INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Modélisation prédictive des interactions entre bactéries et virus bactériophages

DIOGO MANUEL CARVALHO LEITE
Agosto de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Modélisation prédictive des interactions entre bactéries et virus bactériophages

Diogo Manuel Carvalho Leite

Mémoire pour l'obtention du diplôme de Master en Ingénierie Informatique, Spécialisation en Systèmes d'Information et Connaissances

Directeur de thèse : António Constantino Lopes Martins

Co-directeur de thèse : Carlos-Andrés Peña-Reyes

Suivi de projet : Aitana Neves Da Silva

Jury :

Président :

[Nom du président, Catégorie, École]

Rapporteurs:

[Nom du rapporteur, Catégorie, École]

[Nom du rapporteur, Catégorie, École]

Porto, septembre 2016

"O homem é do tamanho do seu sonho"
"L'homme est de la taille de son rêve"

F. Pessoa

Résumé

Actuellement, il existe un grave problème de santé publique dû au fait que les bactéries développent des résistances aux antibiotiques, notamment à cause de la surconsommation d'antibiotiques. Achetés en pharmacie, consommé dans les hôpitaux ou indirectement via la nourriture que l'être humain consomme tous les jours, la consommation de ceux-ci ne cesse de s'accroitre. La phagothérapie, ou le traitement par bactériophages est une alternative prometteuse aux antibiotiques, qui consiste à utiliser des virus « mangeurs » de bactéries pour soigner diverses infections d'origine bactérienne. Cette technique de soins possède plusieurs des avantages des antibiotiques sans ses inconvénients, puisque les bactériophages sont très spécifiques et ne s'attaquent par conséquent qu'aux bactéries à l'origine de l'infection, évitant ainsi les effets secondaires dû à la consommation d'antibiotiques par exemple sur la flore intestinale. Le défi lié à cette technique consiste à identifier rapidement le ou les bactériophages capables d'attaquer une bactérie en particulier, une procédure actuellement réalisée en laboratoire en testant toutes les combinaisons possibles, ce qui est coûteux et nécessite plusieurs jours.

La solution explorée dans ce projet consiste en l'utilisation de techniques computationnelles pour prédire *in silico* si une paire bactérie-bactériophage est capable d'interagir ou pas. Parti d'une base de données contenant plus de 1'000 paires bactérie-bactériophage positives et plus de 1'000 paires négatives pour lesquelles le génome de la bactérie et du bactériophage sont connus, la procédure suivante a été mise en place:

1. Extraction de variables pour créer 19 sets de données utilisés pour entraîner les modèles d'apprentissage automatique ;
2. Sélection et entrainement des algorithmes avec un grand nombre de configurations;
3. Recours à l'approche d'agrégation de modèle pour élaborer un système de votation ;
4. Analyse des résultats.

Le modèle final qui a été développé a permis d'atteindre une performance de plus de 90% d'*accuracy*, de mesure F1, de sensibilité et de spécificité sur un set de validation (*test* set) qui n'avait jamais été utilisé ni pour l'entraînement ni pour la validation croisée. Les bons résultats permettent d'affirmer que l'utilisation de l'apprentissage automatique semble être une approche prometteuse pour répondre à ce problème.

Mots-clés : phagothérapie, apprentissage automatique

Abstract

Currently, there is a serious public health problem because bacteria develop resistance to antibiotics, particularly because of the overuse of antibiotics. Purchased in pharmacies, consumed in hospitals or indirectly via the food that humans consume daily, the consumption of these continues to increase. Phage therapy, i.e. treatment with bacteriophages, is a promising alternative to antibiotics, which involves the use of viruses, which are literally "eaters" of bacteria, to treat various infections caused by bacteria. This treatment technique has several of the advantages of antibiotics, without their drawbacks. Indeed, bacteriophages are highly specific and therefore only attack bacteria causing the infection, avoiding side effects due to antibiotics consumption, e.g. on the intestinal flora. The challenge of this technique is to quickly identify the bacteriophages that attack a particular bacterium, a procedure currently performed in laboratories by testing all possible combinations, which is expensive and requires several days.

The solution explored in this project is the use of computational techniques to predict whether a pair of bacteriophage-bacterium is able to interact or not *in silico*. For a database containing more than 1,000 positive pairs of bacteria-bacteriophage and over 1,000 negative pairs for which the genome of both the bacterium and the bacteriophage are known, the following procedure has been put in place:

1. Extraction of features to create 19 datasets used to train machine learning models;
2. Selection and training of the algorithms with a large number of configurations;
3. Use of ensemble-learning modeling approaches to develop a voting system;
4. Results analysis.

The final model that was developed has achieved a performance of more than 90% accuracy, measurement F1, sensitivity and specificity on a validation set (test set) that had never been used for training nor for cross-validation. These good results let us conclude that the use of machine learning seems to be a promising approach to address this problem.

Keywords: phage therapy, machine learning

Remerciements

Ce document représente la fin de six années passées à l'Instituto Superior de Engenharia do Porto. Ce projet n'aurait pu être réalisé sans l'aide de plusieurs personnes que je ne peux oublier de remercier.

Commençons par mes parents pour leur patience tout au long de ces années, leur aide et soutien, sans lesquels, ce parcours n'aurait pu s'achever. Merci d'avoir cru en moi.

Ma famille et cousins habitant ou ayant habité en Suisse pour la disponibilité et le soutien apportés.

Tous les amis et amies qui ont cru en moi durant ces dernières années, qui ont toujours été présents dans les bons et également dans les mauvais. Un merci tout spécial à Maria Leitão, Paulo Coelho, Tiago Oliveira, Rui Babo, Rafael Grand et Petit qui ont su me soutenir depuis le tout début.

