferentes mas válidos, às ferramentas comparadas, visto que o PhagePro consegue detectar zonas com proteínas associadas a fagos que as outras tools não conseguem.

Em virtude dos resultados obtidos, PhagePro mostrou que é capaz de encontrar e caracterizar profagos em bacterias.

Palavras-Chave: Virus, Bacteriófagos, Aprendizagem de maquina, Galaxy, PhagePro

CONTENTS

ACRONYMS

BLAST	Basic Local Alignment Search Tool
BLASTp	Protein BLAST
PHAST	Phage Search Tools
PHASTER	Phage Search Tool Enhanced Release
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
dsDNA	Double-stranded DNA
dsRNA	Double-stranded RNA
ssDNA	Single-stranded DNA
ssRNA	Single-stranded RNA
SDS-PAGE	Sodium dodecyl sulfate–polyacrylamide gel electrophoresis
NCBI	National Center for Biotechnology Information
WU BLAST	Washington University BLAST
HMM	Hidden Markov Models
tRNA	Transfer RNA
tmRNA	Transfer-messenger RNA

BLASTn	Nucleotide BLAST
ACLAME	A CLAssification of Mobile genetic Elements
CDS	Coding sequence
PGDR	Phage-like dense region
GLIMMER	Gene Locator and Interpolated Markov ModelER
DBSCAN	Density-based spatial clustering of applications with noise
ORF	Open reading frame
PHANTOME	PHage ANnotation TOols and MEthods
pVOG	Prokaryotic Virus Orthologous Groups
CD-HIT	Cluster Database at High Identity with Tolerance
tBLASTn	Translated nucleotide BLAST
PAM	Point Accepted Mutation
BLOSUM	BLOcks of Amino Acid Substitution Matrix
SVM	Support vector machine
RF	Random forests
DT	Decision trees
RBF	Radial basis function
TP	True positive
TN	True negative
FP	False positive

FN	False negative
ROC	Receiver Operating Characteristics
AUC	Area Under the Curve
ECOD	Evolutionary Classification of Protein Domains
Gb	Genbank
PPV	Positive Predictive Value

LIST OF TABLES

1

INTRODUCTION

1.1 MOTIVATION

Bacteriophages are viruses that have high specificity to bacteria and each bacteriophage strain has a small range of hosts that can infect. Furthermore, bacteriophages have two lifecycles, lytic or lysogenic. The lytic cycle is an aggressive lifecycle where the bacteriophage hijacks the hosts machinery to replicate phage parts and create a new phage progeny until the lysis occur and there is the release of the new phages and the death of the bacteria. On the other hand, on the lysogenic lifecycle, the bacteriophage integrates bacterial genome into the genomic pool of the bacteria and successive bacterial generations called a prophage. In some cases, external stimuli can induce the integrated genomic material to adopt an lytic lifecycle that can lead to bacteria death.

These organisms are present in large quantities among all types of environments and have an important role in the dynamic of the environment, independently of the lifecycle, either by reducing the population of a bacteria or by transducing genes of interest. Additionally, their genome is very plastic, with the exception of core genes, meaning that throughout the bacteriophage lifecycles, new proteins could be created or transferred from one bacteria to another leading to alterations in the dynamics of the environment.

These characteristics have awakened scientific interest to explore the possibilities that bacteriophages have in various industries. One example is in the medical field where bacteriophages are being used to treat infections without reported side effects, presenting a possible alternative to antibiotics. Other example that can be explored is precise modifications of bacteria to produce compounds of interest in an optimized way, reducing secondary products or energy consumption. But for these to be possible, an extensive analysis of the bacteriophages is required.

Currently, there are two types of methods to study bacteriophages, *wet lab* or bioinformatics approach. The first approach is more precise but it is limited to the specific situation, offers less generalisation possibilities and requires trained professionals and specialised facilities to produce good results. On the other hand, the increase of phage sequencing increases phage knowledge that allows a bioinformatics approach. This approach has more flexibility to new information, can process more data and does not require highly trained professionals but lacks the depth to each case that the *wet lab* approach offers. Hence, although great strides were made in the bioinformatics field, there are ways to improve and create bioinformatics tools and help scientists to reach their objectives.

So in this work, a supervised machine learning approach is used to predict prophages in bacteria and characterise the found prophages. This approach is based in structural and compositional differences between bacteria and phages that allow a mathematical algorithm to detect differences. This type of algorithm is trained with data features or characteristics that are going to be analysed in every sample and mapped by the algorithm to form a model. After the model learns all features of the data, it can be used to predict new samples that the model has never seen. Beyond this type of learning there are unsupervised and reinforcement learning that learn from data in different ways.

1.2 GOALS

The main objective is to implement tools that can predict and characterise prophages. Below the objectives are described in more detail.

- Review of the state of the art on already existing computational tools.
- Analyse the most commonly used computational tools (PHAST, PHASTER, Prophage Hunter).
- Creating a pipeline that allows the predicting and characterising prophages.
- Implementing the pipeline.

1.3 SCHEDULE

Starting data: November 1st 2019

Week 1-12: Reviewing the state of art.

Week 6-15: Analysing the existing tools for prophage search (PHASTER).

Week 15-30: Developing a pipeline for prophage search and analysis.

Week 31-38: Implementation of the previously developed pipeline.

Week 13-40: Writing the thesis.

1.4 STRUCTURE

This thesis is organised in five chapters.

In the first chapter, I describe my motivation, goals, and the thesis' schedule, including each task's time frame.

In the second chapter, the state of the art is presented. This chapter includes a brief introduction to the viral world, describing the history, viral reproduction, and classification, exploring the bacteriophage thematic. Furthermore, various tools and databases are described and analysed to clarify the current status quo, and outline objectives.

The third chapter includes steps for each tool structure and algorithm.

The fourth chapter illustrates how the tools and algorithms were implemented.

The fifth chapter presents and discusses the tool's results in standard datasets to compare the results with other tools.

The sixth chapter provides the conclusions. This chapter includes future improvements and other possible goals.

2

STATE OF THE ART

2.1 VIRUS AND BACTERIOPHAGES

2.1.1 VIRUS HISTORY

In 1977, three divisions were proposed, *Eubacteria, Archaebacteria and Urkaryotes*, mostly due to differences in their ribosomes [1]. The nomenclatures was changed to *Bacteria, Archaea and Eukarya* to avoid confusion between prokaryote domains [2]. Nowadays, it is possible to prove these divisions through comparative biochemistry and genomics [3]. In contrast, viruses have been studied by the scientific community for a long time without being identified, as various scientists noticed an infectious agent without recognising the exact cause. Only advances in filtration systems and electronic microscopy, allowed for the isolation these viruses. In 1948, Holmes [4], proposed a classification into three groups: *Phaginae*, which infect bacteria, *Phytophaginae*, which infect plants, and *Zoophaginae*, which infect animals [5]. Furthermore, in 1950, the Virus subcommittee of the International Nomenclature Committee, proposed eight criteria to classify virus. In 1957, *Lwoff* summarised viral characteristics that described different viruses to clarify the subject [6]. Another distinction was proposed in 1962 when *Lwoff, Horne* and *Tournier* suggested a classification based on their genetic material and characteristics [7]. As more information was becoming available, in 1984, *Abeles et al.*.

classified virus-based on various characteristics like morphology, genetic material, host [8], making the classification more precise.

2.1.2 Virus classification

Viral classification has become more reliant on chemical information and morphology, as more information became available, and the increased complexity of virus profiling. The division between viruses can be based on four aspects: genetic material characteristics, capsid symmetry, presence or absence of envelope and other capsid information [9, 10].

Undoubtedly, the viral genetic material alters the phenotype of the virus. Hence, the virus can be divided by the type of genetic material present, DNA or RNA, and the type of strand, linear or circular, and double or single. DNA viruses have large variability (single or double-stranded, linear, or circular) despite high genetic stability. These viruses are highly regulated, frequently using enzymes from the host, and their gene expression is usually divided into two phases. The first includes the synthesis of DNA and proteins required for DNA replication, and the second involves the expression of structural proteins. However, gene expression can be divided into early gene expression, middle gene expression and late gene expression. Early gene expression uses hosts machinery (polymerase, sigma factor and promoters) to initiate gene transcription of promoters needed in DNA synthesis and used in regulatory functions, namely, repression of early gene transcription, and transcription of middle genes. In the middle gene expression, various regulatory factors, and promoters, required in the late gene expression, are transcribed. In late gene expression, virion components, like structural proteins, and lysis enzymes are produced [11–14].

Other differentiating virus characteristics include the different host requirements and with the increase of genome size, the virus becomes more independent. Furthermore, host's dependency is associated with the need for host enzymes, like the DNA poly-

merase, and deoxynucleotide triphosphate precursors [15]. RNA viruses are divided into by two groups: riboviruses and retroviruses. Riboviruses are RNA-dependent RNA polymerase (RbRp) viruses. These viruses need a replicase complex, constituted by RbRp, helicases, ATPases, and other proteins, to replicate their genetic material [16]. Alternatively, retroviruses will encode reverse transcriptase and retroviral *integrase* to convert their RNA to DNA [17]. Compared with DNA viruses, both riboviruses and retrovirus, show higher mutation rates, associated with fewer mismatch repair mechanisms [18]. However, repair mechanisms and repair structures, e.g. DNA polymerase, can replicate genetic material to maintain genetic stability, preventing lethal mutations. An advantage for the lack of replicative control, is the genetic heterogeneity. Hence, similar phenotypes can be obtained through different mechanisms, facilitating adaptation to new environments, having a larger ”genetic range” or robustness. These viruses can also be classified as positive and negative strand. Positive strand viruses are mostly single-stranded that encode functional mRNAs. After infection, these viruses can be translated without other processes. Negative strand viruses cannot produce directly infectious virus and cannot be translated immediately, needing a transcription process first. Ambience viruses are a hybrid of both conditions, i.e. can directly transcribe functional mRNA's [16].

As mentioned before, viral morphology also weights in the classification of viruses. The first case is capsid symmetry, e.g. helical and icosahedral symmetry. The capsid is the membrane closest to the genetic material and is constituted by repetitions of the same or similar proteins. In helical symmetry, these proteins are placed in a spiral around the genetic material. Each protein is ordered and, except for the protein chain extremities, has the same structure, with the same number of neighbour proteins. In contrast, icosahedral symmetry has a more spherical shape. The icosahedral structure follows a mathematical rule, and the three-dimensional structure can be calculated. This structure involves all genetic material, and can act as a limiter on the viral genetic material's size and slow evolutionary changes [19]. Additionally, more complex viruses can have another layer, called envelope. This envelope is composed of lipids and proteins

from the virus and the host. Therefore, each envelope will have different compositions, since the host and the proteins coded by the virus may vary. The different composition will provide distinct abilities. For instance, different receptors in this membrane will result in a different capacity to infect hosts. Furthermore, enzymes that could be integrated into the membrane and facilitate the infection process can also be present [20].

The concept of host in viral classification relies on the specific host range that each virus has. This range can be defined as the spectrum of host species that can support a specific viral life cycle. The virus needs to enter the host, release its genetic material, replicate, or integrate their genetic material in the host's genetic pool. For all these crucial tasks, various protein complexes, enzymes and environment conditions are required and, considering that virus only encode a percentage of this information in their genetic material, host support is required. This leads to viral dependence on certain host factors, restricting the virus infection range. However, genetic variations could reduce this dependence and allow to a broader spectrum of infection [21–23]. Viruses can be distinguished as plant, animal and bacteria viruses or bacteriophage.

2.1.3 Bacteriophages

According to the latest ratification (March 2021) of the International Committee on Taxonomy of Viruses, or ICTV [24, 25], phages are distributed in distinct taxonomic groups, according to their morphological characteristics, ranging from phages with dsDNA, ssDNA, dsRNA and ssRNA. Additionally, different characteristics in their morphology are also considered, e.g., in the head proteins or tail proteins (structural proteins). One specific example is the Order Caudovirales [24] being divided in nine families, accordingly to the shape of the tail. One example are the three families, *Myoviridae, Podoviridae, and Siphoviridae* representing the vast majority of phages. These phage have a tail, a long protein structure involved in bacterial phage attachment, and a double strand DNA

[3, 26]. More examples of tail morphologies are the polyhedral shape group, that has seven families, having either single or double-stranded RNA or DNA, single or double-stranded, the filamentous shape group has three families and can have double-strand or single-strand DNA and finally, the pleomorphic shape group has six families and has double-stranded DNA.

As the bacteriophage have high specificity to bacteria, an emerging interest developed in the analysis of bacteriophages to fight bacteria without disrupting the microbiome [27] and the capacity of viral transduction, i.e., the transference of genetic material between bacteria, to add traits or to understand how bacteria can evolve from phage's [28]. However, the interest in the use of phage's is not new. In 1931, experiments were performed by *Félix d'Herelle*, in patients with bacterial infections [29]. Years later, *Bruynoghe* and *Maisin*, published their work in viral use to treat cutaneous furuncles and carbuncles [30]. Although this type of therapy was forgotten for many years, in part, due to the rise of antibiotic treatment, experiments with promising results have emerged in recent years, e.g. in humans, treatment for infections by *Escherichia coli* or *Pseudomonas aeruginosa* have been tested [31] and, in the food industry, phage therapy is gaining its place has a regulator of gut problems in animals, precisely controlling bacteria's presence. Moreover, in plants and processed food, phage's have shown promising results in reducing bacteria' presence [32]. An impressive characteristic is the phage abundance in every biome and microbiome, and the role these viruses perform [33, 34].

This type of treatment has advantages in bacterial infections in comparison with antibiotics. The specificity of the bacteriophages preserves the microbiome, due to the phages being described only to affect the intended bacteria, leaving the remaining biome unaffected, reducing side effects of possible microbiome disruption. Furthermore, it can affect both Gram-positive and Gram-negative bacteria, only depending on the chosen phage [35–37]. Also, if bacteria adapt to the virus [38, 39], genetic engineering can surpass the new bacteria defences [40]. In addition to the use of phages, phage-

derived proteins can be used against bacterial infections, namely endolysins [41]. This characteristic somewhat shows the control that virus have, in this case, in bacteria. Thus, phage therapy is considered a solution to the growing antibiotic resistance bacteria [34]. Another opportunity favouring phage study is the evaluation of their capacity for horizontal gene transfer. This process is denominated transduction whereas genes are transferred from one bacteria another, using phages as vectors [42]. These genes can encode antibiotic resistance proteins, leading to further complications in controlling microorganism populations.

Although phage therapy has promising results on the treatment of bacterial infections, there are limitations that need to be considered. First, for a narrow and precise performance against the infection, the bacteria must be identified prior to treatment. One possibility relies on the use of a broader combinations of phages (phage cocktails). The treatment outcome could vary between patients, e.g., the speed that the body defences act. Hence, the results may not be stable and have different results [27]. Another limitation is that the phage used in to infect the bacteria needs to have a lytic cycle, as only this lifecycle has a predictable outcome of killing bacteria. On the other hand, lysogenic cycle integrates the bacterial genome and cannot have a immediate response to kill the bacteria. Furthermore, phage genetic material can be integrated into the bacterial genome and provide advantages to the bacteria [43], due to this genetic material having selective benefits to the bacteria concerning the environment or the host, e.g., antibiotic resistance [44, 45]. This limitation reveals another problem for therapy application. The bacteriophage genetic material must be identified and sequenced to discard genes like *integrase*, which are responsible for integrating of the genetic material and not the bacterium death, missing the purpose of the phage infection. Other factors include genes responsible for bacterial virulence and resistance against harsh environments [46].

One example of an interesting case study is *Helicobacter pylori*. This bacterial species is highly prevalent in the human population and inhabits the human digestive tract [47]. However, the relationship between humans and *H. pylori* is not new, as this bacteria

can lead to complications in the human host, e.g. carcinomas, ulcers, and alterations in the gut microbiome [48, 49]. Usually, the treatment for these diseases involves using antibiotics like ampicillin or tetracycline. However the rise of antibiotic resistance is a problematic and socioeconomic limitations are becoming a more significant obstacle for the simple antibiotic solution, encouraging scientists to look for new alternatives [50–52]. These factors endorse the study of bacteriophages, using them to infect and lyse *H pylori*, without causing other complications and side effects associated with the use of antibiotics. This treatment has shown promising results in other diseases related with infections caused by, for instance, *Staphylococcus* or *Pseudomonas*, independently of sex or age [53].

2.1.4 Bacteriophage reproduction

The phage's genome path inside a bacteria can be categorised by his reproductive cycle, e.g. lytic [54] or lysogenic [55] (1) and can be divided into six steps, with three being optional or requiring external stimuli: attachment (performed by both), penetration(performed by both), genetic material integration (performed by both), biosynthesis(performed in lytic cycle, lysogenic require external factors), maturation (performed in lytic cycle, lysogenic require external factors) and lysis (performed in lytic cycle, lysogenic require external factors add ref). In both cycles, the bacteriophage adheres to the receptors in the cell wall. In the lytic cycle, after the introduction of the genome in the cytoplasm, the bacteriophage will use the cell replication machinery to replicate the genomic material, leading to the creation of viral parts, e.g. tail proteins and capsid proteins, to assemble more bacteriophage particles. In the end, the cell wall lysis will occur, and newly created bacteriophage will be released to the environment. In the lysogenic cycle, after the genetic material enters the host, the phage genome is integrated into the bacterial chromosome or remains as an episomal element. The phage genetic material

integrated into the bacterial genetic pool is called prophage. This process is usually mediated by the integrase gene that recognises specific attachment sites (based in genetic material homology) named bacterial attachment site (attB) and phage attachment site (attP). Posteriorly, the phage genetic material is replicated and passed to its progeny, without killing the host [56–58]. Sometimes, external stimuli induces replication of viral proteins, which leads the phage to enter lytic cycle [59].

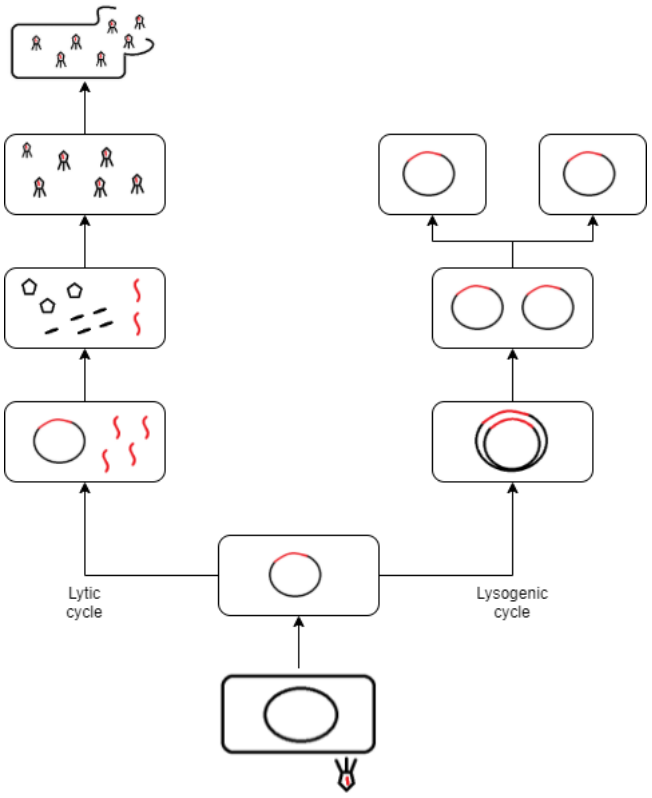


Figure 1: Schematic represention the steps in the lytic and lysogenic phage lifecycles. On the rigth (lysogenic cycle), the phage integrates his genetic material in the bacterial genome passed to next bacterial generations. On the left (lytic cycle), the phage highjacks bacterial machinery and uses it to produce phage parts until the bacteria is lysed.

2.1.5 Bacteriophage identification

In vitro options for phage identification in phage identification, e.g. Cryo-electron microscopy, SDS-PAGE, electronic microscopy, and DNA methods [3]. For example, Cryo-electron microscopy consists of freezing the sample and then analysing the physical structure with computer software. These steps need special equipment and a set of meticulous procedures which have high costs and require time and, specialised technical expertise [60, 61]. Another example is SDS-PAGE, which consists of protein separation, based on its molecular weight [62]. Additionally, if the phage has a non-active lytic cycle, external stimuli, e.g. ultraviolet light, could prove that the bacteria is infected with a phage by inducing the lytic cycle and producing clear spots in the bacterial culture. Furtheremore, as the genetic material is integrated in the bacterial genome, bioinformatics tools started being used on more detailed analyses. These tools started being used when genetic sequencing had cost reductions and yield improvements. Such condition improvements, allowed an accurate read of genetic material that would be used in more complex and explanatory analyses, by identifying features from the bacterial genome [63], which derive from compositional differences between the phage sequences and the bacterial genetic material of different origins, and proteins of interest like *integrase*. Furthermore, when analysing gene presence and distribution, phage genes show a higher homogenised order than bacterial genes, with less strand shifts and higher gene concentration. Moreover, sequence patterns, oligonucleotide frequencies, amino-acid composition, and dinucleotide frequencies are differential factors between phages and bacteria [64–66].

In various studies [67, 68], phage concentration in bacteria was evaluated. An extensive range of prophage presence in bacteria variating with strain was discovered, in some cases accounting for ten percent of the bacterial genomes. Genome size seems to influence the existence and integration as well. Thus, genomes could have suffered an evolutionary pressure to eliminate all non-essential genetic material. Furthermore,

the different phage characteristics can be used as features for phage identification *in silico*. The most relevant differences are GC content [69], oligonucleotide frequencies, codon usage from hosts' genome, and identity to previously annotated known phage's [70]. Another method compares genes with key functions in the bacteriophages, e.g. integration or lysis. However, this method alone could lead to errors as bacteria can produce particles that resemble phage structures, e.g., phage tails and bacterial secretion systems or bacteriocins (antimicrobial compound) [71–73].

2.1.6 Phage gene hallmarks

Phage prediction was often associated with genetic structures thought to be related to phage traits. On the other hand, these structures have a bacterial origin, rather than having exclusivity in phages. For example, gene transfer agents (GTA) [74] are phage like particles present in *Bacteria* and *Archaea* that help in the horizontal gene transfer. Additionally, there are pathogenic islands [75], consisting in additional genes received by horizontal gene transfer and an encapsulins [76] that are a shell-forming proteins.