Professeurs Carlos-Andrés Peña-Reyes et Aitana Neves da Silva pour m'avoir fait confiance et proposé le projet d'études. Pour la disponibilité et la patience démontrées, ainsi que la transmission de toutes leurs connaissances. Aussi aux nombreuses lectures, relectures et corrections apportées à ce même document. Un grand merci.

Je voudrai remercier mon professeur, Antonio Constantino Lopes Martins, pour avoir accepté de me suivre dans ce projet en tant que tuteur de thèse. Pour les conseils et impressions fournis tout au long du développement du projet, merci.

Un remerciement tout particulier, au professeur Nuno Pinto da Silva, pour toute l'aide et les recommandations apportées au début du projet. Mais surtout pour avoir été favorable à ce projet sans émettre quelconques restrictions.

Professeures Elsa Maria Gomes et Susana Claudia Araujo, pour toutes les fois où j'en ai eu besoin, merci de m'avoir expliqué et aidé à résoudre quelques-uns de problèmes liés aux maths tout le long du projet.

Ayant eu l'opportunité d'effectuer deux échanges en Suisse, je ne peux oublier de remercier toutes ces personnes que j'ai connues venant de Corée du Sud, de Thaïlande, d'Espagne, d'Italie, de Finlande, de Suède, de Colombie et d'autres pays, pour leurs sympathies ainsi que tous les moments que l'on a passé ensemble et qui ne pourrons être oubliés.

À toutes ces personnes, je voudrais exprimer ma reconnaissance et ma gratitude.

Table des matières

Liste de Figures

Liste de tableaux

Acronymes et Symboles

Liste d'acronymes

AA	Apprentissage Automatique
ACP	Analyse en Composantes Principales
ADN	Acide Désoxyribonucléique
API	*Application Programming Interface*
ARN	Acide Ribonucléique
ASC	Aire Sous la Courbe ROC
CP	Composante Principale
FN	Faux Négatifs
FP	Faux Positifs
HEIG-VD	Haute École d'Ingénierie et de Gestion du Canton de Vaud
HMMER	*Biological sequence analysis using profile hidden Markof models*
HMM	*Hidden Markov Models*
IPP	Interactions Protéines-Protéines
MVS	Machines à Vecteurs de Support
NN	Réseaux de neurones
OMS	Organisation Mondiale de la Santé
ROC	*Receiver Operating Characteristic*
SIB	*Swiss Institute of Bio-informatics*
VHC	Virus de l'Hépatite C
VN	Vrais Négatifs
VP	Vrais Positifs
VPH	Virus du Papillome

1 Introduction

Les deux principaux moyens dont dispose la médecine moderne pour sauver des vies sont la prescription d'antibiotiques face aux infections sévères et l'usage de procédures chirurgicales sous la protection d'antibiotiques (Nathan & Cars 2014). Une problématique majeure de santé publique liée aux antibiotiques vient du fait que de nombreux agents pathogènes[1] y deviennent résistants. Cette résistance est notamment due à la surconsommation d'antibiotiques (Lu & Koeris 2011) sous prescription, par injection ou tout simplement via les aliments que nous consommons tous les jours comme la viande de bœuf et le poisson (Gayet 2016). Il est par conséquent urgent de trouver une alternative aux antibiotiques pour faire face à ce problème qui, au fil du temps, devient de plus en plus préoccupant.

Une solution ayant un regain d'intérêt grandissant pour contrer la résistance aux antibiotiques consiste en la thérapie par phages (phagothérapie) (Rakhuba et al. 2010). C'est au début du XX^{ème} siècle, suite à la découverte des bactériophages (phages), littéralement des « mangeurs de bactéries », que débutent les premières expériences de traitements d'infections bactériennes à l'aide de cette thérapie (Dublanchet & Fruciano 2008). Au milieu du XX^{ème} siècle cependant, la découverte des antibiotiques et leur fabrication rapide à l'échelle industrielle permettent de traiter les infections bactériennes avec un tel succès, notamment pendant la seconde guerre mondiale, que la phagothérapie est rapidement oubliée dans les pays occidentaux. La phagothérapie a cependant continué d'être utilisée dans les pays de l'Europe de l'Est, étant donné que leurs gouvernements interdisaient toute consommation d'antibiotiques produits par l'industrie pharmaceutique d'Europe Occidentale à cause de la guerre froide. Ces derniers ont donc poursuivi les recherches et administré des phages à des millions de patients souffrant des plus diverses maladies infectieuses sans que ceux-ci ne présentent d'effets secondaires graves connus. Ce n'est que depuis le début du XXI^{ème} siècle, suite à l'émergence des bactéries multi-résistantes, que les pays Occidentaux connaissent un regain d'intérêt pour la phagothérapie (Ravat et al. 2015).

[1] Organismes capables de causer une infection/maladie (Somogyi 2012)

Alors que les antibiotiques ont souvent un large spectre, les bactériophages sont quant à eux beaucoup plus spécifiques sur leurs cibles bactériennes. Le succès de la phagothérapie repose par conséquent sur la correspondance entre une bactérie et son bactériophage (Flores et al. 2011). Actuellement, la sélection des bactériophages permettant d'infecter une bactérie pathogène est faite de façon empirique en laboratoire, ce qui est coûteux et prend du temps. Ce projet a pour but l'élaboration d'une méthodologie « *in silico* » - c'est à dire de manière computationnelle - qui puisse prédire quels phages pourront attaquer une bactérie donnée. Sur le long terme, il s'agit donc de développer un outil permettant de construire dynamiquement les réseaux d'infections bactériophages-bactéries, pour permettre aux médecins de sélectionner le plus rapidement possible un bactériophage thérapeutique pour guérir une infection bactérienne chez un patient.