Phages can easily lose or gain genetic material, which leads to great diversity and small percentage of redundancy in their genetic sequences. Furthermore, this suggests that most genetic material encode genes and consequently, have a higher GC content. Another factor that affects viral genetic composition is the host. Phages tend to adapt their genetic material to the host, not being recognised by the proteins and enzymes that degrade foreign DNA [77–79]. Hence, phage genetic mosaicity must be considered as an essential feature in phage evolution. Thus, sequence recombination is frequent in phage's [80] and suggests that phage characterisation through genetic material would be almost impossible. However, essential genes tend to maintain their identity, such as head or tail genes, genes involved in DNA replication, and nucleotide metabolism genes.

These genes are maintained to keep structure and interactions between genetic material avoiding loss of functions [81].

In vitro prophage identification has disadvantages, e.g. specialised workers and product costs, when used in a larger and more detailed analysis. Computational tools that can be used in prophage prediction, like PHAST [82] (recent version is PHASTER [83]), PhiSpy [84], Prophage hunter [85] are available. These tools help scientists avoid unnecessary steps in their research, such as processing DNA sequences, cutting costs, time and redirecting attention for investigation improvement.

Nevertheless, the tools available at one given moment are only as good as the information available , therefore, with the increasing knowledge availability in databases and the growth of data complexity, an update to tools that can analyse, and correlate data is of great need. Furthermore, new analyses and new approaches that can go along with the advances of experimental tools to answer same or new problems create a vacancy for new research, new work, and new tools.

2.2 COMPUTATIONAL TOOLS

2.2.1 AUXILIARY TOOL GROUPS

The task of creating a main tool capable of predicting prophages integrated in bacteria genetic material is not a linear task and several auxiliary tools have to be put together to evaluate the input. The number of tools and the complexity depends on the main tools' pipeline and, auxiliary tools can be grouped based on their objective.

Homology search group

This group includes tools used to search sequences in databases that could be related to the input sequence. The main tools are BLAST [86], RAPSearch [87] and DIAMOND [88].

BLAST

BLAST or Basic Local Alignment Search Tool is a widely used bioinformatics tool that performs sequence alignment, e.g. nucleotide sequences to a nucleotide database (BLASTn), translated nucleotide sequences to a protein database (BLASTx), amino acid sequences to translated nucleotide databases (tBLASTn), amino acid sequences in protein databases (protein BLAST) and others [89]. This tool aligns sequences scoring matches, mismatches, gaps, and content percentage. Commonly, used matrices include PAM-30, PAM-70, BLOSUM-45, and BLOSUM-62. The program will return similar sequences and the score, e.g., *e-value and bit score*.

RAPSearch

RAPSearch or Reduced Alphabet based Protein similarity Search and the improved version, RAPSearch2, are tools used to search homology between short reads. A key feature presented by this program is speed. The program sacrifices sensitivity for speed, approximately, 100-fold faster than BLAST. This is achieved by reducing sequence length, translating the DNA or RNA sequence to a protein sequence and reducing search alphabet redundancy. The reduction of the alphabet is only made on chemically and structurally similar amino acids, e.g. isoleucine and leucine, that, when replaced, do not affect the final structure or function and, consequently, the result. Furthermore, the program uses a seed extension approach, finding the maximum number of highly similar seeds (subsequence) before evaluating and extending them.

DIAMOND

DIAMOND or double index alignment of next-generation sequencing data is a tool used to search homology between sequences. This software claims to be 20,000 times faster than BLAST with the same level of sensivity (99%). This feature is made possible with better memory bandwith management. Concretly, DIAMOND also uses the seed and extension aproach, where similar subsequences are searched and their localisation stored in an index. DIAMOND uses a doble indexing, sorted lexicographically and traversed together, contrary to BLAST. Furthermore DIAMOND uses longer seeds characterised by weigth and shape. These two parameters are used to describe the number and layout of seed positions. DIAMOND also uses the a reduced amino acid alphabet, similar to RAPSearch.

HMMER Tools

Usually, sequences are identified using alignment tools like BLAST, aligning sequences against databases, identifying it, or finding their homologous sequences. This method proved efficient and fast, but difficulties arose when aligning distant sequences. Thus, profile searches emerged. This approach considers positional information and can calculate relations between sequences. One of the profiling types are Hidden Markov Models. This algorithm can find patterns in sequences, analysing each position as a state and calculating the next step. One example of their use is the analysis of proteins to determine their function and structure or finding homologous sequences. Furthermore, it can be used as an evolutionary algorithm to predict sequence and organism evolutionary behaviour [89].

HMM tools, available at (http://hmmer.org/) [90], can be used to analyse proteins from genomic data, including tools such as pHMMer, HMMsearch, HMMscan and jackHMMer. PHMMer searches for protein sequences in proteins databases and allows gaps in the sequences. HMMscan searches Pfam for protein sequences. HMMsearch searches for

profiles in a proteins sequence database. Lastly, jackHMMer searches protein databases for profiles, protein sequences or multiple sequence alignment. All algorithms use thresholds to select proteins of interest.

In metagenomic data, a HMM phage profile was created using annotated phage proteins from RefSeq. The data was then clustered and curated, and the database vFAM was assembled.

Gene finding tool group

As the name suggests, this group includes tools for finding genes and translation initiation sites. This analysis is done when the number of genes is used to evaluate the input genome or when it is pertinent to analyse and annotate the functional genes in the genome to search their sequences in databases. Several studies have been with tools in this group [91–94] and their results suggest that there are three main tools, GLIMMER [95], GeneMark or GeneMarks [96] and Prodigal [97].

GLIMMER builds an index for the whole genome and then differentiates coding from non-coding genes using interpolated Markov models, whereas GeneMark analyses the FASTA file using an inhomogeneous Markov chain model depending on the sequence type. Prodigal runs differently by scanning all the sequence for stop and start codons, scanning ORFS and saving all guanines and cytosines in each codon, building a frame of model bias to score the start in ORF length and GC codon position, connecting all nodes. A log table, created from 6-mer subsequences, is used to score each gene. Each gene is posteriorly connected with nodes and the genes with a positive score are chosen.

tRNA SEARCH TOOLS

Transfer RNA sequences have become an interesting topic in phage location, due to their importance, in phage insertion and protein expression [98, 99]. This led to the necessity

of development and implementation of tools that could predict where tRNA are located. Between all the existing tools, tRNAscan and Aragorn have been used the most times in phage locating tools.

tRNAscan-SE [100] is a tool to search for tRNA genes in genomic sequences, and is highly used in prophage finding tools, due to tRNA being one of the markers for the boundaries of prophage. This tool accepts a DNA or RNA sequence in FASTA format and uses the tRNAscan and the Pavesi algorithm to find putative tRNA genes. The results of both tools are merged. The putative tRNAs are then analysed by the *covels* algorithm [101], a covariance model originated from the alignment of tRNA. If the score is above twenty bits, the zone is checked for pseudogenes and secondary structures, followed by anticodon and intron analyses, and the result of the discovered tRNA genes is outputted.

Aragorn [102] implements a heuristic algorithm that will search a sequence for certain patterns. More concretely, this sequence will be parsed to find possible loops in the DNA sequence that can represent tRNA genes. These loops are constituted by four stem regions (A, T, D, C) and four loop regions (T, D, C, V). Aragorn searches for loops by locating a specific subsequence corresponding to the T-loop and to the T-stem. Starting from the T-stem, the program will try to locate a specific motif corresponding to the A box, followed by the D-stem and D-loop. Between the D-stem and the T-stem, the A-stem is formed. The V-loop surges upstream the T-stem. Finally, the C-stem and C-loop appear when a specific pattern is found between the D-stem and T-stem. All the possible loops are evaluated with BRUCE program [103], among other criteria.

Databases

Databases are a crucial support for every tool and pipeline. They store all the prevoous information produced and allow constant creation and knowledge improvement. Additionally, a key feature of databases is data comparison, i.e., redundancy and incorrect data

are not included in future works, removing possible errors that could arise from altered results.

Depending on the type of data required, there are numerous databases, some connected to others and some just storing a particular subject data. In biology and in phages, several databases can provide reliable data.

Pfam

Pfam [104] is a database created for protein families and their HMM profiles. Each entry in the database is analysed by HMMer tools, which build a HMMer profile. Then the sequences are queried against Pfamseq, curated and their HMMer profile is identified. Pfamseq was built with only reference protein sequences from UniprotKB [105] to improve speed and resource management, due to the data's growth (size and complexity). In the database, entries are stored by different types like family and domain, being these the most representative. The Evolutionary Classification of Protein Domains (ECOD) [106] is used to group entries in clusters with evolutionary relationships. This database is available at https://pfam.xfam.org/.

NCBI database

Created in 1988, the National Centre of Biotechnology Information (NCBI) is a biological database that gathered and maintain previously created databases, i.e., GenBank has become an excellent support for the scientific community. NCBI harbours various online resources, ranging from data repositories in literature, health, genomes, genes, proteins, and chemicals to online biological data analysing tools, like BLAST. One of the most important features is the close relation with the European Nucleotide Archive (ENA) and DNA Database BANK of Japan (DDBJ) and the scientific community, harbouring new knowledge, while sharing it world-wide and keeping researchers up to date with the work performed in other institutions. Additionally, NCBI provides the Entrez information

retrieving system Application Programming Interface (API). This tool allows the user to retrieve information from the database, parsing all the information and outputs [107]. This database is available at https://www.ncbi.nlm.nih.gov/.

PVOG

Prokaryotic Virus Orthologous Groups (pVOGs) [108] is a database that provides annotations of viral proteins, gene identification, and phylogenetic analyses. It was constructed based on entries from GenBank and RefSeq. Records were manually curated to eliminate non-relevant data. The first draft offered 2912 virus, 77 archeal virus and indeterminate virus. pVOG uses GeneMarks [96], a tool that uses heuristic methods (Markov Models) for predicting the beginning of genes in bacterial genomes (predicting 10393 genes), to improve data quality. Orthologous gene clustering was performed by masking the most' distant' genes and blasting them all-against-all. The new dataset (pVOGs) was compared to the old dataset (POG) to reduce redundancy. Old entries were replaced with new RefSeq references and new and more accurate protein annotations, for instance, gene start and stop.

This dataset was made available in 2016 and updated several times. All information are available in http://dmk-brain.ecn.uiowa.edu/pVOGs/.

ACLAME

ACLAME or A CLAssification of Mobile genetic Elements was established in 2003 and has studied and gathered curated information of agents that can move genes between organisms or inside the same organism [109, 110]. ACLAME uses an HMM pipeline to process proteins based in similarity with other proteins and finds their family and characteristics, thus classifying mobile genetic elements (MGEs). This pipeline uses a Markov clustering algorithm to produce families represented with a set of characteristics. A gene ontology

algorithm is used to perform functional annotation to further improve proteins recall. More information about ACLAME can be found at http://aclame.ulb.ac.be/Classification/description.html .

Sequence clustering tools

Biological data clustering is a method that has become important with the appearance of databases and growing quantities of data. One feature of this method is sequence comparison. This creates the possibility of establishing 'connections' between sequences with domain decomposition, sequence comparison manner, sequence similarity measure, cut-off threshold and transitivity [111]. Two examples of these tools are CD-HIT [112] and DBSCAN [113]. CD-Hit is a clustering program that uses a greedy algorithm, i.e., an algorithm that usually does not reach the best global solution, but quickly obtains a good local solution. The tools that use this program are fast and efficient when analysing a large set of sequences. CD-Hit starts by organising the sequences by decreasing size order, where the first sequence represents the first cluster. Then, the program compares the next sequence with the existing cluster and, if there is a similarity above the established threshold, the sequence is associated with that cluster. If the similarity does not reach a predetermined threshold, a new cluster is formed, represented by that sequence. This comparison is performed until there are no sequences left. The principle is that if sequences have a number of equal peptides, there is similarity between sequences. The filtration mechanism is based in words. There are three types of clustering namely partition-based clustering, hierarchical clustering, and density-based clustering.

DBSCAN [113] or density-based spatial clustering of applications with noise belong to the last one [114]. This algorithm works by defining a point as the centre of a circle and associating all the points inside the area of the circle to that cluster. This algorithm only requires a minimum number of circles that the algorithm will create and the radius of the

circles (epsilon). By establishing a defined radius, this algorithm allows outlier (noise) identification by isolating points that do not belong to any of the clusters, contrary to other algorithms that would be more malleable when establishing the desired cluster.

2.3 MACHINE LEARNING

Machine Learning (ML) [115] is a field of artificial intelligence, representing a different software design approach. Traditionally, programming uses rules and code to form an output, whereas machine learning uses algorithms and data to create the program. This method is used in bioinformatics in various fields, e.g. evolution, genomics and proteomics. The exciting features of machine learning are data handling and processing complex and variate data in large quantities. For instance, in phylogenetics, machine learning can be used to construct phylogenetic trees; in genomics, it can be used in gene finding or motif identification; and proteomics to predict protein function and structure. In general terms, machine learning suits a function to a task as results can be extrapolated from this function without knowing the result *a priori*. Hence, certain concepts need to be considered to understand this field.

Concepts

- Input: Raw data that represent the reality of the problem. Each input is divided in samples or instances that should be independent of each other, in order to not create learning problems. Features describe each sample.
- Feature:Observation made to each sample to explain the problem and can be continuous or categorical. Furthermore, each feature can be independent or dependent. If a feature is independent of other features, it is a good feature to describe the samples and the problem. If a feature is dependent on other features,

normally it is the output or the feature that the other features influence and predict. Good variable need high variance and low correlation among each other.

- Dataset: Data gathered in a database table or matrix, where in the row are samples or instances and in the columns are the features. Normally one of the columns represents the output, the target to predict. The dataset can be divided into training and test datasets for the two machine learning phases, training and testing. Alternatively, the dataset can be divided in several sub datasets where all the data is used to train and test sequentially.

- Training phase: Process of learning is performed. The algorithm parameters are adjusted to learn all the data patterns and relations, to produce the respective output. The result of this process is a model.

- Model: Explains the relation between the data and the output. If a model is successfully trained, it can be used in data outside the data used in the training phase.

- Testing phase: Part of the dataset not used in the training phase is used to evaluate how successful the training phase.

Figure 2 shows an example of a standard workflow for supervised machine learning.

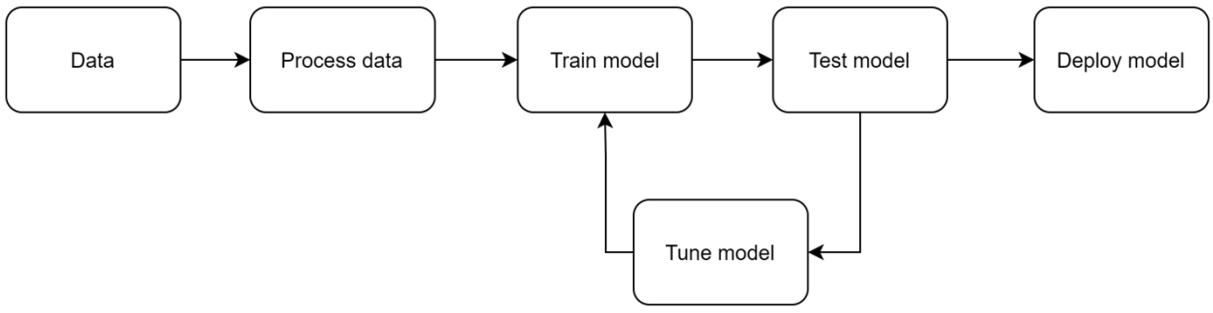


Figure 2: Example of a machine learning workflow for model development. In this case the workflow is associated with a supervised learning algorithm.

Pre-processing

Pre-processing is a crucial step for good quality data. Working with bad data can lead to training issues and create a poorly trained model in testing and deployment. Various steps can be performed to avoid such problems.

The first step when analysing raw data is to handle missing data, which can be classified as 'Missing Completely At Random' (MCAR), 'Missing at Random' (MAR), or 'Missing Not At Random' (MNAR) [116]. The first problem is when values are missing randomly without relation. Such a problem is overcome by removing or substituting the missing value by an extrapolated value, or replacing it with an average value, i.e. mean, median or mode of that feature (column). Removing the row involves the loss of a hole sample which is not the best solution, for small datasets. Data replacement, despite not having lost information, can lead to an overepresentation of unreal values. Missing values dependent on other features are another problem. In this case the problem can be solved by dropping the column or by aggregating features. The last problem is acquiring data meaning that data extraction needs to be revised and corrected. Other problems are duplicated and inconsistent values, which are associated with human errors, either by removing the row or correcting the value.

The second step is encoding categorical data, which is an optional step, depending on the used machine learning algorithm. For instance logistic regression or support vector machines, cannot recognise and use non numerical data. Hence, encoding consists on converting categorical data to a unique value. Categorical values can be divided into ordinal (with inherent order) and nominal (without inherent order). Ordinal data, even when encoded must keep the order between the categorical value to preserve all information thus, not every encoding method works well. Encoding methods, like one-hot encoding, label encoding, ordinal encoding, binary encoding, have specific methods of maintaining data information and relations. For example, one-hot encoding creates the same number of different classes of categorical data, and vectors and encodes the value of '1' to the vector of the respective class and '0' to the other vector. On the other hand, to

preserve class relations, label encoding creates a vector with crescent values to simulate order between classes.

The third step is feature scaling, which rescales all features from the original scale to a finite scale, usually in the ranges of [0,1] or [-1,1] using scalers such as Min-Max scaler, Standard scaler, Max abs scaler, Robust scaler, quantile transformer scaler. This step is crucial to avoid features having larger weights in the learning process, such as distance-based algorithms and is performed by rescaling features' values from the original scale to a finite scale, normally [0,1] or [-1,1]. A short description of the scalers below.

- Min-max scaling: Transforms feature values by using maximum e minimum values.
- Standard scaler: Transforms feature values using mean and standard deviation.
- Max abs scaler: Transforms feature values using maximum absolute value.
- Robust scaler: Transforms feature values using quantile range.
- Quantile Transformer Scaler: Transforms feature values using quantiles information.

Feature selection

As mentioned before, features are used to describe the distribution of the population in the problem. Nevertheless, not all features are good descriptors. Bad features are irrelevant to the problem or redundant, leading to overfitting and high model complexity without improving the results, increasing computational costs and time consumed. Various feature selection methods have been implemented to resolve possible problems that originated from bad feature. These methods balance the dataset reduction with the model performance and can be categorized into three groups [117]:

- Filter method: This method uses a value threshold to exclude columns with lower variance than the threshold value, with only, considering column variance without

creating connections to other columns or the output. Other filter types use the correlation between columns to exclude one of those with a higher correlation (near 1), meaning that having both columns will not bring any relevant information and excluding it will not affect the result [118].

- Wrapper methods: These methods try to find which the best combination of features and can be implemented in various ways depending on the technique used to remove features.
  - The first technique is backward elimination where the model with all the data is inserted into the algorithm and, iteratively, columns (features) will be removed, and the model's performance will be evaluated. If the performance of the model is altered, the column will not be removed.
  - The second technique is Recursive Feature Elimination, consisting on a start model with a determined number of features that will be fitted to the data, recursively, resulting only in the best features.
  - The third technique is forward search where the algorithm will start with a model with one feature, adding subsequent features at a time and evaluating the model's performance. This type of method is greedy and will give the best model with the best features but is more time and computational costly [119].
- Embedded Methods: These methods are included in machine learning models, where during the training process, the training algorithm will store the best features for posterior iterations. LassoCV and tree-based models are examples of this type of method [120].

Splitting data

After ensuring the good quality data and relevant to the problem, the data needs to be split into training and testing dataset to validate the model. The division needs to

ensure enough data for training to avoid over and underepresentation between outcomes. Hence, several methods have been developed to divide the data [121].

- Holdout: Simple division of the dataset in training and testing set, normally using most of the data to train and the rest for testing. This method has the problem of increasing model bias due to not using all the data to train and test, leading to under representing data.

- Cross-validation: This method divides the dataset into subsets without replacement, using it for training and testing until all data is used. Thus, the prediction error will be calculated with all data, instead of only using one part to reduce classification bias. Examples include leave-one-out, leave-out-k cross-validation, stratified K-Fold Cross-Validation and Leave-P-Out Cross-Validation. The first leaves an almost unbiased model but has high variance that can lead to unreliable estimates. The leave out k cross-validation divides data into k subsets using one subset to test and the rest for training. The stratified k-fold cross-validation is used to improve models with imbalanced datasets as this technique ensures, approximately, the number of samples in each class. The last method leave-P-Out Cross-Validation leaves a determined number of samples out instead of a proportion, having a lower bias and requiring high computational power [122].

- Bootstrapping: Contrary to cross-validation, bootstrapping reuses samples to estimate a population statistics and create new sample population maintaining the sample distribution. Bootstrapping takes a single sample and uses it to estimate the population and their features recursively until the desired size is achieved. There are three bootstrapping methods nonparametric, semiparametric and parametric [123].

Learning methods

The learning process is the most crucial step in the machine learning workflow. This step dictates how an algorithm recognizes patterns in data and creates the model used to classify wanted output. Thus, the choice of the learning algorithm, which can describe all the data and be extrapolated to unseen data, is an important task. To this end, learning algorithms can be divided into three major types:

- Supervised learning, where the input data is split in training and testing data and have known labels associated, allowing to train the model with the training data and the label data, build a model and then test the model, by giving the model the testing data without the labels (predicting results) and then comparing the results with the true test labels. Additionally, considering the data and problem premises, two types of algorithms can be used, regression or classification algorithms.
  - Regression considers that the output (Y) for a determined input data is continuous. This type of supervised learning is associated with an error function, e.g. Mean Squared Error (MSE) or Root Mean Squared Error (RMSE), due to continuous results, where each point has variant differences compared to the True results. The main objective for these models, is to have a robust model that can generate outputs for variate data with the smallest error. Examples of regression algorithms are linear regression, support vector regression or logistic regression.
  - Classification algorithms have known data labels associated and classify the output with a finite and determinate value. These models are evaluated with metrics, e.g. accuracy or recall, trying always to maximise these values, with a model that can produce results from the broadest data possible. Examples of classification algorithms are support vector machines, random forest, and Naive Bayes.

- Unsupervised learning is not performed by splitting the data into train and test and the input data does not have labels. Depending on the algorithm used, patterns are found in data and results are drawn from it. These algorithms are used in clustering problems, (e.g. k-means) and dimensionality reduction (e.g. principal component analysis.

- Reinforced learning uses different concepts to create a model. In this case, an agent, the entity that acts, has a state in an environment, in which it will act. The action performed by the agent will change the environment, which will affect the state and produce a reward, that can be negative or positive. The perception that the agent has of the current state is called mapping. If the agent analyses the next environment according to the possible actions he could take, this action is called policy. The function that the agent will try to improve has the best reward according to the current state. In model-based reinforced learning, the agent makes an action based on previous information, using a model created with previous actions and consequences (changes in the environment and rewards). If the action only considers the current state and reward, it is called a free-model reinforced learning [124]. Furthermore, two types of reinforced leaning may be considered, positive and negative. The first promotes performance maximisation, while the other identifies which steps are detrimental to the performance.

Based on the existing literature, supervised learning algorithm are the most used for phage data prediction [125]. The most common machine learning classification algorithms are support vector machines (SVMs) and random forests (RF) [126].

2.3.1 Support Vector Machine

Support Vector Machines (SVMs) create a high-dimensional map with the data features, and iteratively try to maximise the distance between the closest data points. SVMs

separated into linear SVMs, i.e. hard-margin and soft margin and non-linear SVMs, which can compute a new feature space via the use of kernel funtions. Hard-margin SVMs can separate only linear sperable data, while the soft margin adjusts to misclassifications by changing the margin positions. Regarding Kernel functions, for instance, the polynomial kernel tries to adjust the hyperplane to the data with a polynomial transformation, while the radial basis function uses the distance from the points to the centre to adjust the line and improve classification [127]. The following figure 3 is a schematic representation of SVM.

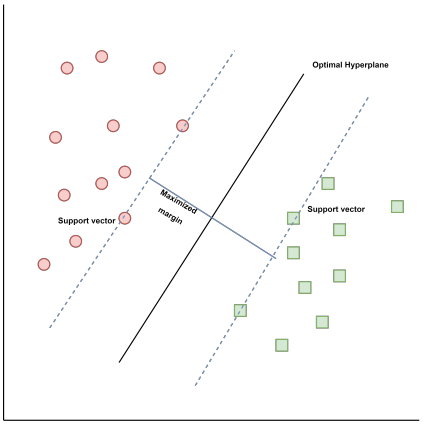


Figure 3: SVM schematic representation, showing the separation of two data classes by maximized margins and a optimal hyperplane. Adapted from [128].

2.3.2 Decision Trees

Decisions trees 4 are decision structures that separate data features through various nodes where each split is evaluated for the cost to the accuracy keeping the lowest cost in the split. Hence, the solution is 'greedy' and not the best overall, leading to a specialisation of the tree to predict one characteristic. There are several methods to improve trees, e.g. stopping splitting features to avoid overfitting or pruning by removing branches that have low importance. The best split occurs when the maximal difference between branches is achieved. One example of the use of decision trees is the random forest algorithm that votes the best decision tree to classify a specific characteristic

maintaining a low correlation between decision trees but a strong capability to predict various data types when in a group.

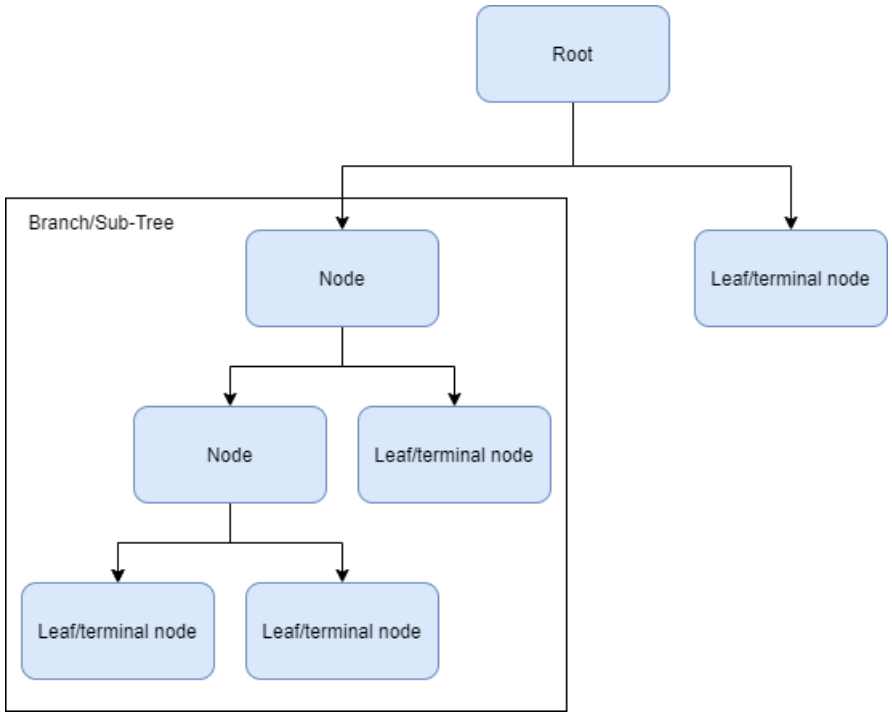


Figure 4: Representation of a decision tree. In this scheme, there are three distinct types of nodes. The root node, has no 'parental' nodes and two 'child' nodes. The other type of nodes are nodes with child nodes and parental nodes that can be called decision nodes and the terminal nodes that only have parental nodes.Adapted from [129].

2.3.3 *Naive Bayes* ALGORITHM

*T*he Naive Bayes algorithm assume the conditional independence of the features according to each class with each feature having its weight in the classification [130]. This assumption is using both the *Bayes* theorem and the *Naïve* assumption. The first calculates the probability of an event happening with the occurrence of a previous event. The second *Naïve* assumption is that each feature is independent of the others.

From these assumptions, there are several applications such as, the Gaussian *Naive Bayes* classifier, where the continuous values are assumed to have a normal distribution,

multinomial *Naive Bayes*, where values have a binomial distribution and *Bernoulli Naive Bayes*, where features are independent Boolean inputs.

2.3.4 K-Nearest Neighbors

The k-Nearest Neighbors algorithm can be used in regression or classification problems. This type of learning algorithm receives the data and tries to classify dependent variable, based on the distance from the point to the near neighbours, using distance functions like Euclidean (direct distance between two points) or Manhattan (absolute sum of the difference between two points).

2.3.5 Gradient Boosting

Gradient Boosting Algorithm uses various learning algorithms to increase the model's robustness, suiting the best learning algorithm to the input data. This algorithm can be compared to the random forest algorithm, where specifically trained trees are used to classify the results, while the gradient boosting algorithm uses different learning algorithms to classify data better.

2.3.6 Multilayer Perceptrons

Multilayer Perceptrons (MLP), Convolutional Neural Networks, and Recurrent Neural Networks are Artificial Neural Networks. These algorithms work as the brain by mimicking the behaviour of passing information between layers of neurons leading to the creation and specialization of several neuron architectures best suited to different problems.

An MLP network consists of an input layer, a hidden layer(s) and an output layer and each layer is composed of neurons, that hold or process information, as shown in figure 5. The input layer has neurons representing the problems information and ideally has the mapping of all the data. The hidden layer is composed of several neurons specialised in making a specific prediction of output by receiving information from the anterior layer and returning a response to the posterior layer. The output layer is responsible for outputting the predicted values. Besides the layers and neurons, weights and bias are important factors in a neural network. These parameters influence the activation of neurons between layers. The activation process consists of adding all weighted inputs to a neuron and using an activation function (or transfer function) for calculating the neuron's value. If the value is above the threshold, the that neuron is activated. Both weight and bias will be randomized and changed between layers to adjust to the data and outputs [131].

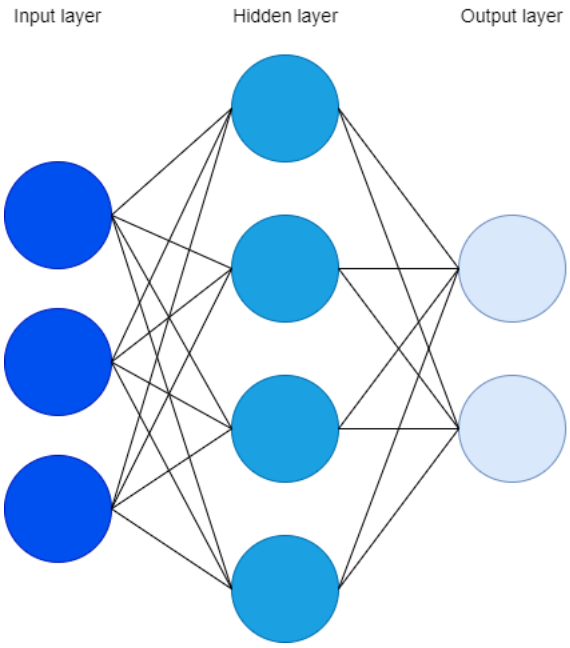


Figure 5: Representation of the MLP supervised algorithm with three layers. Adapted from [132].

2.3.7 Logistic Regression

Logistic regression uses the logistic function on the data to find binary dependencies in the data. If the task is a regression problem, various algorithms can be used. The simplest is linear regression, where the dependent variable (Y) linearly depends on an invariable feature (X). If there is more than one invariable feature, the problem is considered multiple linear regression, otherwise simple linear regression. This algorithm follows the $Y = mx + b$ formula.

2.3.8 Metrics

The testing phase evaluates the training process performance. In this phase, the data that was separated from the training dataset will be predicted using the trained model without using the true labels.

In classification, after prediction, the true labels are compared to the predicted labels and fours scores will be calculated, true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Considering an label/ output X, the true positives are the number of samples with label X which the model correctly predicted true negatives are the number of samples without label X which the model correctly predicted that label; the false positives are the number of samples that the model incorrectly assigned with label X; the false negatives are the number of samples that the model incorrectly assigned other label. These metrics can be displayed in a confusion matrixwhich allows calculating the following scores.

Actual values \ Predicted values	Positive	Negative
Positive	True positive (TN)	False positive (FP)
Negative	False positive (FP)	True negative (TP)

Figure 6: Confusion matrix representation for binary classification . Adapted from [133] .

- Accuracy: Accuracy measures the number of correct predictions in all predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- Recall: Capacity of predicting a label in all the samples that have that label.

$$Recall = \frac{TP}{TP + FN}$$

- Precision: Fraction of correct predictions in all the predictions of that label.

$$Precision = \frac{TP}{TP + FP}$$

- F1 score: Recall and precision are important metrics to evaluate how the model performs but should not be used independently. If recall has the highest value, precision will have the lowest and vice versa. Thus, a trade-off between precision and recall is needed to simultaneously find the best possible value to both metrics. F1 score proposes a harmonic mean between the two metrics, not to be sensitive to extremly large values.

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- ROC/AUC curve: The Receiver operating characteristics (ROC) evaluates the relation of recall and false-positive rates. Furthermore, the area under the curve

(AUC) parameter can improve the model scoring due to calculating the area under the curve created with ROC.

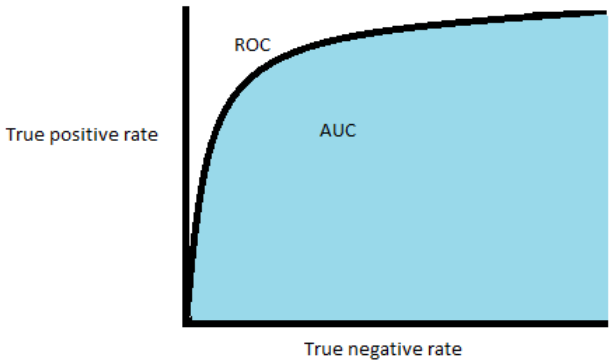


Figure 7: ROC/ AUC representation, where TPR is the true positive rate and FPR is the false positive rate. The area in blue is the area to be maximised. Adapted from [134].

On the other hand, it is not possible to calculate the previous metrics in regression problems. In this case, the model's performance is estimated by calculating the difference error between the true value and the predicted value, using error functions, such as, Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, R squared and adjusted R square.

- Mean Squared Error: Squared difference between points and the regression line.

- Root Mean Squared Error: Standard deviation of the prediction errors. The prediction errors are calculated by the difference between the regression line and the predicted points.

$$RMSE = \frac{\sum(Predicted - Actual)}{Numberofobservations}$$

- Mean Absolute Error: Sum of the difference of every true point to predicted point divided by the number of points.

$$MAE = \frac{\sum(abs(Predicted - Actual))}{Samplesize}$$

- R squared: Measures how much variance does independent variables explain in dependent variables.

$$Rsquared = 1 - \frac{(Explainedvariance)}{Totalvariance}$$

- Adjusted R squared: Modification of R squared with sensibility to the number of predictors in the model. This metric considers the number of predictors that the solution has, increasing the score when the predictor has better results than predicting by change and decreasing the score when the predictor has worst results than predicting by chance.

$$AdjustedRsquared = 1 - \frac{(1 - Rsquared)(Samplesize - 1)}{(Totalsamplesize - Numberofpredictors - 1)}$$

2.3.9 HYPERPARAMETERS

An important part of improving a model is choosing the right algorithm hyperparameters for the data. These parameters will vary with the machine learning algorithm and the data inputted and changed before training, because they will change the way the algorithm learns. Thus, various methods are available to optimise hyperparameter tunning.

- Grid Search: Manually and exhaustive hyperparameter search. This method combines every parameter inputted and trains a model for each combination, returning the best model with the highest scoring metric.
- Random Search: Same as grid search but does not performs an exhaustive search. Instead, randomly tries combinations of hyperparameters.
- Bayesian Optimization: Bayesian optimization uses Bayesian reasoning to find the best hyperparameters. Each iteration, this method recognizes past change estimations to choose the next hyperparameters until reaching the maximal iteration number.
- Gradient-Based Optimization: Uses second derivatives of the training criterion to derive a formula to hyper parameter optimization
- Evolutionary Optimization: Creates a population of hyperparameters distributed in tuples, evaluates each tuple score (fitness function), ranks the hyperparameters based on the score and replaces the worst tuples with changed tuples. Iterates these steps until a satisfactory result is obtained.

These types of hyperparameter techniques are used along with different splitting data techniques to minimize the errors when choosing the best models and hyperparameters.

2.4 PROPHAGE FINDING TOOLS

2.4.1 PHAST

Phage search tool or PHAST, is a tool created to analyse bacterial genomic sequences in the search for prophages. This tool had advantages over the existing tools at the time of development, as it was web implemented, faster, more precise and sensitive, and hence, a broadly used tool in the research community. It uses various programs, such as, GLIMMER [95], BLAST, ARAGORN, tRNAscan-SE and DBSCAN [135] to analyse the sequence and predict the phage location.

As the advantages described before can show, this tool implemented different features in their workflow to improve the final prediction result, like other useful information. The first step was creating a phage and bacterial database with NCBI sequences and other non-identified databases to fasten the search and BLAST time for identifying putative proteins. The first step towards prophage prediction is the genome annotation and comparison. Here, the program takes as input a DNA sequence in FASTA format or annotated GenBank genomes. If the input is a raw DNA sequence, the program will execute certain additional steps to characterise the input. There is ORF identification with the help of GLIMMER. Then, the annotation of translated ORF is performed by BLAST against the PHAST database. After the annotation, the program will locate tRNA and tmRNA in the sequence with tRNAscan-SE and ARAGORN. Finally, BLAST between protein sequences and the phage database is performed. Only e-value scores beneath 10^{-4} are saved for posterior analysis. The second step is the identification of prophage regions and integrity prediction. Here density-based spatial clustering of applications with noise (DBSCAN) is used to form and analyse clusters of sequences, using the size of the cluster and the maximal spatial distance between two neighbour genes in the same cluster as metrics in consideration. On the other hand, an additional task will be performed if the input is a GenBank sequence. The program will perform a search for

phage keywords like *integrase, protease, tail fibre*. If six or more of these protein are found the region is considered a putative prophage. If an *integrase* is found, the adjacent regions are scanned for attachment sites.

There are various outputs for this web tool that can be downloaded to posterior analysis and are graphs and images, that can be explored in the website, that allow the user to explore the prophage regions in the bacterial genome.

2.4.2 PHISPY

PhiSpy introduced a new way to analyse and predict prophages in bacterial genomes. It calculates various characteristics based on the composition of the sequence, 'word' abundance and uses Random Forest as the classification algorithm.

The first step to build a valid classification algorithm was data collection. This step resumes as getting information from PHANTOME server (http://www.phantome.org) of bacterial genome and phage genomes and, then calculating various sequence metrics. The sliding window algorithm was used, considering a forty gene window. The first parameter calculated was the skew of AT and GC. These were calculated by modifying the accumulative calculus of A, C, G and T, for each consecutive gene. This version compensates for local composition differences. The second parameter was the differences in the median protein's length by subtracting the median of all proteins in each window by the median of all proteins in the bacterial genome. The third parameter was computing the orientation of the transcription strand. The genes are divided by their orientation in each window, calculating the maximum window of consecutive genes. The fourth parameter was measuring the abundance of phage 'words'. These 'words' can be described as a group of 12 consecutive base pairs, which allow constructing a library of bacterial, phage and unique (only present in bacteria or phage) words. Then

Shannons's index [136], an index used to calculate species diversity, and the frequency of presence of phage words are calculated. The final parameter was the percentage of homology between the genes in the search window and the phage. If at least ten genes are homologous to the phage genes, the region is considered prophage, otherwise is considered a bacteria. The second step was training and validating the classification algorithm. The training set consisted of five parameters of non-related genomes. If the algorithm found similarities in the genomes parameters, these genomes were considered related. Otherwise, they were considered distant. Then, R [137], a project for statistical analysis (https://www.r-project.org/), was used to rank each window and suggest if the window is phage or bacteria. The third step, is to characterise if the gene is a prophage gene or not. If the gene is in more than half of the search windows, it is considered a phage gene. The last step is defining the *att* sites in predicted prophages. The search zone is extended to 2000 bp beyond the border of the putative prophage region to find the insertion site. If the search finds *intregrase* or *recombinase* or tRNA/tmRNA genes or both, the region is expanded. Otherwise, the initial region is considered. After finding the *attL* e *attR* sites, the proteins in the region are associated with phages subsystems and, if a quarter of the proteins in that region belong to the phage subsystem, the final prevision is the initial prevision.

Finally, a region can be considered prophage, if more than five proteins are related to a phage subsystem (or the region has an unknown protein representing more than half of the proteins in the window).

The output of this tool is the classification of each region in the bacteria genome. This tool also calculates true positives, false positives, false negatives.

2.4.3 PHASTER

The PHASTER tool is an update to the tool PHAST, described above, which made the tool more efficient, fast and precise. Addictionally, it was implemented with various changes to the design and how the output can be explored. Besides all of the following alterations, only the core of the sequence analysis was adjusted.

The first update focused on the database. The tool had a self-updating process, where the new output results are saved in the database, having more data to compare in the future. Hence, the database started growing and becoming slower, having to be optimized.

1. CD-Hit [112] was implemented for database curation and reducing redundancy in bacterial genomes;
2. Update to clustering parameters optimization;
3. Upgrading BLAST to a newer version with faster and increased performance;
4. Upgrade to the partition query sequences;
5. Update to the bacterial database;
6. Better servers;
7. Quick check in the database for the query sequence added, avoiding processing sequences already analysed.

This allowed the tool to decrease queries' running time in both the bacterial database and the viral database.

The output of this tool is also more interactive and allows to download content for posterior analysis.

2.4.4 MARVEL

Marvel [138] is a tool that predicts of prophage in bacteria stem from metagenomic sequences and uses composition features and a random forest classifier.

The basic dataset was created starting the microbial RefSeq dataset, specifically bacterial genomes and dsDNA virus from *Caudovirales* order. Phages with tail were selected as the representative group and constitute the major Ambiental sample. The dataset was divided into ”before” and ”after” 2016, suggesting that the division simulates the use of different tools in recent isolated sequences. The training dataset (before 2016) had a larger number of genome samples than the testing dataset (after 2016). Both datasets were then processed to generate contigs with specific lengths. For each analysed fragment, ten contigs were created with determined lengths, that may or may not have overlap. Bins were created from contigs originated from contigs that originated from the same organism. This process created two datasets to train the classification algorithm. Then the program performs the feature extraction to develop a classifier. The extracted features were:

- DNA k-mer and GC composition;
- The mean length of each gene (total length of predicted CDSs in the genome or contigs in a bin dividing by the total number predicted CDSs);
- The mean spatial distance between genes (mean length of CDSs regions;
- Gene density (number of total CDSs dividing by the length of the genome);
- Frequency of strand change between neighbour genes (sum of the strand shifts number dividing by the number of CDSs genomes);
- Relative ATG frequency (number of ATG triplets in a strand, all the contigs in a bin and the complete genome dividing by the number of 3-mers in the sequence);

- The fraction of significant gene hits in a pVOGs database (each CDS in the genome is search with HMMscan against the database of viral HMM, dividing the number of hits for the total CDS number;

Features that gave small gains were eliminated. The selected features were gene density, strand shifts and the fraction of significant gene hits in a pVOGs database.

The output of this tool is the separation of metagenomic dsDNA phage sequences from bacterial genomes.

2.4.5 PROPHAGE HUNTER

Prophage hunter is a web tool created to predict prophage location and function using a machine learning algorithm, logistic regression, trained by phage genomes retrieved from the NCBI database. Temperate phages with the *integrase* gene. The rest was considered as positive validation. From bacterial hosts, sequence fragments served as negative training and negative validation.

The Prophage Hunter machine learning algorithm is briefly described next. Initially, the genomes are aligned with BLASTn, using an *e-value* of 10^{-5} as a cut-off. Then the prophage genomes are used as the reference dataset, and the genome from the host is considered the negative dataset (the same length was used in phage and genome). Twenty four features are calculated and used in the model, namely protein length, transcriptional orientation, Watson-Crick ratio, transcription strand switch and amino acids composition (one feature per amino acid). The output of this step was the beginning and the end of the prophage, the probability of being active, phylogenetic relation and the protein's functional annotation. The second step is Fuzzy matching. Here, a database is created with annotated phage genomes. Then the CDS are extracted, selected by similarity (above than 75 %). From this database, eleven prophage classes were identified.