1.1 Motivations

Les motivations qui ont mené au choix du thème et projet de dissertation ont été diverses. Dès la deuxième année de Bachelor, le domaine des bases de données et de l'organisation de l'information a su captiver l'attention de l'auteur. Partir d'une grande quantité de données et essayer d'en extraire de nouvelles connaissances peut en effet paraitre insolite. Plusieurs questions se posent lorsque quelqu'un se demande comment est-ce possible qu'en partant de données passées, d'une certaine façon, il soit possible de prédire le futur ? Connaître les technologies associées à ces techniques et comprendre leur fonctionnement ont été les facteurs principaux qui ont fait que l'auteur poursuive un Master au sein de ce domaine d'études. Le monde de l'informatique étant en constante évolution, le fait que ces procédés de découvertes d'informations soient récents ont également pesé lors du choix de la voie à suivre.

L'un des avantages de l'informatique est le fait de pouvoir l'appliquer dans les plus divers domaines tels que l'économie, le multimédia, le tourisme et la médecine, entre autres. Le fait de lier l'informatique à la biologie, la médecine et la santé publique ont largement motivé le choix de l'auteur pour ce projet. Par ailleurs, savoir que le projet fait partie d'une recherche médicale, dans ce cas de la phagothérapie, dont les résultats seront réellement importants et pourront être utilisés en clinique, a été déterminant pour l'auteur.

1.2 Valeur du projet

Le problème de la résistance aux antibiotiques est de plus en plus alarmant, risquant de mener l'humain à ce que (Dublanchet 2014) appelle « l'ère post-antibiotique », où de simples infections pourraient être mortelles. La phagothérapie représente à l'heure actuelle une alternative prometteuse aux antibiotiques pour lutter contre les infections bactériennes multi-résistantes. Il existe de plus en plus de financements pour la recherche sur cette thérapie, celle-ci faisant preuve d'un engouement grandissant auprès de la communauté scientifique suite aux résultats positifs des essais sur animaux et humains réalisés récemment (Matsuzaki et al. 2005; Ravat et al. 2015). La découverte de nouvelles interactions entre bactériophages et bactéries

en laboratoire est cependant coûteuse et longue. Être en mesure de prédire rapidement et à moindre coût quel bactériophage attaque quelle bactérie, tout en comprenant mieux les mécanismes moléculaires sous-jacents à ces interactions, apporterait une grande valeur pour le développement futur de la phagothérapie (Lu & Koeris 2011). La méthodologie développée ici pourra notamment être utile pour soigner les infections bactériennes désormais résistantes aux antibiotiques. Les laboratoires pharmaceutiques, les biologistes, les médecins, l'Organisation Mondiale de la Santé (OMS), l'industrie de l'eau et l'agroalimentaire font partie des segments de clients qui ont un intérêt pour ce type de thérapie (Conti 2013; Vinay et al. 2015; Labrie et al. 2010).

1.3 Objectifs proposés

Le but du projet est de prédire les interactions entre bactéries et bactériophages en se basant sur leurs génomes et en recourant aux méthodologies d'apprentissage automatique. Dans un premier temps, il a été nécessaire (1) de comprendre les concepts biologiques nécessaires pour la conception du projet, (2) analyser et interpréter les données fournies pour ensuite (3) pouvoir élaborer les sets de données qui ont été utilisés pour l'apprentissage automatique. Dans un deuxième temps, des modèles de prédiction ont été développés en utilisant plusieurs techniques de l'état de l'art et de l'apprentissage ensembliste. Finalement, les modèles ont été évalués à l'aide de plusieurs métriques de performance, dont la matrice de confusion, et leur robustesse a été analysée, et comparées entre eux de sorte à pouvoir en tirer des conclusions.

Les objectifs proposés pour l'élaboration de ce projet peuvent être synthétisés de la façon suivante :

Objectifs généraux

- Partant des données fournies, créer divers types de sets de données qui puissent être utilisés dans diverses méthodes d'apprentissage automatique ;
- En utilisant des méthodes d'apprentissage automatique, explorer diverses alternatives pour la modélisation des interactions entre bactéries et bactériophages sur la base d'informations génomiques (séquences de protéines).

Objectifs spécifiques

- Acquérir les connaissances biologiques nécessaires à la compréhension du projet tels que :
    - Savoir ce qu'est le génome, de quoi il est composé, comment il peut être interprété ;
    - Déterminer ce qu'est une protéine, sa composition, sa fonction, comprendre comment elle arrive à interagir avec d'autre protéines ;

        - Connaître ce que sont une bactérie et un bactériophage, comprendre comment ceux-ci interagissent, quelles sont leurs fonctions, la définition et le fonctionnement de la phagothérapie ;
        - Savoir interpréter des données biologiques ;
- Analyser les deux bases de données fournies et comprendre l'information qu'elles contiennent :
    - Phage_bact : contenant de l'information concernant les paires d'interactions entre bactéries et bactériophages positives et négatives ainsi que diverses informations concernant ces deux types de cellules, telles que leurs séquences génomiques et protéique ;
    - DOMINE : contenant l'information concernant les domaines d'interactions contenus dans les protéines.
- Élaborer deux types de sets de données : (1) basé sur les domaines d'interactions entre les protéines, (2) basé sur la composition des séquences protéiques des phages et bactéries ;
- Sélectionner les techniques d'apprentissage automatique qui peuvent être utilisées et les appliquer sur les sets de données précédemment créés.
    - Créer plusieurs modèles prédictifs, analyser les résultats obtenus et en tirer des conclusions ;
        - Lancer les techniques d'apprentissage automatique avec diverses configurations et analyser les résultats obtenus.

Exigences

- Le développement doit être fait en *python* étant donné que l'équipe hôte travaille essentiellement avec cette technologie ;
- Les scripts permettant de générer les sets de données ne doivent pas être dépendants d'un type de base de données et doivent pouvoir être utilisés hors ligne.

SIB | Swiss Institute of Bioinformatics Days:

- Élaboration d'un poster concernant le projet pour la conférence des SIB Days du 7 au 8 juin 2016 à Bienne (env. 350 participants).

1.4 Planification et méthodologie d'investigation

Pour effectuer le suivi du projet, un diagramme de Gantt, dont la version finale se trouve dans l'annexe nº1 « Diagramme de Gant », a été utilisé. Afin que toute l'équipe de supervision puisse suivre l'avancement du projet, le diagramme a été mis en ligne sous forme de feuille de calcul sur la plateforme GoogleDocs[2].

Les quatre premiers mois du projet, d'octobre à janvier, ont été consacrés à l'acquisition des connaissances biologiques et bio-informatiques nécessaires à la compréhension du projet ainsi

[2] Disponible sous : https://goo.gl/ib3RhV

qu'à la découverte et à l'exploration des diverses technologies qui ont été utilisées dans l'ensemble du projet. Les trois mois suivants, de mars à juin, ont été consacrés à l'analyse des données fournies ainsi qu'à la création des scripts permettant de générer les sets de données. Le mois de mai a également été consacré à l'élaboration du poster pour la conférence des SIB Days 2016 et la sélection des algorithmes d'apprentissage automatique à utiliser. Tout le mois de juin a été consacré à l'application des algorithmes d'apprentissage automatique, à l'interprétation des résultats obtenus et à la réalisation de quelques tests d'hypothèses.

Le Tableau 1 représente les activités du diagramme de Gantt avec leurs dates respectives de début et de fin (version complète dans annexe nº 1). (À l'issu de la lecture du point 1, un document a été produit contenant les concepts abordés[3]).

Tableau 1 Extrait du diagramme de Gantt

Activité	Date de début	Date de Fin	Jours
1.0 Introduction à la Biologie et à la Bio-informatique	**05.10.15**	**12.03.15**	**80**
1.1 La Cellule	05.10.15	11.10.15	7
1.2 Le dogme central et la Génomique	12.10.15	23.11.15	43
1.3 L'Expression des Gènes et la Transcriptomique	22.11.15	29.11.15	8
1.4 les Protéines et la Protéomique	30.11.15	10.12.15	11
1.5 Les Bactéries, les Virus et les Bactériophages	11.12.15	17.12.15	7
1.6 Les réseaux d'interactions en biologie	18.12.15	23.12.15	6
1.7 Tutoriels *python* + *hands-on*	26.02.16	06.03.16	5
2.0 Introduction au projet de Master	**10.12.15**	**25.02.16**	**78**
2.1 Contexte dans lequel s'inscrit le projet de Master - *grant FNS*	10.12.15	23.12.15	14
2.2 Définition du problème	04.01.16	10.01.16	7
2.3 Manières d'aborder le problème	11.01.16	18.01.16	8
2.4 Etat de l'art (lecture et rédaction)	19.01.16	21.02.16	34
2.2 Préparation présentation intermédiaire (P1.1)	22.02.16	25.02.16	4
3.0 Sets de données et *SIB Days*	**07.03.16**	**02.06.16**	**88**
3.1 Introduction Python	07.03.16	20.03.16	14
3.2 Implémentation et test variables Domaines	21.03.16	25.04.16	36
3.3 Implémentation et test variables acides aminés	26.04.16	15.05.16	20

[3] Disponible sous : https://goo.gl/SVmDYf

Activité	Date de début	Date de Fin	Jours
3.4 Développement Script MLBD[4]	16.05.16	22.05.16	7
3.6 Élaboration poster *SIB Days*	23.05.16	02.06.16	11
3.5 Génération des sets de données	01.06.16	02.06.16	2
4.0 Apprentissage automatique	**03.06.16**	**19.06.16**	**11**
4.1 Élaboration script génération des résultats	03.06.16	06.06.16	4
4.2 *SIB Days*	07.06.16	08.06.16	**2**
4.3 Analyser premiers résultats	09.06.16	19.06.16	5
4.4 Élaboration script génération des résultats deuxième série	20.06.16	25.06.16	3
4.5 Analyser deuxièmes résultats et tests d'hypothèses	26.06.16	05.07.16	10

Tel qu'expliqué dans la méthodologie de découverte de connaissances proposée par (Fayyad et al. 1996) et décrite dans le chapitre 4 (page 35), la plus grande partie du projet, points 1 à 3, a été consacrée à l'acquisition de compétences dans le champ d'étude, à l'analyse des données fournies et à l'élaboration des sets de données. Sur ces bases solides, l'application des techniques d'apprentissage automatique, interprétations et validations des résultats a ensuite pu être réalisé en un peu plus d'un mois. Le projet ayant démarré en octobre 2015 et celui-ci se terminant en Juillet 2016, la première partie, points 1 à 3, n'a été finalisée qu'en Mai 2016, représentant plus de 8 mois de travail, alors que la deuxième, point 4, n'en a nécessité que deux.