InterproScan is then used to search the Pfam database for domain functions, categorise those functions in classes, and create relations between classes and prophage activity. Four of those domain classes were found to be highly correlated with active prophages, namely:

- Assembly;
- Hypothetical protein;
- Infection;
- Unsorted;

These classes are used as initial regions of the prophage and compared to bacterial genomic sequences using BLASTx. The sequences with previous classes are selected for posterior analysis. The last step is to locate boundaries. The selected sequences that do not have an infection, assembly, packing, or *integrase* classes are not included in the posterior analysis, due to these classes being important for prophage activity. The final process is to search for *att* sites twenty Kb upstream and downstream from the remaining sequences remaining. The *att* pair selected to represent the borders, is the one with the better alignment bit score.

The output obtained from this tool is a table with the candidate prophage, start and end of the sequence, the category (active or inactive), the score, the closest phage, the number of genes and downloadable DNA, protein and CDS files.

Table 1: Summary list of tools and their most relevant characteristics. Sensitivity and PPV values were taken from the tool publication article or comparisons made between newer tools and tools already available.

Tool	Classifier	Input	Output	Sensitivity	PPV	References
Phage Finder (2006)	Phage like hits	Information file GenBank file	Information file	67%	94%	[139]
Prophinder (2008)	Phage like hits density	GenBank file	Web service Information file	79%	94%	[140]
PHAST (2011)	Gene density	DNA sequence GenBank file	Web service Information file	85%	94%	[82]
PhySpy (2012)	Machine learning	GenBank file	Information file	94%	99%	[141]
VirSorter (2015)	Viral genes	DNA sequence FASTA file	Web service Information file	75%	84%	[142]
PHASTER (2016)	Gene density	DNA sequence GenBank file	Web service Information file	87%	91%	[83]
VRprofile (2018)	Gene density Gene analysis	DNA sequence	Information file	85%	66%	[143]
Marvel (2018)	Machine learning	FASTA file	Information file	87%	91%	[138]
Prophage Hunter (2019)	Machine learning/clustering	FASTA file GFF file	Information file	99%	-	[85]
ProphET (2019)	Prophage density	FASTA file GFF file	Information file	73%	84%	[144]
Phigaro (2020)	Prophage density	FASTA file	Information file	-	89%	[145]

3

METHODS

3.1 DATA COLLECTION

Creating a robust dataset by gathering and cleaning data, is a fundamental step to explain and discriminate all the problem variables. Here, bacterial sequences constitute the negative dataset and phage's sequences the positive dataset. The first step for the construction of the dataset is gathering available data. The same period of time report by *Song* [85], was used to build the negative dataset as a limitation for extracting of the accession numbers and host from the NBCI virus database [146]. Then, a BLAST search was performed between phages and hosts to determine which bacteria had phage parts and their coordinates. Phage accession numbers were extracted from NBCI virus database. The second step was the manual curation of the BLAST search result file in order to remove unwanted characters and facilitate an automatic parsing, favouring the data processing. The third step consisted of removing the phage's sequences from the bacterial genome. In this step, the Entrez Programming Utilities [147] was used to access bacterial genomes in the NCBI nucleotide database, remove the phage sequence from the bacterial genome, and store the sequences. In the final step the features that will be used to differentiate phage from bacteria were considered. The features were calculated for 10000 base pairs and 5000 base pairs windows throughout the DNA sequences (both curated bacterial and phage's). An additional step was performed to create a synthetic

phage dataset, where phage gene sequences were randomly deleted or inserted and completed with random genetic material from the bacteria. As bacterial genetic material sequences are larger than the phage genetic material sequences, the dataset had more bacteria sequences than phage sequences creating an unbalanced dataset. Each dataset was split into two datasets to ensure model training and optimization, using the dataset with 70% of all data to train and test models and the second dataset (other 30%) for tunning of the algorithm parameters. Finally, an oversampling algorithm (SMOTE) and an undersampling algorithm (RandomUnderSampler) were implemented to reduce the class imbalance.

3.2 FEATURES

A dataset feature can be explained as being a property that can describe the analysed data. In this case, 42 features were calculated based on three major genetic material characteristics.

- Gene density

This group is based on protein density along with the genetic sequence window.

1. The first calculated feature was the number of genes in the same orientation divided by the number of genes found in the window.
2. The second calculated feature was based on the sum of all the genes in the window, divided by the window's total length.
3. The third calculated feature was the number of genes in the negative strand divided by the number of genes in the positive strand.
4. The last calculated feature was the number of changes in the orientation of the transcription.

It is not yet documented why there is a difference between organisms and their transcription but it was proposed that the lack of RNA and DNA polymerase and disruption of terminus might justify the orientation of the phage sequence. [148, 149]. Also, studies have shown a preference between transcription in the positive strand in contrast to the negative strand [150].

- Genetic material composition

This group is based on the composition of the genetic sequence.

1. The first feature is the amino acid percentage, therefore accounting for twenty features (alanine(A), cysteine (C), aspartic acid (D), glutamic acid (E), phenylalanine (F), glycine (G), histidine (H), isoleucine (I), lysine (K), leucine (L), methionine (M), asparagine (N), proline (P), glutamine (Q), arginine (R), serine (S), threonine (T), valine (V), tryptophan (W) and tyrosine (Y)).
2. The second feature is the guanine and cytosine abundance in the sequence.
3. The last feature is skew in the abundance of guanine and cytosine to adenine and thymine.

Several studies have shown that the organism composition varies with the availability of amino acids in the environment and allows classifying organisms [151–153]. A study [154] was conducted in several environments where multiple organisms were analysed, and their relative amino acid composition and GC content calculated, showing that the relative amino acid composition and GC content changed with the environment. Considering that a bacterium is an environment for a phage, these features can represent interesting study methods. Thus, considering that each bacterium can be contemplated as the phage environment, the bacterial amino acid composition will vary the composition of the phage. Furthermore, phage's can be considered mobile genetic elements, which could mean that their host could change in each life cycle, resulting in a composition that can slightly vary, leading to a different composition than the host. In addition to these features, GC skew has also proven to be a feature that can characterise organisms [155].

- Dinucleotide abundance

The last group is the relative dinucleotide abundance. In this category sixteen features are calculated from all possible nucleotide combinations CpC, CpG, CpU, CpA, GpC, GpG, GpU, GpA, UpC, UpG, UpU, UpA, ApC, ApG, ApU and ApA. These features have have potential to be used to differentiate bacteria from phage, because of the heterogeneity between the bacterial genome and the phage sequences caused by the different origin of both organisms [65, 156, 157].

3.3 Data Pre-processing

Building datasets with features obtained from different methods will create values with different orders of magnitude between features. Thus, when using these features, the ones with a higher order of magnitude could influence the model rendering the other useless. Here, several methods were tested to pre-process three datasets (10000 bp dataset, 5000 bp dataset and dataset with alterations):

- Min-max scaler (MinMaxScaler function)
- Robust scaler (RobustScaler function)
- Standard scaler (StandardScaler function)
- Normalizer (Normalizer function)

3.4 MODELS

Several machine learning algorithms could be fitted to the data to precisely classify the desired output. In this work were used:

- Support Vector Machine (SVM) using SVC function;
- Random Forest (RF) using RandomForestClassifier function;
- K — Nearest Neighbors (knn) using KNeighborsClassifier function;
- Gaussian Naive Bayes (GNB) using GaussianNB function;
- Multi-layer Perceptron (MLP) using MLPClassifier function;

3.5 FEATURE SELECTION

Feature selection is an important method to use in a machine learning pipeline to reduce running time, computational burdens and reduce the probability of overfitting. In this work, the implemented method used a variance threshold to remove all columns with a variance lower than 0.5, more specifically, columns that did not increase model performance.

3.6 PERFORMANCE EVALUATION

A performance evaluation was implemented to evaluate how the model predicts the desired output accurately. To this end, several metrics must be applied to the model.

The first models created from 70 % of the data were evaluated with precision, recall, F1 score, and Receiver Operating Characteristic Area Under the Curve (ROC-AUC). A

confusion matrix was also generated to distinguish the model predictions further, either being phage or bacteria.

In the second instance, to further improve the created models, a cross-validation algorithm was implemented with three splits to choose the best parameters for each model. Moreover, a grid search algorithm with a five splits cross-validation, was used to choose the best model with the best parameters using the metrics f-score, ROC-AUC, accuracy, precision, and recall scoring each one of them.

In the third instance, model result predictions were compared to zones of other tools that had already been validated.

3.7 MODEL OPTIMIZATION

Model optimization is used to further improve the created models through a careful analysis of parameters combined to obtain the best model with optimized parameters. Here, different classification algorithms were fitted to create various prediction models, and several parameters were tested and evaluated, as shown in the following table 2.

Table 2: Hyper parameters tested for each algorithm.

Algorithm	Parameter	Values
GNB	N_estimators	100,500,1000
	Learning_rate	1,0.5,0.1,0.01
RF	N_estimators	100,500, 1000
	Min_samples_split	3,5,7
	Criterion	Gini, entropy
SVC	C	0.1,1,10
	Kernel	Linear, rbf, poly, sigmoid
Knn	Leaf_size	3,5,7,9
	N_neighbors	3,5,7
	P	1,2
MLP	hidden_layer_sizes	(50,50,50), (50,100,50), (150,100,50)
	Activation	Tanh, relu
	Solver	Sgd, adam
	Alpha	0.0001,0.05
	Learning_rate	Constant, adaptive

A grid search algorithm was implemented to evaluate each model with the respective combination of parameters. Following the parameter selection, the model was fitted and evaluated with five splits in cross-validation.

Furthermore, five models were created for each algorithm with a Stratified Shuffle Split of the 5000 bp dataset to reduce data bias.

3.8 Protein Annotation

An important step in prophage characterization is protein identification. This step requires searching databases to find the name and molecular function of all proteins inside the putative prophage region. To this end, three tools were implemented.

1. The first implemented tool was DIAMOND [88]. Before the tool can be used, a database needs to be implemented to align the proteins in the putative region with

known proteins. Thus, two databases were applied using the *'makedb'* method. The data used to build this database was from Swissprot [158] identification numbers with the respective sequences in FASTA format. After this step, diamong 'blastp' method can use to align unknown proteins to proteins in the database. These results present themselves with protein, hit, length of the query sequence, the query of the sequence, positions were blasted, e-value and bit score.

2. The second implemented tool was InterproScan [159, 160]. This tool calculates various metrics for each protein sequence, like HMM profile and patterns, to explore multiple databases, like Pfam and PANTHER, and characterise the query sequence. Various outputs can be extracted from this tool depending on the databases that are being used. In this work, the output is the protein identification, the protein domain and molecular function associated with the domain and scores attributed to the comparison between the query and the sequences in the databases. To further improve protein identification, a database from ACLAME [109] was extracted with the help of a request python package to a format with database id, name, function, and sequence. The data was then processed to drop missing or repeated values and convert database functions to functions the activity process evaluation could interpret. Diamond was implemented to compare this database with the inputted proteins.

3. The third tool implemented was BLAST to compare entire phage sequences to the zones found. This method allows the identification of the prophage genomic region.

3.9 BOUNDARY LOCATING

Boundary locating is a step to find more realistic borders of each putative prophage instead of limiting the putative prophage to an artificial step of the search window used in the 10000 bp or 5000 bp division.

This step was implemented with the search of 'truncated' genes. The insertion of the bacteriophage in the bacterial genome is made in certain regions. These regions have been documented on specific regions and few generalizations can be made. One possible generalization is that the bacteriophage inserts itself, within a gene, splitting it into two. Hence, this algorithm extends the search zone in one kb to each side and then using 200 bp window combinations from each side to search for a gene with BLASTp in the Swissprot database. If a gene is located, the gene with the best bit score is selected and the region is added to the original window coordinates. Another implemented method was the tRNAscan-SE program that scans the main sequence and outputs the tRNA coordinates in the sequence. If the tRNA is close to a putative prophage zone, this coordinate will be the new prophage boundary.

3.10 ACTIVITY SCORING ALGORITHM

The dynamics established between bacterial defence against phage infections and the respective phage evasion mechanisms are proven to be complex [77]. While the phage can insert the totality of the genetic material, it can suffer several alterations like insertions or deletions, that could disrupt some vital phage functions. Thus, a scoring algorithm was implemented to evaluate the potential of the prophage to be functional when induced.

The first step to develop an accurate and robust algorithm capable of evaluating prophages possible functionality is to define which proteins each phage requires to have a viable reproductive cycle. Studies have already made progress in this matter by

identifying, for example, what does the smallest phage codify [161] or analyse certain prophage genomic structures that are possible to be induced [151, 162]. Unfortunately, due to the existence of a large variety of phages in nature with different compositions and protein requirements, a structure that could be applied to all prophages cannot be developed. Thus, the general characteristic in common is the phases in the phage's lifecycle. These phases were used to create classes for the proteins based on their functions. In an approach of learning by the available examples, a dataset of phages and their reproductive cycle was downloaded from ACLAME [109] database with the help of *request* python package.

The second step was creating a protein profile to characterize the phage and its lifestyle. All proteins in the CDS regions with products or notes were extracted and stored in a dictionary for each NCBI accession number. These extracted products or notes represent the required proteins/functions to enable the correspondent lifestyle.

The third step is processing the products and notes extracted in the last step to establish be possible comparisons between phage compositions. Thus, this step has two phases: creating a dictionary/language capable of being understood by the computer to attribute a class to the protein and the second the attribution of a class to each protein and creation of rules that can represent a class phage constitution. The first phase required searching for proteins' name and attributing them to a phage lifecycle phase class. A word dictionary from the ACLAME database was created to further improve the classification of each protein by enriching the vocabulary. This process required scrapping of all phage functions described in the database with the associated protein description. Then, manually, all words associated with the function and not in the dictionary, were added. For the second phase, products and notes from the previous step were compared with words and expressions in the dictionary and a class was attributed. Some word relations were implemented for further optimisation. The final process was creating rules that could describe putative phage functions and the phage capacity to have a complete lifecycle.

3.11 GALAXY

Galaxy is a workflow engine that allows the use and integration of bioinformatic tools in a user-friendly manner, available in the galaxy tool shed [163].

The tool's implementation in a structure suitable for galaxy integration was performed by downloading galaxy instance and installing following the documentation for the Windows subsystem https://galaxyproject.org/admin/config/windows/. After the installation of the galaxy instance, Planemo, which is a command-line tool found in https://planemo.readthedocs.io/en/latest/writing_standalone.html, was used to build an Extensible Markup Language (XML) wrapper file for the python implemented tool with the description of the requirements, commands with which the tool was going to function and which file was going to use, inputs and outputs. An additional XML file was created to search and install for packages not included in the galaxy tool shed.

4

DEVELOPMENT

This work was developed in Python 3.7.5 using the integrated development environment Pycharm [164], google colaboratory and dockers. It is possible to divide the workflow into five major parts that are represented bellow.

4.1 DATA COLLECTION

Data collection was made by manually downloading a dataset of phages and their hosts from NCBI virus website. This allowed establishing the first dataset of bacteria names and phages's accession numbers.

As the phages infect and integrate themselves in the hosts genomic sequences, the phage sequences integrated into the bacteria genomic sequences needed must be removed. A *'blast'* script was developed to find the coordinates of the prophages in bacteria. The main function of this script uses *NCBIWWW.qblast* to access BLAST services in NCBI and takes as input the file with the phage accession numbers and the hosts' names. The output of the main function is the coordinates of the phage and the bacteria accession number, writing them in a *.csv* file. The ' [' and '] ' were removed and' ; ' was added in the end to automatise the parsing process. After this step, a function was implemented to find coordinates of phages in the bacteria, writing the result as a dictionary where the keys are the bacteria, and the values are coordinates. Finally, the dictionary

is exported in a .json file. Then, the sequence associated with the bacteria's accession number in the NCBI database was downloaded. The sequences within the coordinates retrieved from the dictionary were saved into a file, together with the information on the DNA's strand (positive or negative strand). Finally, all phage accession numbers in the NCBI virus database were manually downloaded and processed in the feature extraction step to retrieve more phage sequences. Such procedure was used to create the first and second datasets, with and 5000 base pairs windows, respectively. The second dataset was then split in to five divisions and each was used for the models. The third dataset was created with the same bacteria sequences but different phage sequences. In this case, three types of phage samples were used (altered with insertions, deletions and regular). The first type was phage sequences with insertions where some genes were removed and replaced, in the same position, with random bacterial sequences of the same size. The second type was gene deletions in the phage sequence, adding the bacterial sequence with the deleted size. The third type were regular sequences.

4.2 FEATURE EXTRACTION

Feature extraction was performed in two phases, one procedure for extracting features from the bacteria and the other to extract features from phages.

4.2.1 BACTERIA FEATURE EXTRACTION

The feature extraction procedure for bacteria required the result file from the data collection, where the sequences in common with phages and bacteria (BLAST results) were saved. This file was a dictionary structured with the bacterial accession number as keys, and the sequence in common with the phage and a number representing the state of

their position in the bacterial strand (’1’ if not inverted ’ 0’ if inverted) as values. For each bacterium, various sequences having the state 1 or 0 adjacent to it were available. This file was loaded, using two functions that load the sequence from the NCBI nucleotide database. The first function loads the bacterium’s accession number and the second removes the portion of the sequence in the dictionary from the bacterial sequence. This step was performed to remove possible phage ’contamination’s’, as phages integrate the bacterial sequence and may change the bacteria’s overall constitution. After removing of the sequences, an algorithm to search and find the genes and the translation initiation site identification was used. This step was performed because removing of the prophage sequences from the bacteria’s genome could alter gene count and gene positions, influencing features that depend on the gene sequence. This sequence was divided into 10000 bp or 5000 bp fragments using a sliding window algorithm and each window was processed by twelve functions to calculate features.

4.2.2 Prophage feature extraction

The genetic material information was required to calculate the phages’ features. Thus, for each NCBI’s accession number two functions were applied. The first is used to retrieve the DNA sequence and the second to retrieve the positions of the genes, allowing to count the number of genes, their length, and their position in the transcription (inverted or not inverted). Windows were created and analysed with twelve feature functions using a similar sliding window algorithm.

4.2.3 FEATURES

Eigth functions were implemented for transcription orientation, gene density, number of genes in the negative strand versus the number of genes in the positive strand, number of genes in the same transcription, amino acid percentages, GC percentage, GC skew and dinucleotide abundance to calculate all features:

1. Transcription orientation: Calculates the number of genes in the same transcription orientation.
2. Gene density: Calculates the gene density by dividing the number of the genes by the total length of the window.
3. Ratio negative-strand vs positive-strand: Function that divids the number of genes in the negative strand by the number of genes in the positive strand.
4. Transcription orientation change ratio: Calculates the number of changes of transcription, by dividing the number of the genes by the number of changes.
5. Amino acid quantity: calculates the percentage of amino acids in the window. This function creates twenty features, one for each amino acid.
6. GC percentage: calculates the percentage o guanine and cytosine in the window.
7. GC skew: Calculates the difference between guanine, cytosine and adenine, thymine in the window.
8. Dinucleotide abundance: Calculates the percentage of each pair combination of nucleotides in the window.

4.3 MODEL DEVELOPMENT, TESTING AND IMPROVEMENT

All models were developed in a google collaboration notebook, as the testing and improvement of the models, through the steps describe below:

1. The first step was loading the data into the notebook and associate the output column. Twenty percent of all the data was saved into a file as an independent test phase in this step. In a quick analysis to prevent an imbalanced dataset, two methods to under and over sampling were implemented resulting in three datasets (normal, undersample and oversample).

2. The second step was pre-processing the dataset and feature selection resulting in a *'scaler.sav'* file for each type of dataset with the scaler variables saved to use in posterior data. The feature selection provided an array with the same number of columns as the dataset (no feature extracted).

3. The third step consisted of loading the algorithms, building the first models with the three datasets, and creating the scaler. All models were evaluated, and the models were saved using a *joblib* package.

4. The fourth step was independent testing where the data was loaded, prepossessed with the scaler created in step 2 and the results calculated.

The first dataset had an additional step where a pipeline was used with two steps of analysis. The first uses *Gridsearchcv* to find the best parameters to the data and algorithm and the second uses a cross validation to select the best model for the best tested parameters based in scoring metrics. The result is a trained model and scores.

4.4 BOUNDARY LOCATION

Boundary location finding was implemented in the main function *'inser site'* with two auxiliary functions *'diamond blst swiss'* and *'trnascan'* The main function takes as input the start and end coordinate of the window of the putative prophage and an identification number, called *cv*, that is, a number associated with each putative prophage. This function creates 200 bp zones from the 1000 bp upstream and downstream from the putative prophage zone and parse all sequences by joining one window from each side until all combinations are performed, translating, and writing the sequences in a FASTA file. Then the *diamond blst swiss* function uses DIAMOND to align sequences against the Swissprot database. The resulting file contains the genes associated with the zone and each zone has two numbers associated with each window.

The resulting file is parsed and the zone containing genes with more than half of the length of the zone and the higher bit score is selected as the new zone.

Another process of finding the real insertion site of the putative prophage zone is to use the *trnascan* function that calls the tRNAscan-SE program to find tRNAs in the main sequence. After this analysis, if a tRNA is close to the sequence it will become the new boundary of the putative prophage zone.

The final step is adding the extra sequence to each side of the putative prophage zone. Hence, this function returns the number of base pairs that will be added to each side. For instance, zone 'D0-U2', where '0' represents the first window on the downstream side and '2' represents the third window on the upstream side. Thus, in this case, two hundred base pair will be added to the left side and six hundred to the right side of the putative prophage.

4.5 ACTIVITY SCORING

The steps for making an algorithm that calculates the phage's activity score can be divided into four parts.

1. The first part was creating a dataset of phages with their lifestyle to be possible a comparison between them and a prophage. So, a function called *'phage lifestyle'* was created. This function uses *request* package to scrap a data source for the dataset and saves it in a *.csv* file.

2. The second part was creating a *language* that could be interpreted and create the relation between the protein function and a certain bacteriophage lifecycle. So, to create a dictionary, a main function named *'functi simp'* and an auxiliary function named *'prot in phi'* were created. The main function scrapes the ACLAME database for the identification number and the function of the protein. Then uses each identification number to get the description of the protein and associates the function of the protein to its description. The results were the function and description of a protein saved in a *.csv* file.