1.5 Structure du document

Cette thèse est divisée en sept chapitres. Le premier chapitre intitulé « Introduction » (page 1) contient une brève description du projet, les motivations qui ont mené au choix de ce dernier, les objectifs proposés par la Haute École d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD), la planification et méthodologie utilisée lors du développement. Le second chapitre désigné « Contextualisation et État de l'art » (page 9) aborde le problème de la résistance aux antibiotiques, la solution proposée, les connaissances biologiques et informatiques actuelles permettant d'arriver à un prototype. Le troisième chapitre nommé « Analyse de valeur » (page 29) aborde les aspects référents à la valeur du marché, au potentiel de rentabilité de la solution et au type de négociation envisageable pour le projet. Le quatrième chapitre intitulé « Design – approche théorique » (page 35) présente les résultats obtenus par d'autres auteurs ayant élaborés des solutions en relation avec le projet proposé, la méthodologie qui a été utilisée lors le développement du projet, la description des données fournies ainsi que les aspects pertinents jugés utiles pour la création des sets de données, la sélection de ceux-ci et les informations concernant les techniques d'apprentissage automatique utilisées. Le cinquième chapitre nommé « Implémentation – approche technique » (page 59) aborde les détails

[4] *Machine Learning on Big Data*

d’implémentation de tous les scripts élaborés durant le projet. Le sixième chapitre désigné « Évaluation et test » (page 91) présente les tests effectués, les résultats obtenus, tests statistiques d’hypothèses appliqués et les résultats obtenus en comparant le projet développé avec celui d’autres auteurs. Le septième et dernier chapitre de « Conclusion » (page 111) résume les aspects les plus importants du projet, les limitations de celui-ci, les résultats obtenus et décrit les futures étapes.

2 Contextualisation et État de l'art

Il est souhaitable que tout projet de recherche et développement apporte une solution à un problème. C'est pourquoi une première recherche de littérature concernant la résistance des antibiotiques a été menée afin de contextualiser le problème et déterminer son impact potentiel. Afin d'explorer techniquement la solution proposée, une recherche de littérature abordant les connaissances actuelles en biologie des phages, interactions protéine-protéines et apprentissage automatique a aussi été réalisée.

2.1 Contextualisation du problème

Pour l'Organisation Mondiale de la Santé (OMS), les problèmes de santé publique liés à la résistance aux antibiotiques sont de plus en plus alarmants et requièrent une solution le plus rapidement possible (World Heat Organization 2014). La phagothérapie représente à ce titre une alternative prometteuse. Cependant, la procédure permettant la découverte de phages thérapeutiques ciblant une bactérie donnée est longue et coûteuse. Une méthode permettant de prédire rapidement et à moindre coût les réseaux d'infection phages-bactéries est donc nécessaire.

Des études récentes ont permis de mieux comprendre les interactions entre phages et bactéries au niveau moléculaire, mettant en évidence l'importance de certaines protéines (Seed 2015; Labrie et al. 2010). Puisque les protéines sont encodées dans le génome, une approche computationnelle basée sur les génomes des phages et des bactéries pourrait par conséquent être développée pour prédire leurs interactions. Par ailleurs, étant donné la fréquence de mutations génétiques très élevée chez les bactéries (Denamur et Matic 2006) et les phages (Sanjuan et al. 2010), un outil computationnel permettrait de construire des réseaux d'infection dynamiques et à large échelle, rendant bien mieux compte des options thérapeutiques.

Le projet consiste par conséquent en l'élaboration d'un outil d'apprentissage automatique permettant la prédiction de nouvelles interactions phage-bactérie basée sur l'information génomique. Le développement de cet outil contribuera au développement de la phagothérapie, en facilitant notamment la sélection de phages thérapeutiques à partir de réseaux d'interactions bactéries-bactériophage.

Étant donnés (i) l'importance sociétale du projet (Ravat et al. 2015), (ii) les progrès récents en biologie des phages, (iii) le succès de la bio-informatique et de l'apprentissage automatique pour prédire des comportements biologiques complexes (Larranaga 2006) et (iv) l'abondance de génomes disponibles en ligne grâce au séquençage d'un grand nombre de bactéries et de phages, le projet proposé semble d'actualité, ambitieux, et réaliste.

2.2 Bio-informatique

D'après le dictionnaire Larousse, la bio-informatique est l'« application de la recherche en informatique au progrès des connaissances dans les sciences de la vie. » (Larousse 2016). Cette discipline se concentre entre autres sur le traitement et l'analyse des données génomiques, c'est-à-dire des données issues du séquençage du génome humain et animal (Claveri 2007). La bio-informatique n'est cependant pas une simple combinaison de l'informatique et de la biologie mais une discipline en soi, qui cherche notamment à comprendre le fonctionnement du génome humain, et à prédire la fonction des gènes (France Université Numérique 2005).

La bio-informatique est également impliquée dans plusieurs projets de recherche englobant la médecine personnalisée (Gerristsen et al. 2016). Alors que les traitements utilisés pour soigner les maladies sont souvent similaires d'un patient à un autre, avec éventuellement un dosage différent, la médecine personnalisée cherche à élaborer un traitement adapté au patient ou à une sous-population de patients qui présente des biomarqueurs communs, comme des mutations génétiques communes au sein de leur tumeur cancéreuse, permettant ainsi un traitement ciblé et mieux adapté (Institut national du cancer 2012). Ceci démontre que la cohabitation de l'informatique, de la bio-informatique et de la biologie devient un phénomène de plus en plus important. Le projet développé au cours de cette thèse peut être vu comme de la médecine personnalisée, puisque le traitement consistera en l'utilisation de bactériophages qui auront été sélectionnés pour combattre spécifiquement les bactéries présentes chez le patient, en se basant sur le génome de ces dernières.

2.3 État de l'Art

Une étude approfondie concernant l'état actuel de la recherche dans le domaine d'étude a été réalisée. Celle-ci contemple notamment les divers aspects jugés importants pour la contextualisation biologique et informatique. La première partie, relative à la biologie, aborde les concepts de génome, de bactéries, de bactériophages, de mécanismes d'infection, de phagothérapie ainsi que quelques expériences concernant cette dernière. La deuxième partie

met en relation la biologie et l'informatique, en décrivant la bio-informatique et l'apprentissage automatique avec ses différentes composantes telles que le set de données, l'extraction et la sélection de variables, la création de modèles, leur évaluation, ainsi que quelques exemples d'application dans le domaine d'étude.

2.3.1 Biologie

La biologie est la science qui étudies les espèces vivantes et les lois de la vie (Larouss 2016). En partant de connaissances biologiques, cette section explique le fonctionnement de la phagothérapie au niveau moléculaire : comment est-ce que les phages attaquent les bactéries, et à leur tour, comment est-ce que les bactéries se défendent ? Un aperçu des données génomiques disponibles actuellement est également présenté afin d'expliquer le processus d'obtention des données qui a été utilisé par le groupe hôte pour la création du set de données. Pour donner des réponses aux questions, il est nécessaire : (i) connaître le génome, savoir comment est-ce qu'à partir de celui-ci sont créés les protéines, (ii) savoir la différence entre un phage et bactérie, connaître leur mécanisme d'interaction, (iii) la façon que ceux-ci ont de s'attaquer et se défendre et (iv) comprendre comment est obtenue l'information avec laquelle le projet va être développé.

2.3.1.1 Génome

Tous les organismes connus jusqu'à présent contiennent du matériel génétique, soit sous la forme d'acide désoxyribonucléique (ADN) dont la structure est une hélice à double brins complémentaires l'un à l'autre et composés chacun d'une séquence de plusieurs nucléotides ayant pour base azotée une molécule parmi l'Adénine (A), la Cytosine (C), la Guanine (G) et la Thymine (T) (Petsko et al. 2008), soit sous la forme d'acide ribonucléique (ARN). L'Adénine est complémentaire à la Thymine et la Cytosine l'est à la Guanine (Figure 1). Le génome humain contient plus de 3 milliards de nucléotides (Watson & Berry 2003).

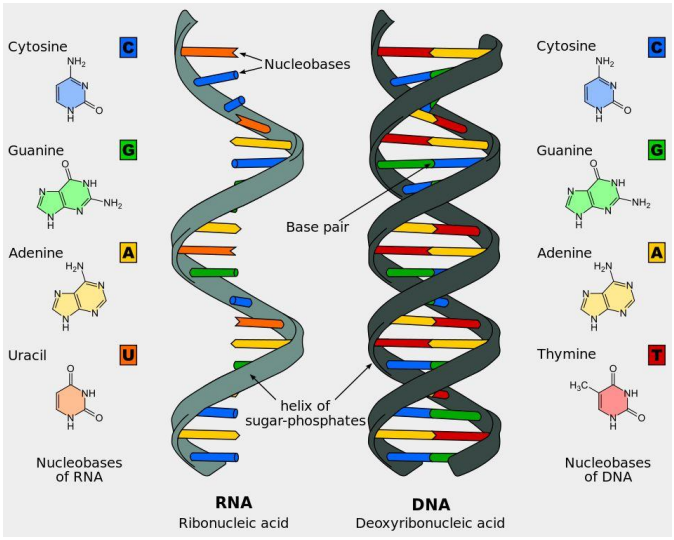


Figure 1 Séquence d'ARN (à gauche) et d'ADN (à droite) (Fikes 2014)

L'ADN dit « codant » sert à encoder les protéines de l'organisme. Le génome humain contient autour de 20'000-25'000 gènes (Johnson et al. 2011; Vermeesch & Matthijs 2014). La fabrication de protéines débute avec la transcription d'un fragment d'ADN (un gène) en ARN, par correspondance d'acides nucléiques, représenté sur la Figure 2 (Reis et al. 2015).

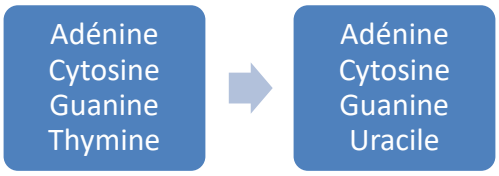


Figure 2 Correspondance ADN-ARN

Une fois le brin d'ARN créé, le processus de traduction s'initie. Celui-ci consiste en la lecture de la séquence de nucléotides par groups de trois – appelés codon. Chaque codon, représenté sur la Figure 3, représente un acide aminé. C'est la séquence d'acides aminés qui constituera une protéine (Reis et al. 2015). La Figure 4 schématise tout le processus de fabrication des protéines.