3. The third step was a manual curation of associating each protein or description to a class in the *language* dictionary and organising the results from the second part to the same language in the dictionary. The selected functions are adsorption, penetration, integration, biosynthesis, assembly, lysis, hypothetical and undetermined.

4. The fourth step was processing each phage and its lifestyle. The main function named 'make perf phage' was created with three auxiliary functions named *'get phage df', 'org struct'* and *'save phage perf'* . The main function uses *'get phage df'* to load the phage data in a dataframe with the NCBI accession number and the phage lifestyle. Then it uses the *'org struct'* function to get the product or note of the CDS and tRNA proteins, which is processed with the *language* dictionary,

resulting in an array with the ratio of all protein in the classes divided by all proteins in the phage. The last function is used to save the results in a .csv file.

Acquiring data from phages that had known and inducible lifestyles allowed creating of the rules implemented in the final step. The final step is the scoring algorithm. Using the results obtained above, enable the creation of several rules that could represent an active phage and their functions or the lack of them and how cryptic the phage was. After comparing the results above and adjusts to zones with several phage proteins, these rules were established to exclude the phage:

Table 3: Description of the reasons used to evaluate the putative prophage zones. If any of the reasons are full filled, the zone being analysed is excluded from the results.

Reason	Description
1	Less than ten proteins.
2	Less than three known phage's proteins.
3	If the composition of the phage is less than 50 % of unknown proteins and there are no known proteins or the composition of the phage is less than 82 % of unknown proteins.
4	If the known composition of the phage is more than 50 % hypothetical and unknown proteins, and integration, assembly and penetration proteins have not been identified.
5	If the composition of the phage is 75 % biosynthesis and integration and undetermined is unavailable.
6	If less than 20 % of the phages are unknown and unrelated with biosynthesis and no integration, assembly and penetration are found.
7	If the composition consists in 90% of biosynthesis, hypothetical and undetermined and it has less than 7 hits in known phage proteins.

These conditions classify the phage's capacity and allow the exclusion of various incomplete phages based on the proteins found in the region.

5

RESULTS

One of the objectives of this work was to create a model that could differentiate phage sequences inserted in a bacterial genome. As this is not a linear problem, dataset 1 (with 10000 bp parts), dataset two (with 5000 bp parts) and dataset 3 (5000 bp with altered phage sequences) were created to represent various features distributions. All datasets were pre-processed, and their features calculated, balanced and used to train models. These models were evaluated, and the results were compared to zones predicted by other tools (Prophage Hunter and PHASTER). Only the zones predicted by both tools were used to evaluate the newly trained models to ensure that these had a high probability of being a prophage.

5.0.1 DATA COLLECTION

The first part of data collection was required the bacteria and phages sequences. Since using all bacteria with prophages would result in a huge dataset that would be very time consuming and would create an even more unbalanced dataset, only 5006 bacteria accession numbers were obtained by aligning 3594 phage's present in the NCBI Virus database (until 2018) to the respective host with BLAST. The bacteriophage dataset was obtained by downloading all phages in the NCBI virus database, with a complete genome (date 10/11/2020), which comprised around 43776 phage accession numbers.

Dataset 1

The negative part of dataset 1 was built with bacterial genomes, from which the phage sequences were removed, and split into 10000 bp windows. For the positive part of dataset 1 the phages were split into 10000 bp windows, resulting in 576983 samples from bacteria sequences and 56253 samples from phage sequences, creating a highly unbalanced dataset. Hence, an oversampling algorithm (SMOTE) and an undersampling algorithm (RandomUnderSampler) were implemented, resulting in the oversampled dataset having 1153966 samples of bacteria and phage and in the undersampled dataset with 56253 bacteria and 56253 phage samples.

Dataset 2

Dataset 2 followed the same procedure performed in dataset 1 but with windows of 5000 bp resulting in 1398656 samples from bacteria sequences and 141577 samples from phage sequences. From the tests performed in dataset 1, the results between undersampling and oversampling did not vary significantly and therefore, to avoid a bigger computational effort and time spent in model development, only undersampling was performed, creating a dataset with 141577 bacteria and 141577 phage samples.

Dataset 3

The negative part of dataset 3 was obtained with the same method as for dataset 2 (with 5000 bp windows). The positive part used 5000 bp windows, where each window was processed two times beyond the normal process. The first process used insertions and the second were deletions, resulting in a dataset with 1398656 samples from bacteria sequences and 536532 samples from phage sequences. Undersampling was implemented too, resulting in a dataset with 536532 bacteria and 536532 phage's

samples. The option to use 5000 bp windows to the third dataset was supported by the fact that dataset 2 provided better results than dataset 1.

5.0.2 Feature analysis

The number and orientation of genes are necessary when creating four of the features. Prodigal was used to predict the localization and orientation of the genes in the bacterial genomes. This method was used as the removing the common sequences between phage and bacteria could lead to the disturbance of gene zones/coordinates. On the other hand, for all phage accession numbers, NCBI nucleotide genome information was used to extract gene localization and orientation.

A feature analysis was performed to evaluate feature distribution between datasets and between phage and bacteria, from the full 42 features. The table 4 exhibits the mean of the features of each dataset to demonstrate how the features vary between dataset, organism (Phage or Bacteria), size of the window and the alterations to the phage sequences and several conclusions could be drawn.

Table 4: Feature values for each dataset. There are three types of values for each bacteria and phage to notice the differences between each variation of the datasets. Furthermore, differences between features can be compared, either between bacteria-phage or D. 1 - Bac - D. 2 - Bac, for example.

Features	D. 1 - Bac	D. 1 - Phage	D. 2 - Bac	D. 2 - Phage	D. 3 - Bac	D. 3 - Phage
Transcription orientation	1.629	4.124	0.988	2.209	0.988	1.76
Protein length	982.626	808.412	994.404	843.639	994.404	222.947
Watson and Crick use ratio	1.664	0.962	1.112	0.499	1.112	0.395
Strand switch in transcription	5.245	13.265	4.15	7.683	4.15	5.914
A	0.078	0.062	0.079	0.062	0.079	0.058
C	0.032	0.028	0.032	0.028	0.032	0.029
D	0.028	0.034	0.027	0.035	0.027	0.033
E	0.027	0.035	0.026	0.035	0.026	0.034
F	0.043	0.039	0.042	0.041	0.042	0.046
G	0.056	0.058	0.055	0.057	0.055	0.052
H	0.028	0.028	0.027	0.028	0.027	0.028
I	0.056	0.052	0.055	0.054	0.055	0.06
K	0.042	0.043	0.042	0.044	0.042	0.048
L	0.086	0.089	0.085	0.091	0.085	0.094
M	0.017	0.017	0.018	0.017	0.018	0.018
N	0.037	0.037	0.036	0.039	0.036	0.042
P	0.059	0.052	0.06	0.051	0.06	0.046
Q	0.034	0.035	0.034	0.035	0.034	0.036
R	0.096	0.091	0.096	0.087	0.096	0.079
S	0.088	0.089	0.092	0.088	0.092	0.087
T	0.054	0.06	0.055	0.06	0.055	0.057
V	0.053	0.058	0.053	0.058	0.053	0.056
W	0.017	0.017	0.018	0.017	0.018	0.017
Y	0.027	0.03	0.026	0.03	0.026	0.033
GC_content	49.243	47.894	49.6	47.377	49.6	44.747
GC_skew	0.0	0.021	0.0	0.018	0.0	0.017
CpC	0.922	0.914	0.917	0.919	0.917	0.939
CpG	1.083	0.939	1.087	0.933	1.087	0.916
CpU	0.875	1.018	0.876	1.012	0.876	1.001
CpA	1.104	1.087	1.113	1.093	1.113	1.1
GpC	1.247	1.03	1.248	1.05	1.248	1.102
GpG	0.921	0.944	0.916	0.943	0.916	0.967
GpU	0.896	0.976	0.899	0.973	0.899	0.93
GpA	0.979	1.087	0.979	1.073	0.979	1.039
UpC	0.978	1.083	0.978	1.067	0.978	1.033
UpG	1.106	1.103	1.115	1.108	1.115	1.118
UpU	1.146	1.021	1.147	1.032	1.147	1.069
UpA	0.741	0.74	0.735	0.75	0.735	0.769
ApC	0.896	1.014	0.899	1.005	0.899	0.96
ApG	0.874	0.969	0.874	0.972	0.874	0.961
ApU	1.1	0.99	1.106	0.99	1.106	0.998
ApA	1.147	1.037	1.148	1.045	1.148	1.082

Differences between bacteria and phage's

In dataset one with bacterial sequences (D. 1 - Bac) and dataset one with phage sequences (D. 1 - Phage) it is possible to notice differences in features related to the number and orientation of genes in the subsequence and dimer constitution, along with smaller differences in amino acids. The same differences are noticed in dataset two with bacteria sequences (D. 2 - Bac) and dataset two with phage sequences (D. 2 - Phage), suggesting a difference in structural composition between phage and bacteria sequences.

Differences between window sizes (10000 bp and 5000 bp)

Two window sizes were tested to divided bacterial and phage genomes, 10000 bp (D. 1 – Bac and D. 1 - Phage) and 5000 bp (D. 2 – Bac and D. 2 – Phage). Comparing both bacterial datasets (D. 1 – Bac and D.2 – Bac) it is possible to notice some differences in the features related to the gene structure. The same was not possible to confirm in the constitutional features (amino acid and dimer).

Differences between phage and phage with alterations

The phages' part of dataset three (D. 3 – Phage) was created with normal and altered phage subsequences. These alterations were made by deleting genes from the subsequences and completing the sequences with a portion of the hosts genome and insertions by inserting a random host genome portion in a random position of the genome and removing the last genes. As shown in table 4, D. 2 – Phage and D. 3 – Phage columns, these changes in the phage subsequences impacted all features but as expected, highly impacted features related to the structure and localization of the genes.

After analysing the distribution of the features, it is possible to confirm that there is a different order of magnitude between the feature columns. Therefore, the next step

consisted of testing different scaler algorithms to transform the data into the same order of magnitude and understand if a scaler is indeed required to improve results. Hence, 10% of the dataset was randomly selected and balanced, to have the same number of phage sequences and bacterial sequences, to ptest various scaler methods posteriorly.

In tables 5, 6, 7, the F1 score, rounded to two decimal cases, is shown for all combinations possible.

Table 5: F1 score of tested machine learning algorithms with different scaller methods of dataset one.

Algorithm	MinMax	Standard	Robust	Normalizer	None
SVM	0.82	0.91	0.91	0.32	0.33
Random forest	0.91	0.91	0.90	0.87	0.90
Knn	0.83	0.89	0.89	0.70	0.62
GradientBoosting	0.84	0.84	0.84	0.78	0.84
MLP	0.88	0.91	0.91	0.71	0.66

Table 6: F1 score of tested machine learning algorithms with different scaler methods of dataset two.

Algorithm	MinMax	Standard	Robust	Normalizer	None
SVM	0.73	0.86	0.84	0.22	0.18
Random forest	0.87	0.87	0.87	0.83	0.87
Knn	0.77	0.83	0.85	0.67	0.56
GradientBoosting	0.77	0.77	0.77	0.68	0.77
MLP	0.83	0.89	0.88	0.65	0.64

Table 7: F1 score of tested machine learning algorithms with different scaler methods of dataset three.

Algorithm	MinMax	Standard	Robust	Normalizer	None
SVM	0.93	0.97	0.97	0.87	0.86
Random forest	0.97	0.97	0.97	0.96	0.97
Knn	0.90	0.96	0.96	0.93	0.90
GradientBoosting	0.94	0.94	0.94	0.92	0.94
MLP	0.96	0.97	0.97	0.93	0.93

Tables 5, 6, 7 show that from the five possibilities tested for each dataset (four different scaler methods and no method use) the standard scaler produced the best results and therefore it will be used in the next steps. Another noticeable result is that the random forest and gradient boosting algorithms seem not to be influenced by the scaler methods.

5.0.3 Feature selection

After accessing the best scaler, the next step is feature selection to exclude possible redundant and uninformative features. A variance method was applied to evaluate the variance between features to this end.

- Variance: This test was performed using a variance threshold of 0.5, resulting in no columns being dropped. These results have shown no values among all samples with low variance.

5.0.4 Assessing dataset unbalance

The first dataset was used to test if the skew between the numbers of phage sequences and bacterial affects model performance. When the model fits the data, it will learn its

patterns and to associate the characteristics of each sample with the desired output. This phase is successful if the model can learn and correctly predict the testing data, and can to be generalized to other data outside the dataset. In this case, the ability to generalize the model can be affected by two problems underfitting and overfitting. In the first case, underfitting refers to the failure of the model to learn the patterns in the training dataset leading to the incapacity of generalization. This problem derives from bad or poor data, specifically, when there is not enough information. For example, not enough samples or features, fail to create a good enough target function or a complete mapping of the data. In the second case, overfitting refers to over tunning the target function or exceeding the data's strict mapping, leaving no flexibility to new data. This problem arises when the model learns the training data too well but performs poorly when chllanged with new data. Another step that can lead to overfitting is hyperparameter tunning. If the hyperparameters found and tested are too strict to the training data, that can lead to a poor model performance with new data. Therefore, assessing the dataset balancing is an important step.

The results for the three datasets, dataset 1 shown in table 8, dataset 2 shown in table 9 and dataset 3 shown in table 10, were tested for:

1. Normal: without balancing.
2. Undersampling: randomly reducing the number of the class with the most samples to match the number of the smallest class.
3. Oversampling: matching the smallest class number of samples to the class with the most samples

Table 8: Dataset 1 (576983 samples from bacteria sequences and 56253 samples from phage sequences) results for all machine learning algorithms tested with a balancing method.

Algorithm	Normal	Undersampling	Oversampling
SVM	0.92	0.96	0.97
Random forest	0.90	0.95	0.98
Knn	0.86	0.94	0.98
GradientBoosting	0.84	0.94	0.95
MLP	0.90	0.96	0.98

Table 9: Dataset 2 (1398656 samples from bacteria sequences and 141577 samples from phage sequences) results for all machine learning algorithms tested with a balancing method.

Algorithm	Normal	Undersampling	Oversampling
SVM	0.89	0.95	0.97
Random forest	0.88	0.95	0.98
Knn	0.85	0.94	0.98
GradientBoosting	0.78	0.92	0.93
MLP	0.90	0.95	0.97

Table 10: Dataset 3 (1398656 samples from bacteria sequences and 536532 samples from phage sequences) results for all machine learning algorithms tested with a balancing method.

Algorithm	Normal	Undersampling	Oversampling
SVM	0.96	0.97	0.98
Random forest	0.96	0.97	0.98
Knn	0.95	0.96	0.98
GradientBoosting	0.93	0.95	0.95
MLP	0.96	0.97	0.98

These results suggest that using a balancing method is better than using the no balancing method. Likewise, it can be seen that oversampling provides a negligible improvement to the model (undersampling +0.01 and in oversampling +0.02).

5.0.5 MODEL OPTIMIZATION

As referenced before, the model optimization can be performed by tunning the algorithm hyperparameters to fit the data. An exhaustive hyperparameter search method (GridSearchcCV) was implemented to determine the highest scores using three cross-validation splits, to test whether hyperparameter tunning was relevant to this work. So, 30% of dataset 1 was not used in training, but along with balancing methods, was used to test if the models and hyperparameters performed better with balanced data. In table 11, the F1 scores before hyperparameter tunning for all tested machine learning algorithms are shown.

Table 11: Model performance before optimization.

Model	Normal	Und	Over
SVM	0.94	0.97	0.97
Random forest	0.99	0.98	0.99
Knn	0.97	0.98	0.99
GradientBoosting	0.93	0.81	0.83
MLP	0.99	0.98	0.99

After testing all possible combinations of hyperparameters, presented in table 2 in section 3.6, the best combinations were found and tested with the same datasets used in the previous test to establish comparisons. Table 12 exhibits the selected hyperparameters with the associated results (F1 score).

Table 12: Model performance for each dataset balance method after optimization.

Model	Hyperparameters	Normal	Und	Over
SVM	C = 10, class_weight='balanced', kernel='rbf'	0.97	0.90	0.91
Random forest	n_estimators=500, min_samples_split=3, criterion='entropy', class_weight='balanced'	0.96	0.83	0.86
Knn	leaf_size = 10, n_neighbors=7, weights='distance'	0.98	0.92	0.97
GradientBoosting	learning_rate=0.1, n_estimators=1000,	0.93	0.86	0.93
MLP	hidden_layer_sizes= (50,50,50), learning_rate='constant', activation = 'tanh', solver='adam', alpha = 0.05	0.95	0.91	0.91

Overall, hyperparameter tuning, except for the dataset without balancing in SVMs and gradient boosting, reduced F1 score. Several possible problems could be the root of the problem associated with the decrease in model performance. The confusion matrix of each model was obtained to explain which prediction values got worse. The values presented in tables 13 and 14 are TP (samples correctly classified as bacteria), FP (samples incorrectly classified as Phage), TN (samples correctly classified as Phage) and FN (samples incorrectly classified as Bacteria) for all algorithms.

Table 13: Confusion matrix for each machine learning algorithm before hyperparameter tunning.

Model	TP	FP	TN	FN
Gradient boosting algorithm with undersampling	23849	259	16924	7184
Gradient boosting algorithm	21845	2263	22830	1278
Gradient boosting algorithm with oversampling	23706	402	17702	6406
Multi-layer perceptron algorithm with undersampling	23451	657	23920	188
Multi-layer perceptron algorithm	23831	277	23797	311
Multi-layer perceptron algorithm with oversampling	23713	395	24028	80
K-nearest neighbors algorithm with undersampling	23469	639	23656	452
K-nearest neighbors algorithm	23980	128	22935	1173
K-nearest neighbors algorithm with oversampling	23607	501	24082	26
Random forest algorithm with undersampling	23445	663	23975	133
Random forest algorithm	23994	114	23703	405
Random forest algorithm with oversampling	23891	217	24037	71
Support vector machine algorithm with undersampling	23395	713	23362	746
Support vector machine algorithm	23622	486	21869	2239
Support vector machine algorithm with overersampling	23515	593	23536	572

Table 14: Confusion matrix for each machine learning algorithm after hyperparameter tunning.

Model	TP	FP	TN	FN
Gradient boosting algorithm with undersampling	23755	353	18501	5607
Gradient boosting algorithm	23751	357	21508	2600
Gradient boosting algorithm with oversampling	23829	279	21956	2152
Multi-layer perceptron algorithm with undersampling	23260	848	20757	3351
Multi-layer perceptron algorithm	23829	279	21956	2152
Multi-layer perceptron algorithm with oversampling	23506	602	20756	3352
K-nearest neighbors algorithm with undersampling	23839	269	20900	3208
K-nearest neighbors algorithm	23974	134	23085	1023
K-nearest neighbors algorithm with oversampling	23705	403	23148	960
Random forest algorithm with undersampling	23973	135	17411	6697
Random forest algorithm	24006	102	22530	1578
Random forest algorithm with oversampling	23851	257	18337	5771
Support vector machine algorithm with undersampling	23332	776	20253	3855
Support vector machine algorithm	23159	949	23680	428
Support vector machine algorithm with oversampling	23348	760	20824	3284

Limited improvements

Support vector machine and gradient boosting models trained with datasets without balancing were the only models that improved performance. In SVM, samples incorrectly classified as phage (FN) decreased showing an improvement when classifying bacteria. Contrary to the improvement in reducing FN, samples incorrectly classified as bacteria (FP) increased. In GB, the opposite occurred. Here, the number of FP decreased after the optimization, but the number of FN increased.

Overall deterioration

Globally, all the models tended to get worst after the optimization. The total number of incorrect FP values decreased from 8307 to 6503 samples after optimization. However, the number of FN increased in a greater proportion from 21264 before optimization to

46018 after optimization. This tendency seems to indicate that after optimization, the models got overfitted bacteria-wise.

Two reasons can help to explain these results. The first reason is overfitting hyperparameters, meaning that adjusting the algorithm decision function could lead to inflexibility when classifying samples. The second possible reason is that the tested hyperparameter and the used method may not be the best.

Hence, hyperparameter tunning was not used in posterior analyses, as the scores, without tunning, were already very high. Additionally, as the scores were high, using hyperparameter tunning could overfit the model and hyperparameters to the existing data, making sample classification outside the training dataset, harder.

5.0.6 Model selection

The three models from the algorithms (Random forests and Multi-layer perceptron) were selected to further testing in light of the previous results (table 11). Prophage Hunter and PHASTER were used to analyse a group of 119 bacterial genomes (chromosomes and plasmids), whose name were taken from the supplementary material of [60] and sequences were taken from the NCBI database to prove that the models could be extrapolated to predict phage sequences from bacterial sequences in real situations.

Each FASTA was submitted to Prophage Hunter and PHASTER, and the predicted zones' coordinates were saved into a file using default options. Then the same dataset was processed by PhagePro's models and the algorithm of boundary location to join adjacent zones and expand to more real boundaries. The coordinates were saved into a file for posterior comparison. The third-party tools' predicted zones were intersected and only common zones were selected to evaluate PhagePro's results. The intersection aims to increase the confidence of the predicted zone, as both tools have different methods for scoring based in the presence of phage proteins that increases the probability of being a

phage zone. This step reduced the possible zones, 327 predicted by Prophage Hunter and 374 predicted by PHASTER, to 154 common zones between.

Although, further *wet lab* validation is needed, using these high confidence putative zones establish a strong base to differentiate model performance by aiming to predict the highest number of putative phage zones. This does not mean that there are only those putative prophage zones in the bacteria, but increasing the capability of predicting high confidence putative zones, can also increase the capability of predicting zones that are not found by other tools, as it is demonstrated in the case study. The number of common zones between the three tools is shown in table 15.

Table 15: Number of common zones between the selected models and the zones predicted and intersected by Prophage Hunter and PHASTER.

Model	Number of common zones
Multi-layer perceptron with oversampling	40
Multi-layer perceptron with undersampling	43
Multi-layer perceptron without dataset balancing	29
Random forest model with oversampling	21
Random forest model with undersampling	43
Random forest model without dataset balancing	8

Table 15 shows that the selected models had a poor performance when tested in real bacterial genomes, chromosomes and plasmids. This performance issue could be associated with the size of the region that is being analysed. Specifically, the region could be too large to be sensitive to small prophage sequences. Additionally, the region could have a profile similar to the host, favouring the classification as bacteria. Another explanation could be that the region suffered alterations with integration or replication of the bacterial genetic material. The phage features were extracted from isolated phages, whose composition could be different from the composition of an integrated phage.

On the other hand, it can be noticed that the models trained with undersampling performed better when presented data outside the artificial training/testing/validation

datasets. Therefore, this data balancing method was used in dataset 2 and dataset 3 to process the unbalanced data.

After building dataset 2 with undersampling, the algorithms that had the highest number of common zones in dataset 1 (Multi-layer perceptron and Random forest) were used to train models and test them. Tables 16 and 17, present the F1 score and confusion matrix for both models:

Table 16: Table with the algorithm used to train the model and the respective F1 score.

Model	F1 score
Multi-layer perceptron	0.92
Random forest	0.93

Table 17: Table with the confusion matrix of both models.

Model	TP	FP	TN	FN
Multi-layer perceptron	55178	5498	56358	4318
Random forest	55719	4957	56490	4186

The results presented in the table above 17 show that the models trained with smaller windows (5000 bp) have the worst overall performance distinguishing bacteria from phage. Nevertheless, this score could not be real as table 15 shows. Thus, both models were tested with the dataset containing 154 common zones. The model trained with Multi-layer perceptron algorithm was able to predict 81 (out of 154) and the model trained with Random forest was able to predict 106. These results suggest that having smaller windows can lead to an increase in the number of zones found. Therefore, to explore different combinations of the dataset, the undersampled dataset was divided and shuffled into five different datasets, training one model with each division and the models tested against the common zones.

Table 18: Results for each division of the dataset and model associated with the common phages.

Model	Number of common zones
Multi-layer perceptron division 1	82
Multi-layer perceptron division 2	110
Multi-layer perceptron division 3	89
Multi-layer perceptron division 4	99
Multi-layer perceptron division 5	75
Random forest division 1	74
Random forest division 2	82
Random forest division 3	71
Random forest division 4	66
Random forest division 5	69

Table 18 shows that overall, the division of the dataset worsen random forest results compared to the results without division though improving the results for Multi-layer perceptron models. From all those models, Multi-layer perceptron division 2 was the model selected to integrate PhagePro, showing better results when predicting high confidence integrated zones.

Dataset 3 was built using undersampling and tested against the common zones to reduce errors when predicting zones with alterations. The results in table 19, show that the best model is the model trained with the SVM algorithm and was therefore selected to be integrated into PhagePro.

Table 19: Number of common zones between the selected models trained with the artificial dataset and the zones predicted and intersected by Prophage Hunter and PHASTER.

Model	Common zones (154)
Multi-layer perceptron algorithm	76
Support Vector machine algorithm	94
Gradient boosting algorithm	90
K-nearest neighbors algorithm	88
Random forest algorithm	73

5.0.7 PUBLISHING ON GALAXY

PhagePro was implemented in the Galaxy framework which provides easy access and a simple interface for bioinformaticians and non-bioinformatics with less computational experience. According to the galaxy structure, the tool can be divided into input, history, and outputs/data visualisation. PhagePro was implemented to require only a FASTA file with the bacterial genome, as shown in figure 8.

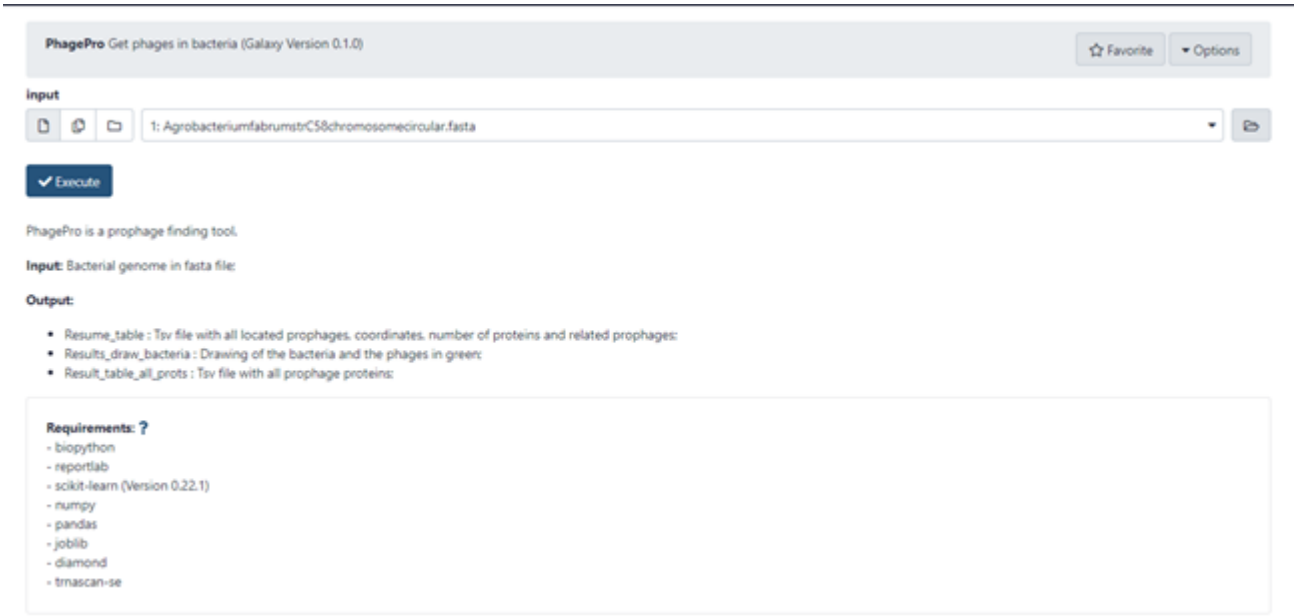


Figure 8: Main page of PhagePro in Galaxy.

For each PhagePro's execution, four elements are produced in the history section. The first represent the input and the other different results. As depicted below 9, from the bottom to top, the first element is the uploaded FASTA file, the second is the table with all prophages and their characteristics, the third element is the drawing of the bacteria and the fourth is all the proteins found and their name.

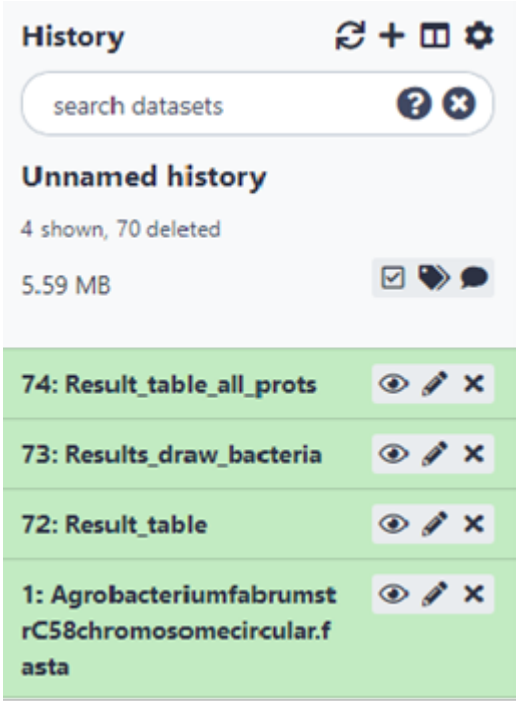


Figure 9: Menu where the results appear.

Regarding data/results visualization, the first result's is a summary table with the prophages candidates, their coordinates, the number of proteins and related phage's. The second result is an illustration, in a pdf format, of the bacteria and the found prophages in green. The last element is a table with all proteins found, where the first column represents the correspondent phage, the second column represents the hits in the Interproscan database, the third is the hits in the NCBI database and the last two columns are the start and end coordinates of the protein. All results can be downloaded in the interface or visualized. The result examples for the simulated example are presented in the following figure.

	Start	End	Protein number	Related phages
cand_172	430677	457559	56	No related phages
cand_469	1173774	1186485	23	No related phages

Figure 10: Resume table of *Agrobacterium fabrum strC58 chromosome circular* results. These results include the coordinates of each prophage, the protein number and related phages.

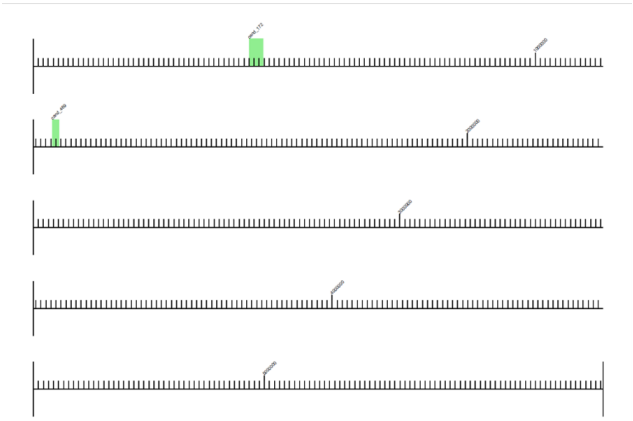


Figure 11: Drawing of *Agrobacterium fabrum strC58 chromosome circular* and the localization of the prophages included in the final result.

	Interproscan	NCBI database	Start	End
cand_172_0_0	UvrD-like helicase C-terminal domain	Nan	431014	433149
cand_172_1	Nan	Nan	433149	433829
cand_172_1_0	Uncharacterized conserved protein (DUF2290)	Nan	433169	433849
cand_172_2	Nan	site-specific integrase	433831	434859
cand_172_2_0	Phage integrase family	Nan	433851	434879
cand_172_3	Nan	Nan	434846	435076
cand_172_4	Nan	DUF2303 family protein	435142	436047
cand_172_4_0	Uncharacterized conserved protein (DUF2303)	Nan	435162	436067
cand_172_5	Nan	hypothetical protein	436107	436439
cand_172_6	Nan	Nan	436470	436670
cand_172_7	Nan	Nan	436730	437041
cand_172_8	Nan	Nan	437097	437288
cand_172_9	Nan	Nan	437346	437492
cand_172_10	Nan	tellurite resistance protein	437495	437917
cand_172_10_0	Tellurite resistance protein TerB	Nan	437515	437937
cand_172_11	Nan	LexA-like protein	438416	439198
cand_172_11_0	Helix-turn-helix	Nan	438436	439218
cand_172_11_1	Peptidase S24/S26A/S26B/S26C	Nan	438456	439238
cand_172_11_2	Peptidase S24-like	Nan	438476	439258
cand_172_12	Nan	Nan	439224	439523
cand_172_13	Nan	hypothetical protein	439628	440125
cand_172_14	Nan	Nan	440122	440373
cand_172_15	Nan	hypothetical protein	440373	441758
cand_172_16	Nan	Nan	441758	441913
cand_172_17	Nan	Nan	441917	442468
cand_172_18	Nan	hypothetical protein	442444	443106
cand_172_19	Nan	hypothetical protein	443103	443396
cand_172_20	Nan	Nan	443396	444616
cand_172_20_0	Helix-turn-helix domain	Nan	443416	444636
cand_172_21	Nan	Nan	444606	445292
cand_172_21_0	Transcription termination factor nusG	Nan	444626	445312
cand_172_22	Nan	hypothetical protein	445537	446106
cand_172_23	Nan	Nan	446267	447001
cand_172_24	Nan	Nan	446964	447221
cand_172_24_0	Phage terminase large subunit (GpA)	Nan	446984	447241
cand_172_25	Nan	Nan	447298	447621

Figure 12: Table with the protein of one of the prophage candidates *Agrobacterium fabrum strC58 chromosome circular*. This table shows from which database the name proteins come from and their coordinates.

5.0.8 Case Study

Agrobacterium fabrum strC68

The Agrobacterium fabrum strC68 genome, along with its plasmids, was submitted to the tool in FASTA format to test the full performance and development of the tool. The tool performance will be evaluated through several analyses of different parts of the workflow. The first is to analyse the number of putative prophage windows that the tool found. Up to this part, the tool divided the genome in 5000 bp windows with 2500 bp

overlap in each one of them, calculated 42 features, classified the zones as prophage or bacteria, merged adjacent zones and expanded each border, either by finding tRNA in the proximity or by finding the best BLAST hit in the conjugation of small fragments upstream and downstream. These steps resulted in 94 zones with structural and compositional resemblance to a phage sequence, which were passed on the ensuing analysis.

Table 25 show the predicted zones with the respective coordinates.

The second part is analysing the constitution of each putative zone. For each putative prophage zone, all proteins found inside the boundaries were sought in Interpro and Pfam and SwissProt database (using the NCBI request to find the name of the protein). Then each protein was associated with a function of the phage lifecycle, using keywords. For this task, a dictionary where the keys are the lifecycle functions, and values each lifecycle's function keywords was used. Table 26 contains the number of functions found in each putative zone.

After obtaining all possible functions for the proteins in the putative prophage zone, the third part is excluding all putative zones that do not meet the established rules. As shown in table 27, each candidate is assigned to one zone, or with the reason that led to the exclusion of that zone.

After evaluating all zones 92 out of the 94 zones were excluded. Most zones were excluded for reason 3 in table 3, which clearly shows that there is still a large gap in information to annotate proteins, that needs to be addressed to have more detailed and accurate results. Nevertheless, with the current information, as the putative prophage zones include 82% of protein without association to a function on the phage lifecycle they were excluded. Moreover, the second reason to exclude prophage zones was reason 4 in the same table. Hence, although the percentage of known proteins function is higher, they are not associated with important phage lifecycle functions, such as, *integrase* for integration, *capsid* for assembly and *tail* for penetration.

The final results for the case study is shown in table 20 and table 21:

Table 20: Final candidates and their characterisation.

Candidate	Start	End	Number of proteins	Number of hits in the prophage protein database
Candidate 172	430677	457559	56	14
Candidate 469	1173774	1186485	23	7

Table 21: Proteins of the final candidates associated with a phages' lifecycle function.

Candidate	Number of proteins associated with phage functions
Candidate 172	biosynthesis: 5, undetermined: 38, hypothetical: 3, integration: 2, assembly: 2, penetration: 5, lysis: 1
Candidate 469	biosynthesis: 6, undetermined: 13, penetration: 2, integration: 2

If the proteins of each candidate are analysed with more detail, there are good markers that the predicted region is a prophage region. As shown in table 21, Candidate 172 comprises proteins that are more exclusive to phage's, namely proteins associated with the *integrase* family, several proteins associated with phage *tails* and proteins associated with the regulation of phage protein expression. Candidate 469, although smaller, contains proteins associated with phage *tails*, phage integration and proteins responsible for evasion to bacterial restriction mechanisms.

The same FASTA file was submitted to Prophage Hunter and PHASTER, providing the following results:

Prophage Hunter with similarity matching

Table 22: Results for Prophage Hunter with similarity matching.

Predicted zones	Start	End	Score
1	430704	460982	0.71
2	3628690	3647612	0.82
3	5580626	5605220	0.86

Prophage Hunter without similarity matching

Table 23: Results for Prophage Hunter without similarity matching.

Predicted zones	Start	End	Score
1	430704	460982	0.71
2	549543	565973	0.92
3	2962819	2981286	0.68

PHASTER

Table 24: Results for PHASTER.

Predicted zones	Start	End	Score
1	430704	460936	Intact
2	940715	959523	Incomplete
3	5277049	5281544	Questionable

At first glance, it is possible to notice that there is only one zone in common with our results; thus, a more in-depth analysis was performed. From the three predicted zones shown in table 22, only the first match PhagePro's final results. The second zone has a high score (0.82), and an overlapping zone was predicted by our tool (candidate 1454). Nevertheless, this zone was rejected because of reason 3 in table 3. The third zone has a high score (0.86), and a smaller overlapping zone is included in the preliminary PhagePro results (table 25). Nevertheless, the zone was rejected because of reason 7 in table 3. In this last case, it is possible that if the boundary location method were adjusted, this zone could be included in the final results, because it had already six hits in the prophage protein database.

Table 23 has three predicted zones, with the first matching PhagePro. The second zone has a high score (0.92), overlapping the preliminary PhagePro results (candidate 221 in table 25). A more in-depth analysis of candidate 221 reveals that the zone has no hits with the prophage protein database and was rejected for reason 3. The third

zone with a lower score (0.68), is also present in the preliminary results of PhagePro (candidate 1187). However, it was rejected because of reason 7, making it a highly unknown zone.

Table 24 has three predicted zones but only the first in common with PhagePro final results. Nevertheless, our model was able to find a smaller zone inside the second predicted zone (candidate 381) but, although the zone had good indicators of being a prophage (contains *tail* and *lysis* associated proteins). However, the zone did not pass the prediction confidence conditions because of reason 1. The last predicted zone contains about 4500 bp and seven proteins, thus being excluded from PhagePro's final results. Our tool predicted an adjacent zone with a small overlapping sequence segment (candidate 2105). Still, the zone was rejected due to reason 3.

6

CONCLUSION

The main objective of this work was to create a tool that could help users find prophages in bacterial genomes to track changes in bacterial genomes using phages as vectors. Five machine learning algorithms were used to train models and the resulting models were tested with different datasets obtained using different techniques. Additionally, two balancing methods were tested to reduce the skew in phage representation in the datasets, either by producing more phage instances or reducing bacterial instances. Other methods were created to approximate this computational approach to reality. Furthermore, two approximations to find the real prophage boundaries were implemented and structural conditions were established to classify phage activity, thus improving the prediction by excluding zones that do not have known phage proteins or have a highly degraded structure.

Combinations of the different machine learning algorithms and scaling methods were tested for each dataset, to determine the best scaling method. The scaling method that had the best results overall was the standard scaling method from all combinations. Moreover, the random forest algorithm and gradient boosting do not seem to be influenced by scaling methods expect for the normalizer that deteriorates model's performance. Regarding the balancing methods, a portion of all the datasets was tested using an oversampler and an undersampler and combining the balancing methods with the plus models, resulting in slightly better results in all models when using the oversampler. Likewise, different combinations of hyperparameter were tested using as data

a portion of dataset one. The results have shown that, for the most part, tunning the hyperparameters made the model performance worst, possibly by overfitting bacteria predictions. This hypothesis is reinforced by comparing the number of FP in table 11 (before hyperparameter tunning) and table 12 (after hyperparameter tunning). After finding the best hyperparameters, there was an increase in erroneous phage predictions. In addition, a comparison was made between zones predicted by models in development and zones predicted by Prophage Hunter and PHASTER. The first set of models (trained with 10000 bp windows) performed poorly, as the best models only reached 43 of 154 zones. The second set of models was created using previously acquired knowledge with the algorithms mentioned above, but only with undersampling as a balancing method , as the best models were obtained with undersampling. Although the F1 score metric decreased, the number of zones predicted by the random forest algorithm reached 106 of 154 zones. Additionally, a new approach to reduce model bias was tested, using a stratified split of the data, resulting in five splits (five datasets), and each one used to train and test models. All models created from the stratified split were used in the 154 zones, whereas one of models using Multi-layer perceptron algorithm achieved 110 zones.

Finally, to bridge gaps in the prediction of prophage zones with bacterial alterations, dataset 3 was built and used to train and test models, using undersampling as a balancing method. From the five machine-learning algorithms tested, Support Vector Machine achieved the highest number of zones found (94/154).

Regarding the performance of the tool in the case study, PhagePro predicted two zones with good phage indicators, such as the different composition and structure of the sequence in the bacterial genome and exclusive phage proteins correlated with prophage proteins (integrase, tail, among others). Furthermore, comparing the results with the results of Prophage Hunter and PHASTER, it is possible to notice some limitations in the other tools that PhagePro is capable of surpassing, such as excluding zones with little protein knowledge and small zones.

The main objective for this thesis was accomplished; building a tool that could find and characterise prophage sequences in bacterial genomes, but there are limitations and future work that will be addressed to improve the tool and its capabilities.

Regarding limitations, three major areas can be improved. The first is related to predictive models. As mentioned before, the models with the best scores detecting the common zones, and selected to incorporate the tool workflow, did not predict all existing zones, showing that there is room for improvements. Either by trying different machine learning algorithms or approaches to the classification problem, perhaps deep learning would be more suited for this task, or a better hyperparameter optimization method. Additionally, adding more phage information or samples could provide a broader representation of the phage population. One example of this information is phage sequences, extracted from bacteria through scientific experiments, that could have different feature variations making the model more aware of phage alterations. The second major area that could be improved is boundary locating. Although the methods implemented in this tool make a good approximation to the real prophage boundaries, predicting the exact phage insertion location can be a good improvement. It would remove unwanted bacteria parts or add smaller but important fragments of putative prophages. The last major area to be improved is phage activity classification. Unfortunately, there is little information on the topic of what is a complete prophage or a prophage capable of performing all steps in his lifecycle. Furthermore, a good portion of phage proteins is hypothetical, making the classification of these proteins to a lifecycle step difficult. Other smaller improvements include more input options to allow other file types to be processed or a simple sequence without needing a file. Moreover, prophage representation in the results could be improved to allow the user to interact with the different results and illustrations.

6.1 FUTURE WORK

Future work can help to improve the quality of the predictions and user experience. Additionally, when the tool is available to other users, user feedback could highlight problems/optimizations. Nevertheless, the following improvement road map is proposed:

- Increase phage data, either by gather experimental data or building new features.
- Allow prophage illustration.
- Allow the user to select what conditions to exclude putative prophage zones to apply in the workflow, allowing the user to explore less known phages that have proteins less represented in databases.
- Decrease tool run time.
- Testing different hyperparameters and hyperparameter tunning methods to improve the prediction results.
- Gather information about the structure and localization of integration sites, BLASTing prophage sequences to bacterial genomes, saving small size hits (for example, less than 250 bp) and searching the location of the BLAST hit in both genomes, seeing in which location/ protein is located.
- Improve boundary location.
- Build a database with phage related information, for example, phage composition in NCBI databases and prophages experimentally extracted, conserved protein sites and different variations for major phage proteins, like *integrase, capsid* and *tail*.
- A method to compare differences between genomes, either by inputting the genomes or comparing the found prophage genomes using the tool with genomes stored in the database.

- A predictive model that could bridge gaps between the protein and her function in the phage lifecycle, for example by calculating features directly from the sequence to the function avoiding searching databases and surpassing knowledge limitations in databases.