1e	2e								3e
	U		C		A		G		
U	Phe	F	Ser	S	Tyr	Y	Cys	C	U
	Phe	F	Ser	S	Tyr	Y	Cys	C	C
	Leu	L	Ser	S	stop		stop		A
	Leu	L	Ser	S	stop		Trp	W	G
C	Leu	L	Pro	P	His	H	Arg	R	U
	Leu	L	Pro	P	His	H	Arg	R	C
	Leu	L	Pro	P	Gln	Q	Arg	R	A
	Leu	L	Pro	P	Gln	Q	Arg	R	G
A	Ile	I	Thr	T	Asn	N	Ser	S	U
	Ile	I	Thr	T	Asn	N	Ser	S	C
	Ile	I	Thr	T	Lys	K	Arg	R	A
	Met	M	Thr	T	Lys	K	Arg	R	G
G	Val	V	Ala	A	Asp	D	Gly	G	U
	Val	V	Ala	A	Asp	D	Gly	G	C
	Val	V	Ala	A	Glu	E	Gly	G	A
	Val	V	Ala	A	Glu	E	Gly	G	G

signification des abréviations		
phénylalanine	Phe	F
leucine	Leu	L
isoleucine	Ile	I
méthionine	Met	M
valine	Val	V
sérine	Ser	S
proline	Pro	P
thréonine	Thr	T
alanine	Ala	A
tyrosine	Tyr	Y
histidine	His	H
glutamine	Gln	Q
asparagine	Asn	N
lysine	Lys	K
acide aspartique	Asp	D
acide glutamique	Glu	E
cystéine	Cys	C
tryptophane	Trp	W
arginine	Arg	R
glycine	Gly	G

Figure 3 Code génétique (Gallien 2005)

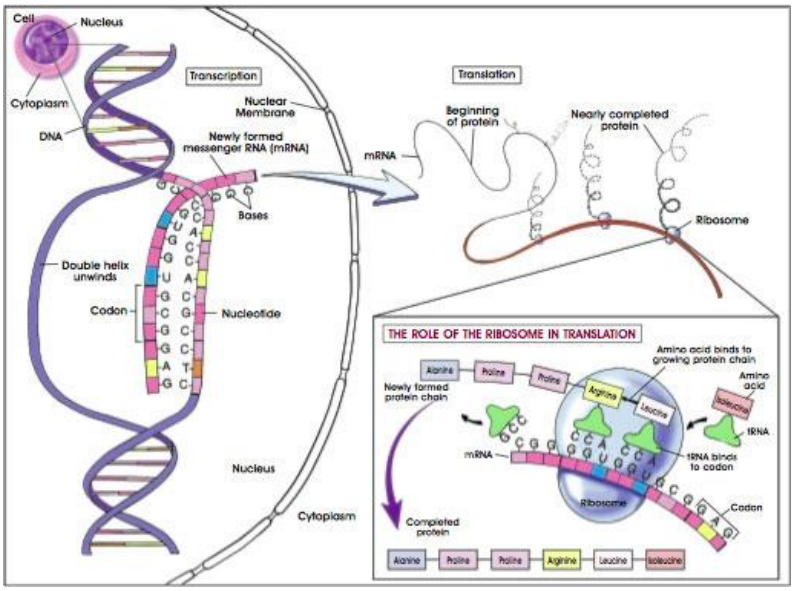


Figure 4 Fabrication d'une protéine : transcription et traduction (Donohue 2012)

2.3.1.2 Bactéries, bactériophages et interactions phage-bactérie

Les bactériophages, aussi appelés « phages », sont des virus qui ne s'attaquent qu'aux bactéries (Figure 5 et Figure 6). Leur nom signifie littéralement « mangeurs de bactéries ». Les bactéries et les phages font partie des organismes qu'il est possible de trouver en grande abondance dans les environnements naturels, anthropiques et dans le corps humain. On estime qu'il existe plus de 100 millions d'espèces de phages, mais seule une petite fraction a été séquencée jusqu'à présent (McNair et al. 2012).

Les phages ont la particularité d'être spécifiques à une espèce de bactéries. De plus, au sein de cette espèce, ils ne sont généralement capables d'infecter qu'un sous-ensemble de souches bactériennes (Flores et al. 2011). Ceci les rend d'autant plus spécifiques. Bien que de nombreuses expériences aient été effectuées pour découvrir ces interactions (Flores et al. 2011; Beckett & Williams 2013), un grand nombre d'entre elles n'ont jamais été testées expérimentalement et restent inconnues. Par ailleurs, il est important de noter que ces interactions sont dynamiques, puisque les phages et les bactéries évoluent rapidement (Sanjuán et al. 2010; Denamur & Matic 2006). Face à cette grande quantité de données dynamiques, une approche computationnelle semble la plus adaptée.

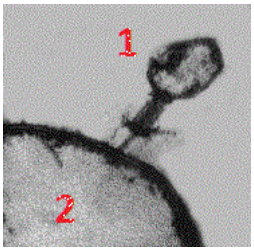


Figure 5 Phage (1) infectant une bactérie (2) – Adapté de (Collman 2003)

2.3.1.3 Mécanismes d'infection (attaque) et de défense

Afin de pouvoir prédire les interactions *in silico,* il est primordial de comprendre quels sont les mécanismes qui permettent aux bactériophages de découvrir, entrer, se lier et finalement infecter leurs bactéries hôtes, tout comme les mécanismes permettant à une bactérie de se défendre.

Afin d'infecter les bactéries, les phages disposent de protéines *receptor-binding* qui leur permettent de reconnaître les protéines réceptrices se trouvant à la surface membranaire de diverses bactéries. Après la reconnaissance, le phage se fixe à la surface de la bactérie et l'infecte en injectant son génome à l'intérieur, comme illustré sur la Figure 5 (Rakhuba et al. 2010). Dès que le génome se trouve à l'intérieur de la bactérie, celui-ci est classifié de virulent ou tempéré selon le cycle de vie qu'il suivra, voir Figure 6. Les bactériophages classifiés de virulents suivent le cycle lytique, qui consiste en la réplication du génome maintes fois pour produire de nombreux phages et, suite à la lyse cellulaire, les répandre dans l'environnement. Les bactériophages classés comme tempérés suivent le cycle lysogénique. Dans ce cas, le génome du bactériophage s'intègre à celui de la bactérie et suit le cycle de reproduction cellulaire jusqu'à ce que les conditions dans lesquelles se trouve la bactérie changent, par exemple suite à des dommages subis par la bactérie ou à des conditions de stress. C'est seulement à ce moment-là que la bactérie suivra le cycle lytique pour générer de nouveaux bactériophages. Un outil de classification déterminant le cycle de vie d'un bactériophage a été développé en se basant sur la séquence génomique du phage (McNair et al. 2012).