- A hub or workflow of tools that could expand the knowledge of phages or events caused by phages. For example, creating a tool that could simulate the behavior of a bacteria or community of bacteria when a certain concentration of phages is added or removed from the environment. Or, if the objective is to increase or create a metabolic pathway to produce new products, a tool that could simulate the behavior of a bacterium when a phage enters the bacteria, hijacking the bacterial machinery or integrating the phage genome in the bacteria changing the expression of metabolites or creating new metabolic pathways.

BIBLIOGRAPHY

[1] Woese, C. R., & Fox, G. E. (1977). *Phylogenetic structure of the prokaryotic domain: The primary kingdoms (archaebacteria/eubacteria/urkaryote/16S ribosomal RNA/molecular phylogeny)* (tech. rep. No. 11).

[2] Rivera, M. C., & Lake, J. A. (2004). The ring of life provides evidence for a genome fusion origin of eukaryotes. *Nature*, *431*(7005), 152–155.

[3] Clokie, M., & Kropinski, A. (2009). *Bacteriophages: Volume I*.

[4] H Andrewes, B. C. (1955). *The Classification of Viruses*.

[5] Steinhaus, E. A. (n.d.). *NOMENCLATURE AND CLASSIFICATION OF INSECT VIRUSES'*.

[6] LWOFF, A. (1957). The Concept of Virus. *Journal of General Microbiology*, *17*(1), 239–253.

[7] LWOFF, A., HORNE, R., & TOURNIER, P. (1962). A system of viruses. *Cold Spring Harbor symposia on quantitative biology*, *27*, 51–55.

[8] Abeles, A. L., Snyder, K. M., & Chattoraj, D. K. (1984). P1 plasmid replication: Replicon structure. *Journal of Molecular Biology*, *173*(3), 307–324.

[9] Gelderblom, H. R. (1996). *Structure and Classification of Viruses* (4th edition). University of Texas Medical Branch at Galveston.

[10] Norrby, E. (n.d.). *The morphology of virus particles . Classification of viruses*.

[11] Campbell, A. M. [Allan M]. (n.d.). *Bacteriophages*.

[12] Yang, H., Ma, Y., Wang, Y., Yang, H., Shen, W., & Chen, X. (2014). Transcription regulation mechanisms of bacteriophages: Recent advances and future prospects.

[13] Gruffat, H., Marchione, R., & Manet, E. (2016). Herpesvirus Late Gene Expression: A Viral-Specific Pre-initiation Complex Is Key. *Frontiers in Microbiology*, *7*(JUN), 869.

[14] White, E. A., & Spector, D. H. (2007). Early viral gene expression and function. *Human Herpesviruses: Biology, Therapy, and Immunoprophylaxis*.

[15] Replication, V. D. N. A. (2017). Introduction to DNA Viruses, 0–5.

[16] Payne, S. (2017). Introduction to RNA Viruses. *Viruses*, 97–105.

[17] Barr, J. N., & Fearns, R. (2016). Genetic Instability of RNA Viruses. *Genome Stability: From Virus to Human Application*, (January), 21–35.

[18] Sanjuán, R., Nebot, M. R., Chirico, N., Mansky, L. M., & Belshaw, R. (2010). Viral Mutation Rates. *JOURNAL OF VIROLOGY*, *84*(19), 9733–9748.

[19] Norrby, E. (1983). The morphology of virus particles. Classification of viruses. In *Textbook of medical virology* (pp. 4–16).

[20] STRAUSS, J. H., & STRAUSS, E. G. (2008). The Structure of Viruses. In *Viruses and human disease* (pp. 35–62).

[21] Fermin, G. (2018). Host Range, Host-Virus Interactions, and Virus Transmission. In *Viruses: Molecular biology, host interactions, and applications to biotechnology* (pp. 101–134).

[22] McLeish, M. J., Fraile, A., & García-Arenal, F. (2018). Ecological Complexity in Plant Virus Host Range Evolution. *Advances in Virus Research*, *101*, 293–339.

[23] Suttle, C. A. (2000). Cyanobacteria and Eukaryotic Algae. *Algae*, 247–296.

[24] International Committee on Taxonomy of Viruses (ICTV). (n.d.).

[25] Turner, D., Kropinski, A. M., & Adriaenssens, E. M. (2021). A Roadmap for Genome-Based Phage Taxonomy. *Viruses*, *13*(3).

[26] Chibani, C. M., Farr, A., Klama, S., Dietrich, S., & Liesegang, H. (2019). Classifying the unclassified: A phage classification method. *Viruses*, *11*(2).

[27] Domingo-Calap, P., & Delgado-Martínez, J. (2018). Bacteriophages: Protagonists of a post-antibiotic era. *Antibiotics*, *7*(3), 1–16.

[28] Thomas, C. E., Ehrhardt, A., & Kay, M. A. (2003). Progress and problems with the use of viral vectors for gene therapy.

[29] d'Herelle, F. (1931). *BACTERIOPHAGE AS A TREATMENT IN ACUTE MEDICAL AND SURGICAL INFECTIONS**. Yale University School of Medicine.

[30] Wittebole, X., De Roock, S., & Opal, S. M. (2014). A historical overview of bacteriophage therapy as an alternative to antibiotics for the treatment of bacterial pathogens. *Virulence*, *5*(1), 226–235.

[31] Roach, D. R., & Debarbieux, L. (2017). Phage therapy: awakening a sleeping giant. *Emerging Topics in Life Sciences*, *1*(1), 93–103.

[32] Endersen, L., O'Mahony, J., Hill, C., Ross, R. P., McAuliffe, O., & Coffey, A. (2014). Phage Therapy in the Food Industry. *Annual Review of Food Science and Technology*, *5*(1), 327–349.

[33] Sano, E., Carlson, S., Wegley, L., & Rohwer, F. (2004). Movement of viruses between biomes. *Applied and Environmental Microbiology*, *70*(10), 5842–5846.

[34] Cobián, A. G., Uemes, G., Youle, M., Cantú, V. A., Cantú, C., Felts, B., … Rohwer, F. (n.d.). Viruses as Winners in the Game of Life.

[35] Principi, N., Silvestri, E., & Esposito, S. (2019). Advantages and Limitations of Bacteriophages for the Treatment of Bacterial Infections. *Frontiers in Pharmacology*, *10*(MAY), 513.

[36] Manohar, P., Nachimuthu, R., & Lopes, B. S. (n.d.). The therapeutic potential of bacteriophages targeting gram-negative bacteria using Galleria mellonella infection model.

[37] Fischetti, V. A. (2011). Exploiting what phage have evolved to control gram-positive pathogens. *Bacteriophage*, *1*(4), 188–194.

[38] Kaján, G. L., Doszpoly, ·. A., Zoltán, ·., Tarján, L., Vidovszky, M. Z., & Papp, ·. T. (2020). Virus-Host Coevolution with a Focus on Animal and Human DNA Viruses. *Journal of Molecular Evolution*, *88*, 41–56.

[39] Barrangou, R., & Marraffini, L. A. (2014). CRISPR-Cas systems: prokaryotes upgrade to adaptive immunity.

[40] Guenther, C. M., Kuypers, B. E., Lam, M. T., Robinson, T. M., Zhao, J., & Suh, J. (n.d.). Synthetic Virology: Engineering Viruses for Gene Delivery.

[41] Krause, R. M. (n.d.). *STUDIES ON THE BACTERIOPHAGES OF HEMOLYTIC STREPTOCOCCI II. ANTIGENS I˜ELEASED FROM THE STREPTOCOCCAL CELL WALL BY A PHAGE-AssoCIATED LYSIN*.

[42] Vale, F. F. [Filipa F.], & Lehours, P. (2018). Relating phage genomes to helicobacter pylori population structure: General steps using whole-genome sequencing data.

[43] Almand, E. A., Moore, M. D., & Jaykus, L. A. (2017). Virus-bacteria interactions: An emerging topic in human infection. *Viruses*, *9*(3).

[44] Haaber, J., Leisner, J. J., Cohn, M. T., Catalan-Moreno, A., Nielsen, J. B., Westh, H., . . . Ingmer, H. (2016). Bacterial viruses enable their host to acquire antibiotic resistance genes from neighbouring cells. *Nature Communications*, *7*.

[45] Chen, J., Quiles-Puchalt, N., Chiang, Y. N., Bacigalupe, R., Fillol-Salom, A., Chee, M. S. J., . . . Penadés, J. R. (2018). Genome hypermobility by lateral transduction. *Science*, *362*(6411), 207–212.

[46] Torres-Barceló, C. (n.d.). The disparate effects of bacteriophages on antibiotic-resistant bacteria.

[47] Hooi, J. K., Lai, W. Y., Ng, W. K., Suen, M. M., Underwood, F. E., Tanyingoh, D., . . . Ng, S. C. (2017). Global Prevalence of Helicobacter pylori Infection: Systematic Review and Meta-Analysis. *Gastroenterology*, *153*(2), 420–429.

[48] Kamangar, F., Qiao, Y. L., Blaser, M. J., Sun, X. D., Katki, H., Fan, J. H., . . . Dawsey, S. M. (2007). Helicobacter pylori and oesophageal and gastric cancers in a prospective study in China. *British Journal of Cancer*, *96*(1), 172–176.

[49] Bravo, D., Hoare, A., Soto, C., Valenzuela, M. A., & Quest, A. F. (2018). Helicobacter pylori in human health and disease: Mechanisms for local gastric and systemic effects.

[50] De Francesco, V., Giorgio, F., Hassan, C., Manes, G., Vannella, L., Panella, C., . . . Zullo, A. (2010). *Worldwide H. pylori Antibiotic Resistance: a Systematic Review* (tech. rep. No. 4).

[51] Qasim, A., O'Morain, C. A., & O'Connor, H. J. (2009). ¡i¿Helicobacter pylori¡/i¿ eradication: role of individual therapy constituents and therapy duration. *Fundamental & Clinical Pharmacology*, *23*(1), 43–52.

[52] Ghannad, M. S., & Mohammadi, A. (2012). Bacteriophage: Time to re-evaluate the potential of phage therapy as a promising agent to control multidrug-resistant bacteria.

[53] Slopek, S., Durlakowa, I., Weber-Dabrowska, B., Kucharewicz-Krukowska, A., Dabrowski, M., & Bisikiewicz, R. (1983). Results of bacteriophage treatment of suppurative bacterial infections. II. Detailed evaluation of the results.

[54] Blader, I., Coleman, B., Chen, C.-T., & Gubbels, M.-J. (n.d.). The lytic cycle of Toxoplasma gondii: 15 years later.

[55] Howard-Varona, C., Hargreaves, K. R., Abedon, S. T., & Sullivan, M. B. (2017). MINI REVIEW Lysogeny in nature: mechanisms, impact and ecology of temperate phages Why study lysogeny? *Nature Publishing Group*, *11*, 1511–1520.

[56] Horie, M., Honda, T., Suzuki, Y., Kobayashi, Y., Daito, T., Oshida, T., . . . Tomonaga, K. (2010). Endogenous non-retroviral RNA virus elements in mammalian genomes. *Nature*.

[57] Feschotte, C. (n.d.). Virology: Bornavirus enters the genome.

[58] Malik, S. S., Azem-e-Zahra, S., Kim, K. M., Caetano-Anollés, G., & Nasir, A. (2017). Do Viruses Exchange Genes across Superkingdoms of Life? *Frontiers in Microbiology*, *8*(OCT), 2110.

[59] Carlson, K. (2005). *Working with bacteriophages: common techniques and methodological approaches*.

[60] Casjens, S. (2003). Prophages and bacterial genomics: What have we learned so far? *Molecular Microbiology*, *49*(2), 277–300.

[61] Oleastro, M., & Vale, F. F. [F F]. (2013). Helicobacter pylori eradication – the alternatives beyond antibiotics. (January), 1656–1667.

[62] Clokie, M. R. J., & Kropinski, A. M. (2009). Bacteriophages : methods and protocols. *Methods in molecular biology*, *501*, xxii, 307 p.

[63] Sharma, D., Priyadarshini, P., & Vrati, S. (2015). Unraveling the Web of Viroinformatics: Computational Tools and Databases in Virus Research.

[64] Ahlgren, N. A., Ren, J., Lu, Y. Y., Fuhrman, J. A., & Sun, F. (2017). Alignment-free d2 oligonucleotide frequency dissimilarity measure improves prediction of hosts from metagenomically-derived viral sequences. *Nucleic Acids Research*, *45*(1), 39–53.

[65] Srividhya, K. V., Alaguraj, V., Poornima, G., Kumar, D., Singh, G. P., Raghavenderan, L., . . . Krishnaswamy, S. (2007). Identification of Prophages in Bacterial Genomes by Dinucleotide Relative Abundance Difference. *PLoS ONE*, *2*(11), e1193.

[66] Ru, X., Li, L., & Wang, C. (2019). Identification of phage viral proteins with hybrid sequence features. *Frontiers in Microbiology*, *10*(MAR), 507.

[67] Bossi, L., Fuentes, J. A., Mora, G., & Figueroa-Bossi, N. (2003). Prophage Contribution to Bacterial Population Dynamics. *Journal of Bacteriology*, *185*(21), 6467–6471.

[68] Hatfull, G. F., & Hendrix, R. W. [Roger W]. (2011). Bacteriophages and their Genomes.

[69] Nelson, K. E., Weinel, C., Paulsen, I. T., Dodson, R. J., Hilbert, H., Martins dos Santos, V. A. P., . . . Fraser, C. M. (2002). Complete genome sequence and comparative analysis of the metabolically versatile Pseudomonas putida KT2440. *Environmental Microbiology*, *4*(12), 799–808.

[70] Edwards, R. A., Mcnair, K., Faust, K., Raes, J., & Dutilh, B. E. (2016). Computational approaches to predict bacteriophage-host relationships. *FEMS Microbiology Reviews*, *048*, 258–272.

[71] Thaler, J.-O., Baghdiguian, S., Noe, N., & Boemare, N. (1995). *Purification and Characterization of Xenorhabdicin, a Phage Tail-Like Bacteriocin, from the Lysogenic Strain F1 of Xenorhabdus nematophilus* (tech. rep. No. 5).

[72] Und, M., Weihenstephan, L., Zink, R., Loessner, M. J., & Scherer, S. (1995). Characterization of cryptic prophages (monocins) in Listeria and sequence analysis of a holinlendolysin gene. *Microbiology*, *141*, 2577–2584.

[73] Hoa NGUYEN, A., Tomita, T., Hirota, M., Sato, T., & Kamio, Y. (1999). Bioscience, Biotechnology, and Biochemistry A Simple Purification Method and Morphology and Component Analyses for Carotovoricin Er, a Phage-tail-like Bacteriocin from the Plant Pathogen Erwinia carotovora Er A Simple Purification Method and Morphology and Component Analyses for Carotovoricin Er, a Phage-tail-like Bacteriocin from the Plant Pathogen. *Bioscience, Biotechnology, and Biochemistry*, *63*(8), 1360–1369.

[74] Lang, A. S., & Beatty, J. T. (2007). Importance of widespread gene transfer agent genes in α-proteobacteria. *Trends in Microbiology*, *15*(2), 54–62.

[75] Novick, R. P., Christie, G. E., & Penadés, J. R. (2010). The phage-related chromosomal islands of Gram-positive bacteria NIH Public Access $watermark-text $watermark-text $watermark-text. *Nat Rev Microbiol*, *8*(8), 541–551.

[76] Sutter, M., Boehringer, D., Gutmann, S., Günther, S., Prangishvili, D., Loessner, M. J., … Ban, N. (2008). Structural basis of enzyme encapsulation into a bacterial nanocompartment. *Nature Structural and Molecular Biology*, *15*(9), 939–947.

[77] Labrie, S. J., Samson, J. E., & Moineau, S. (2010). Bacteriophage resistance mechanisms. *Nature Reviews Microbiology*, *8*(5), 317–327.

[78] Leskinen, K., Blasdel, B. G., Lavigne, R., & Skurnik, M. (n.d.). RNA-Sequencing Reveals the Progression of Phage-Host Interactions between R1-37 and Yersinia enterocolitica.

[79] Mojardín, L., & Salas, M. (2016). Global Transcriptional Analysis of Virus-Host Interactions between Phage 29 and Bacillus subtilis. *90*.

[80] Dion, M. B., Oechslin, F., & Moineau, S. (2020). Phage diversity, genomics and phylogeny. *Nature Reviews Microbiology*, *18*(3), 125–138.

[81] Sallie, R. (2005). Virology Journal Replicative Homeostasis: A fundamental mechanism mediating selective viral replication and escape mutation.

[82] Zhou, Y., Liang, Y., Lynch, K. H., Dennis, J. J., & Wishart, D. S. (n.d.). PHAST: A Fast Phage Search Tool.

[83] Arndt, D., Grant, J. R., Marcu, A., Sajed, T., Pon, A., Liang, Y., & Wishart, D. S. (2016). PHASTER: a better, faster version of the PHAST phage search tool. *Nucleic acids research*, *44*(W1), W16–W21.

[84] Akhter, S., Aziz, R. K., & Edwards, R. A. (n.d.). PhiSpy: a novel algorithm for finding prophages in bacterial genomes that combines similarity-and composition-based strategies.

[85] Song, W., Sun, H. X., Zhang, C., Cheng, L., Peng, Y., Deng, Z., ... Xiao, M. (2019). Prophage Hunter: an integrative hunting tool for active prophages. *Nucleic acids research*, *47*(W1), W74–W80.

[86] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs* (tech. rep. No. 17). Oxford University Press.

[87] Ye, Y., Choi, J. H., & Tang, H. (2011). RAPSearch: A fast protein similarity search tool for short reads. *BMC Bioinformatics*, *12*(1), 1–10.

[88] Buchfink, B., Xie, C., & Huson, D. H. (2014). Fast and sensitive protein alignment using DIAMOND. *Nature Methods*, *12*(1), 59–60.

[89] Krallinger, M., & Valencia, A. (2005). Text-mining and information-retrieval services for molecular biology.

[90] Eddy, S. R. [Sean R]. (n.d.). *Profile hidden Markov models*.

[91] Brief review: Gene-Finding for Bacterial Genomes — BioBam. (n.d.).

[92] Team I Gene Prediction Group - Compgenomics 2019. (n.d.).

[93] Team II Gene Prediction Group - Compgenomics 2019. (n.d.).

[94] Team III Gene Prediction Group - Compgenomics 2019. (n.d.).

[95] Salzberg, S. L., Deicher, A. L., Kasif, S., & White, O. (1998). Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, *26*(2), 544–548.

[96] Besemer, J., Lomsadze, A., & Borodovsky, M. (2001). *GeneMarkS: a self-training method for prediction of gene starts in microbial genomes. Implications for finding sequence motifs in regulatory regions* (tech. rep. No. 12).

[97] Hyatt, D., Chen, G. L., LoCascio, P. F., Land, M. L., Larimer, F. W., & Hauser, L. J. (2010). Prodigal: Prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, *11*, 119.

[98] Bailly-Bechet, M., Vergassola, M., & Rocha, E. (2007). Causes for the intriguing presence of tRNAs in phages. *Genome Research*, *17*(10), 1486–1495.

[99] Ventura, M., Lee, J. H., Canchaya, C., Zink, R., Leahy, S., Moreno-Munoz, J. A., ... Van Sinderen, D. (2005). Prophage-like elements in bifidobacteria: Insights from genomics, transcription, integration, distribution, and phylogenetic analysis. *Applied and Environmental Microbiology*, *71*(12), 8692–8705.

[100] Lowe, T. M., & Eddy, S. R. [Sean R]. (1997). *tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence* (tech. rep. No. 5).

[101] Eddy, S. R. [Sean R.], & Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucleic Acids Research*, *22*(11), 2079–2088.

[102] Laslett, D., & Canback, B. (n.d.). ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences.

[103] Laslett, D., Canback, B., & Andersson, S. (2002). BRUCE: A program for the detection of transfer-messenger RNA genes in nucleotide sequences. *Nucleic Acids Research*, *30*(15), 3449–3453.

[104] El-Gebali, S., Mistry, J., Bateman, A., Eddy, S. R., Luciani, A., Potter, S. C., ... Finn, R. D. (2019). The Pfam protein families database in 2019. *Nucleic Acids Research*, *47*(D1), D427–D432.

[105] D158-D169. (2017). UniProt: the universal protein knowledgebase The UniProt Consortium. *Nucleic Acids Research*, *45*.

[106] Cheng, H., Schaeffer, R. D., Liao, Y., Kinch, L. N., & Pei, J. (2014). ECOD: An Evolutionary Classification of Protein Domains. *PLoS Comput Biol*, *10*(12), 1003926.

[107] NCBI Resource Coordinators. (2015). Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, *44*, 7–19.

[108] Grazziotin, A. L., Koonin, E. V., & Kristensen, D. M. (2017). Prokaryotic Virus Orthologous Groups (pVOGs): A resource for comparative genomics and protein family annotation. *Nucleic Acids Research*, *45*(D1), D491–D498.

[109] Leplae, R. [R.]. (2004). ACLAME: A CLAssification of Mobile genetic Elements. *Nucleic Acids Research*, *32*(90001), 45D–49.

[110] Leplae, R. [Raphaël], Lima-Mendez, G., & Toussaint, A. (2009). ACLAME: A CLAssification of mobile genetic elements, update 2010. *Nucleic Acids Research*, *38*(SUPPL.1), D57.

[111] Liu, H., & Teow, L. N. (2005). Performance evaluation of protein sequence clustering tools. In *Lecture notes in computer science* (Vol. 3515, *2*, pp. 877–885).

[112] Fu, L., Niu, B., Zhu, Z., Wu, S., & Li, W. (2012). Sequence analysis CD-HIT: accelerated for clustering the next-generation sequencing data. *28*(23), 3150–3152.

[113] Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*.

[114] DBSCAN Clustering — Explained. Detailed theorotical explanation and... — by Soner Yıldırım — Towards Data Science. (n.d.).

[115] Tong, J. C. (2013). *Cross-Validation*.

[116] How to Deal with Missing Values in Your Dataset - KDnuggets. (n.d.).

[117] Feature Selection and Feature Extraction in Machine Learning: An Overview — by Mehul Ved — Medium. (n.d.).

[118] How to Choose a Feature Selection Method For Machine Learning. (n.d.).

[119] Feature Selection using Wrapper Method - Python Implementation. (n.d.).

[120] Hands-on with Feature Selection Techniques: Embedded Methods — by Younes Charfaoui — Heartbeat. (n.d.).

[121] (2) (PDF) A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. (n.d.).

[122] Berrar, D. (2018). Cross-validation. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, *1-3*, 542–545.

[123] 15.3 - Bootstrapping — STAT 555. (n.d.).

[124] Dayan, P., & Niv, Y. (2008). Reinforcement learning: The Good, The Bad and The Ugly.

[125] Zhang, M., Yang, L., Ren, J., Ahlgren, N. A., Fuhrman, J. A., & Sun, F. (2017). Prediction of virus-host infectious association by supervised learning methods. *BMC Bioinformatics*, *18*(Suppl 3).

[126] Boulesteix, A. L., Janitza, S., Kruppa, J., & König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(6), 493–507.

[127] 15.3 - Bootstrapping — STAT 555. (n.d.).

[128] Support Vector Machines(SVM) — An Overview — by Rushikesh Pupale — Towards Data Science. (n.d.).

[129] Decision Tree Algorithm, Explained - KDnuggets. (n.d.).

[130] Mohammed, M., Khan, M. B., & Bashie, E. B. M. (2016). *Machine learning: Algorithms and applications*.

[131] Crash Course On Multi-Layer Perceptron Neural Networks. (n.d.).

[132] 1.17. Neural network models (supervised) — scikit-learn 0.24.2 documentation. (n.d.).

[133] Model Evaluation Metrics in Machine Learning - KDnuggets. (n.d.).

[134] Choosing Evaluation Metrics For Classification Model. (n.d.).

[135] Ram, A., Jalal, S., Jalal, A. S., & Kumar, M. (2010). A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases. *International Journal of Computer Applications*, *3*(6), 1–4.

[136] *Diversity Indices: Shannon and Simpson*. (n.d.).

[137] R Core Team. (2017). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria.

[138] Amgarten, D., Braga, L. P., da Silva, A. M., & Setubal, J. C. (2018). MARVEL, a tool for prediction of bacteriophage sequences in metagenomic bins. *Frontiers in Genetics*, *9*(AUG), 1–8.

[139] Fouts, D. E. (2006). Phage_Finder: Automated identification and classification of prophage regions in complete bacterial genome sequences. *Nucleic Acids Research*, *34*(20), 5839–5851.

[140] Lima-Mendez, G., Van Helden, J., Toussaint, A., & Leplae, R. (2008). Prophinder: A computational tool for prophage prediction in prokaryotic genomes. *Bioinformatics*, *24*(6), 863–865.

[141] Akhter, S., Aziz, R. K., & Edwards, R. A. (2012). PhiSpy: A novel algorithm for finding prophages in bacterial genomes that combines similarity-and composition-based strategies. *Nucleic Acids Research*, *40*(16), 1–13.

[142] Ren, J., Ahlgren, N. A., Lu, Y. Y., Fuhrman, J. A., & Sun, F. (2017). VirFinder: a novel k-mer based tool for identifying viral sequences from assembled metagenomic data. *Microbiome*, *5*(1), 69.

[143] Li, J., Tai, C., Deng, Z., Zhong, W., He, Y., & Ou, H. Y. (2018). VRprofile: gene-cluster-detection-based profiling of virulence and antibiotic resistance traits encoded within genome sequences of pathogenic bacteria. *Briefings in bioinformatics*, *19*(4), 566–574.

[144] Reis-Cunha, J. L., Bartholomeu, D. C., Manson, A. L., Earl, A. M., & Cerqueira, G. C. (2019). ProphET, prophage estimation tool: A standalone prophage sequence prediction tool with self-updating reference database. *PLoS ONE*, *14*(10), 1–11.

[145] Starikova, E. V., Tikhonova, P. O., Prianichnikov, N. A., Rands, C. M., Zdobnov, E. M., Ilina, E. N., & Govorun, V. M. (2020). Phigaro: High-throughput prophage sequence annotation. *Bioinformatics*, *36*(12), 3882–3884.

[146] Hatcher, E. L., Zhdanov, S. A., Bao, Y., Blinkova, O., Nawrocki, E. P., Ostapchuck, Y., . . . Rodney Brister, J. (2017). Virus Variation Resource-improved response to emergent viral outbreaks. *Nucleic Acids Research*, *45*(D1), D482–D490.

[147] Entrez Programming Utilities Help - NCBI Bookshelf. (n.d.).

[148] Canchaya, C., Fournous, G., & Brüssow, H. (2004). The impact of prophages on bacterial chromosomes.

[149] Campbell, A. M. [Allan M.]. (2002). Preferential Orientation Preferential Orientation of Natural Lambdoid Prophages and Bacterial Chromosome Organization. *Theoretical Population Biology*, *61*(4), 503–507.

[150] Bujak, K., Decewicz, P., Kaminski, J., & Radlinska, M. (2020). Identification, characterization, and genomic analysis of novel serratia temperate phages from a gold mine. *International Journal of Molecular Sciences*, *21*(18), 1–28.

[151] Mavrich, T. N., Casey, E., Oliveira, J., Bottacini, F., James, K., Franz, C. M., … van Sinderen, D. (2018). Characterization and induction of prophages in human gut-associated Bifidobacterium hosts. *Scientific Reports*, *8*(1), 1–17.

[152] Kogay, R., Neely, T. B., Birnbaum, D. P., Hankel, C. R., Shakya, M., & Zhaxybayeva, O. (2019). Machine-Learning Classification Suggests That Many Alphaproteobacterial Prophages May Instead Be Gene Transfer Agents. *Genome Biology and Evolution*, *11*(10), 2941–2953.

[153] Du, M. Z., Liu, S., Zeng, Z., Alemayehu, L. A., Wei, W., & Guo, F. B. (2018). Amino acid compositions contribute to the proteins' evolution under the influence of their abundances and genomic GC content. *Scientific Reports*, *8*(1), 1–9.

[154] Moura, A., Savageau, M. A., & Alves, R. (2013). Relative Amino Acid Composition Signatures of Organisms and Environments. *PLoS ONE*, *8*(10), e77319.

[155] Grigoriev, A. (1998). Analyzing genomes with cumulative skew diagrams. *Nucleic Acids Research*, *26*(10), 2286–2290.

[156] Lucks, J. B., Nelson, D. R., Kudla, G. R., & Plotkin, J. B. (2008). Genome Landscapes and Bacteriophage Codon Usage. *PLoS Comput Biol*, *4*(2), 1000001.

[157] Bahir, I., Fromer, M., Prat, Y., & Linial, M. (2009). Viral adaptation to host: a proteome-based analysis of codon usage and amino acid preferences. *Molecular Systems Biology*, *5*, 311.

[158] O'Donovan, C., Martin, M. J., Gattiker, A., Gasteiger, E., Bairoch, A., & Apweiler, R. (2002). High-quality protein knowledge resource: SWISS-PROT and TrEMBL. *Briefings in bioinformatics*, *3*(3), 275–284.

[159] Jones, P., Binns, D., Chang, H. Y., Fraser, M., Li, W., McAnulla, C., ... Hunter, S. (2014). InterProScan 5: Genome-scale protein function classification. *Bioinformatics*, *30*(9), 1236–1240.

[160] Blum, M., Chang, H.-Y., Chuguransky, S., Grego, T., Kandasaamy, S., Mitchell, A., ... Finn, R. D. (2020). The InterPro protein families and domains database: 20 years on. *Nucleic Acids Research*, (1).

[161] Brüssow, H., & Hendrix, R. W. [Roger W.]. (2002). Phage Genomics: Small is beautiful.

[162] Banks, D. J., Lei, B., & Musser, J. M. (2003). Prophage Induction and Expression of Prophage-Encoded Virulence Factors in Group A Streptococcus Serotype M3 Strain MGAS315. *Infection and Immunity*, *71*(12), 7079–7086.

[163] Afgan, E., Baker, D., Batut, B., Van Den Beek, M., Bouvier, D., Ech, M., ... Blankenberg, D. (2018). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, *46*(W1), W537–W544.

[164] PyCharm: the Python IDE for Professional Developers by JetBrains. (n.d.).

7

SUPPORT MATERIAL

7.1 DETAILS OF RESULTS

Table 25: This table shows the coordinates of each the region in the bacterial genomes predicted by the two models. These predicted regions are possible prophages and therefore are candidates to posterior analysis.

Candidate	Start	End
cand_0	0	4493
cand_10	26768	33411
cand_70	176435	192608
cand_90	225744	231038
cand_99	248655	253566
cand_130	326869	330948
cand_172	430677	457559
cand_208	521539	531747
cand_221	553440	562858
cand_257	643780	648223
cand_299	748346	753122
cand_309	772779	777231
cand_360	901286	906032

cand_381	952990	958158
cand_402	1005698	1030650
cand_419	1048008	1052217
cand_443	1108801	1113392
cand_469	1173774	1186485
cand_480	1200490	1222035
cand_500	1251588	1260798
cand_524	1310382	1331605
cand_549	1373181	1377948
cand_567	1418967	1423649
cand_585	1463055	1467377
cand_600	1491924	1505366
cand_610	1525991	1530476
cand_659	1648496	1653758
cand_686	1711154	1721296
cand_702	1756761	1761753
cand_724	1813147	1825943
cand_738	1845550	1850536
cand_762	1905891	1938858
cand_795	1988551	1993415
cand_823	2058583	2062279
cand_832	2081194	2085717
cand_856	2142229	2154370
cand_880	2202048	2207023
cand_886	2217210	2221609
cand_913	2283922	2288995
cand_957	2393646	2397884

cand_979	2446566	2453423
cand_996	2492645	2495951
cand_1003	2508595	2516741
cand_1037	2593746	2609929
cand_1072	2680635	2687598
cand_1104	2761964	2766767
cand_1136	2840340	2851344
cand_1187	2968743	2986566
cand_1253	3133434	3139308
cand_1398	3496185	3506431
cand_1413	3530422	3547906
cand_1454	3636037	3640953
cand_1471	3677759	3682339
cand_1477	3693478	3698473
cand_1507	3768525	3793157
cand_1527	3818219	3823561
cand_1568	3920943	3934890
cand_1578	3945479	3949971
cand_1609	4023164	4028647
cand_1637	4093373	4103114
cand_1651	4127799	4132888
cand_1669	4173908	4178794
cand_1746	4366597	4386413
cand_1757	4396525	4402407
cand_1801	4503724	4508657
cand_1808	4520712	4525302
cand_1840	4602919	4607420

cand_1853	4633754	4638447
cand_1870	4676105	4680697
cand_1896	4740624	4746825
cand_1913	4783549	4789182
cand_1923	4808597	4813394
cand_1957	4893050	4902960
cand_1970	4926137	4952689
cand_1985	4963529	4968468
cand_2003	5009305	5014355
cand_2014	5038175	5044635
cand_2033	5083269	5090982
cand_2052	5131369	5136433
cand_2076	5190558	5195156
cand_2087	5219297	5224972
cand_2096	5241734	5246644
cand_2105	5264064	5278117
cand_2130	5325459	5343222
cand_2158	5395323	5400338
cand_2165	5412992	5423909
cand_2180	5451348	5456268
cand_2187	5468843	5486117
cand_2199	5498390	5502924
cand_2213	5533790	5538315
cand_2224	5561750	5571488
cand_2238	5595244	5610663
cand_2249	5623695	5666281
cand_2264	5662099	5666281

Table 26: Table with the number of the candidates proteins assigned to each function in the phage lifecycle. These functions are vital in the exclusion of bad quality candidates due to assessing if the phage has known proteins that can be assign to known phage functions inside the host.

Candidate	Number of proteins associated with each lifecycle function
cand_0	undetermined: 11, biosynthesis: 4
cand_10	biosynthesis: 9, undetermined: 6
cand_70	undetermined: 28, biosynthesis: 8
cand_90	undetermined: 10, penetration: 2, assembly: 1
cand_99	undetermined: 8, integration: 1, biosynthesis: 4
cand_130	undetermined: 6, assembly: 2
cand_172	biosynthesis: 5, undetermined: 38, hypothetical: 3, integration: 2, assembly: 2, penetration: 5, lysis: 1
cand_208	biosynthesis: 5, undetermined: 13, adsorption: 1, hypothetical: 1
cand_221	undetermined: 17, biosynthesis: 1
cand_257	undetermined: 1, hypothetical: 7
cand_299	biosynthesis: 7
cand_309	undetermined: 3
cand_360	biosynthesis: 5, undetermined: 6
cand_381	undetermined: 3, biosynthesis: 1, penetration: 1
cand_402	biosynthesis: 8, undetermined: 48, assembly: 1, hypothetical: 1
cand_419	undetermined: 9, biosynthesis: 3
cand_443	undetermined: 8, biosynthesis: 3
cand_469	biosynthesis: 6, undetermined: 13, penetration: 2, integration: 2
cand_480	undetermined: 43, biosynthesis: 4, hypothetical: 4, adsorption: 1
cand_500	undetermined: 11, biosynthesis: 3
cand_524	undetermined: 34, biosynthesis: 13, hypothetical: 2, lysis: 3, penetration: 1, assembly: 2

cand_549	undetermined: 13, biosynthesis: 6
cand_567	undetermined: 8, biosynthesis: 1
cand_585	hypothetical: 1, penetration: 2, undetermined: 5, biosynthesis: 2
cand_600	undetermined: 26, integration: 6, biosynthesis: 7
cand_610	undetermined: 7
cand_659	biosynthesis: 1, undetermined: 7, hypothetical: 1
cand_686	hypothetical: 2, undetermined: 10, biosynthesis: 9
cand_702	undetermined: 8, assembly: 1
cand_724	hypothetical: 1, lysis: 1, undetermined: 17, biosynthesis: 6, adsorption: 1
cand_738	undetermined: 7, biosynthesis: 1
cand_762	undetermined: 102, integration: 4, biosynthesis: 22
cand_795	biosynthesis: 5, undetermined: 9, integration: 2
cand_823	biosynthesis: 7
cand_832	hypothetical: 1, undetermined: 1, biosynthesis: 2, lysis: 1
cand_856	undetermined: 20, biosynthesis: 5, hypothetical: 2
cand_880	biosynthesis: 2, undetermined: 9, lysis: 1, hypothetical: 1
cand_886	undetermined: 10, hypothetical: 1
cand_913	undetermined: 8, assembly: 2
cand_957	undetermined: 8
cand_979	biosynthesis: 1, undetermined: 7, integration: 1
cand_996	undetermined: 1
cand_1003	biosynthesis: 3, undetermined: 3
cand_1037	undetermined: 21, biosynthesis: 3, integration: 6, penetration: 3
cand_1072	undetermined: 16, assembly: 2, biosynthesis: 1
cand_1104	biosynthesis: 4, undetermined: 6, lysis: 1
cand_1136	undetermined: 10, biosynthesis: 3, hypothetical: 1, adsorption: 1

cand_1187	undetermined: 30, biosynthesis: 6, integration: 1
cand_1253	undetermined: 3
cand_1398	undetermined: 12, hypothetical: 2, biosynthesis: 3
cand_1413	undetermined: 21, hypothetical: 4, penetration: 2, biosynthesis: 4
cand_1454	undetermined: 12
cand_1471	undetermined: 10, biosynthesis: 2, hypothetical: 2
cand_1477	undetermined: 8, lysis: 1
cand_1507	undetermined: 29, biosynthesis: 6, integration: 1
cand_1527	biosynthesis: 2, undetermined: 4, adsorption: 1
cand_1568	undetermined: 9, biosynthesis: 7
cand_1578	undetermined: 11
cand_1609	biosynthesis: 4, undetermined: 9, lysis: 1
cand_1637	undetermined: 19, hypothetical: 2, biosynthesis: 1
cand_1651	undetermined: 9, biosynthesis: 2
cand_1669	undetermined: 9
cand_1746	undetermined: 39, hypothetical: 1, adsorption: 2, integration: 1, biosynthesis: 5
cand_1757	hypothetical: 1, undetermined: 7, biosynthesis: 3
cand_1801	undetermined: 9
cand_1808	undetermined: 9, biosynthesis: 2
cand_1840	biosynthesis: 1, undetermined: 6
cand_1853	undetermined: 6, biosynthesis: 2
cand_1870	undetermined: 7
cand_1896	undetermined: 9
cand_1913	biosynthesis: 7, undetermined: 4
cand_1923	undetermined: 5, biosynthesis: 2
cand_1957	biosynthesis: 6, undetermined: 15

cand_1970	hypothetical: 4, undetermined: 31, assembly: 1, biosynthesis: 3, integration: 2
cand_1985	undetermined: 5, biosynthesis: 5, integration: 2
cand_2003	biosynthesis: 1, undetermined: 1, penetration: 1
cand_2014	biosynthesis: 3, undetermined: 8, hypothetical: 1, integration: 1
cand_2033	undetermined: 9, integration: 1, assembly: 1, penetration: 1, biosynthesis: 5
cand_2052	undetermined: 5, integration: 1
cand_2076	undetermined: 5, biosynthesis: 1, hypothetical: 1
cand_2087	undetermined: 8, biosynthesis: 1, hypothetical: 1, adsorption: 2, integration: 3
cand_2096	hypothetical: 1, undetermined: 7, biosynthesis: 1
cand_2105	penetration: 3, integration: 5, undetermined: 18, hypothetical: 2
cand_2130	undetermined: 30, biosynthesis: 7, assembly: 1
cand_2158	undetermined: 10, biosynthesis: 1, hypothetical: 1, assembly: 1
cand_2165	biosynthesis: 7, undetermined: 17
cand_2180	hypothetical: 1, undetermined: 6, biosynthesis: 1
cand_2187	undetermined: 20, biosynthesis: 1
cand_2199	undetermined: 5, biosynthesis: 2, integration: 1
cand_2213	biosynthesis: 3, lysis: 2, undetermined: 3, integration: 1
cand_2224	undetermined: 15, biosynthesis: 5
cand_2238	undetermined: 26, hypothetical: 1, biosynthesis: 4, integration: 2, assembly: 1
cand_2249	adsorption: 1, undetermined: 48, hypothetical: 1, integration: 8, biosynthesis: 10, penetration: 3
cand_2264	undetermined: 1

Table 27: This table presents the reason or reasons of the why the candidates where excluded of the final result and which ones are staying in the final result.

Candidate	Reason
cand_0	50% undetermined e hypothetical proteins, without integration, assembly, and penetration proteins
cand_10	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_70	50% undetermined e hypothetical proteins, without integration, assembly, and penetration proteins
cand_90	Less than 3 proteinas associated with phages
cand_99	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_130	Less than 10 proteins
cand_172	
cand_208	50% undetermined e hypothetical proteins, without integration, assembly, and penetration proteins
cand_221	82% undetermined and hypothetical proteins
cand_257	82% undetermined and hypothetical proteins
cand_299	75% biosynthesis proteins, without hypothetical, undetermined and integration
cand_309	82% undetermined and hypothetical proteins
cand_360	82% undetermined and hypothetical proteins
cand_381	Less than 10 proteins
cand_402	82% undetermined and hypothetical proteins
cand_419	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_443	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins

cand_469	
cand_480	82% undetermined and hypothetical proteins
cand_500	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_524	Less than 3 proteinas associated with phages
cand_549	82% undetermined and hypothetical proteins
cand_567	82% undetermined and hypothetical proteins
cand_585	Less than 10 proteins
cand_600	Less than 3 proteinas associated with phages
cand_610	82% undetermined and hypothetical proteins
cand_659	82% undetermined and hypothetical proteins
cand_686	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_702	82% undetermined and hypothetical proteins
cand_724	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_738	82% undetermined and hypothetical proteins
cand_762	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_795	82% undetermined and hypothetical proteins
cand_823	75% biosynthesis proteins, without hypothetical, undetermined and integration
cand_832	Less than 10 proteins
cand_856	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_880	82% undetermined and hypothetical proteins
cand_886	82% undetermined and hypothetical proteins

cand_913	82% undetermined and hypothetical proteins
cand_957	82% undetermined and hypothetical proteins
cand_979	Less than 10 proteins
cand_996	82% undetermined and hypothetical proteins
cand_1003	Less than 10 proteins
cand_1037	82% undetermined and hypothetical proteins
cand_1072	82% undetermined and hypothetical proteins
cand_1104	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1136	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1187	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_1253	82% undetermined and hypothetical proteins
cand_1398	82% undetermined and hypothetical proteins
cand_1413	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_1454	82% undetermined and hypothetical proteins
cand_1471	82% undetermined and hypothetical proteins
cand_1477	82% undetermined and hypothetical proteins
cand_1507	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_1527	82% undetermined and hypothetical proteins
cand_1568	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1578	82% undetermined and hypothetical proteins
cand_1609	82% undetermined and hypothetical proteins

cand_1637	82% undetermined and hypothetical proteins
cand_1651	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1669	82% undetermined and hypothetical proteins
cand_1746	82% undetermined and hypothetical proteins
cand_1757	82% undetermined and hypothetical proteins
cand_1801	82% undetermined and hypothetical proteins
cand_1808	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1840	82% undetermined and hypothetical proteins
cand_1853	82% undetermined and hypothetical proteins
cand_1870	82% undetermined and hypothetical proteins
cand_1896	82% undetermined and hypothetical proteins
cand_1913	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_1923	82% undetermined and hypothetical proteins
cand_1957	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_1970	82% undetermined and hypothetical proteins
cand_1985	Less than 3 proteinas associated with phages
cand_2003	Less than 10 proteins
cand_2014	82% undetermined and hypothetical proteins
cand_2033	Less than 3 proteinas associated with phages
cand_2052	82% undetermined and hypothetical proteins
cand_2076	82% undetermined and hypothetical proteins
cand_2087	82% undetermined and hypothetical proteins
cand_2096	82% undetermined and hypothetical proteins

cand_2105	82% undetermined and hypothetical proteins
cand_2130	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_2158	82% undetermined and hypothetical proteins
cand_2165	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_2180	82% undetermined and hypothetical proteins
cand_2187	82% undetermined and hypothetical proteins
cand_2199	Less than 10 proteins
cand_2213	Less than 10 proteins
cand_2224	50% undetermined e hypothetical proteins, without integration, assembly and penetration proteins
cand_2238	Constitution above 90% of biosynthesis, hypothetical e undetermined and less than 7 hits in known phage proteins
cand_2249	Less than 3 proteinas associated with phages
cand_2264	82% undetermined and hypothetical proteins