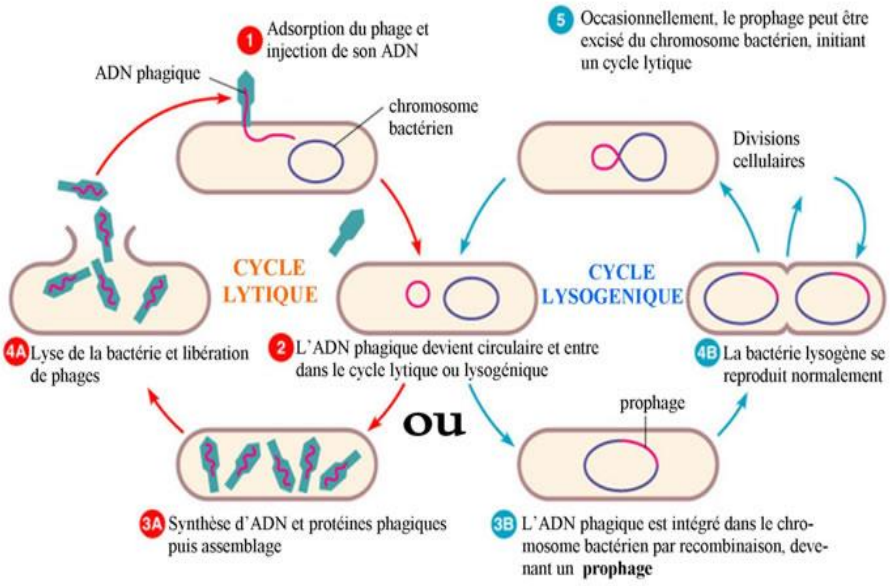


Figure 6 Cycle infectieux – Adapté et traduit de (Reece et al. 2013)

Pour chaque bactérie existante, on estime qu'il existe plus de dix bactériophages (Samson et al. 2013). Afin de se défendre, les bactéries sont en constante évolution. Cependant, les phages évoluent également rapidement afin de pouvoir accroître leur gamme de bactéries hôte (Beckett & Williams 2013).

Pour empêcher qu'un bactériophage ne se lie à ses récepteurs, la bactérie dispose de plusieurs mécanismes de défense, classés en trois catégories et illustrés sur la Figure 7: Le blocage des récepteurs de bactériophages, qui consiste à cacher, à l'aide de protéines, les récepteurs présents sur la membrane cellulaire (A) ; la production d'une matrice extracellulaire, qui consiste en la création d'une barrière servant à cacher les récepteurs (B); la production d'inhibiteurs, qui consiste en la liaison de molécules, présentes dans l'environnement bactérien, aux récepteurs, les rendant de fait indisponibles pour se lier aux bactériophages (C) (Labrie et al. 2010; Seed 2015). Par ailleurs, les bactéries ont acquis la capacité de bloquer l'injection d'ADN d'un phage qu'elles auraient déjà « rencontré » au préalable. Finalement, les bactéries disposent de plusieurs mécanismes pour couper de l'ADN reconnu comme étranger et le rendre ainsi inutile (Seed 2015).

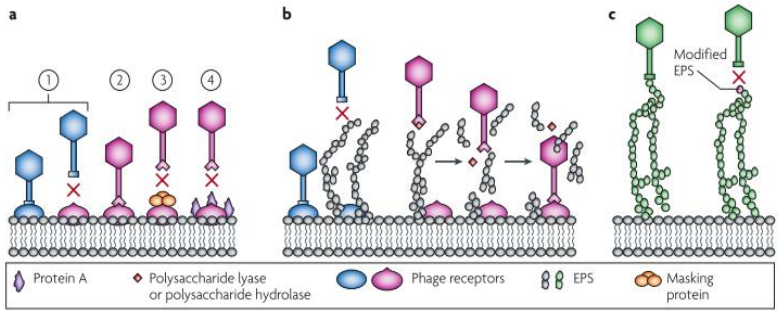


Figure 7 Méthodes de défenses des bactéries (Labrie et al. 2010)

Pour pouvoir contrer les mécanismes de résistances des bactéries, les phages évoluent leurs stratégies d'attaque grâce à des mutations ponctuelles au sein de leur génome, de réarrangements génomiques ou d'échanges génomiques avec d'autres virus afin d'obtenir de nouvelles caractéristiques (Samson et al. 2013).

2.3.1.4 Phagothérapie

Les phages peuvent être, selon la perspective sous laquelle ils sont observés, vu comme une bonne ou une mauvaise chose. Pour les professionnels de l'industrie qui travaillent avec des bactéries pour leurs procédés de fermentation, les bactériophages sont un problème étant donné qu'ils tuent les bactéries et diminuent donc la qualité de leurs produits. Dans le domaine de la médecine par contre, les phages sont vus comme un possible complément des antibiotiques, puisque les phages peuvent être utilisés pour tuer des bactéries multi-résistantes aux antibiotiques (Rakhuba et al. 2010).

La phagothérapie consiste en l'utilisation de phages afin qu'ils puissent infecter les bactéries causant des maladies et les tuer. Ce type de thérapie, contrairement aux antibiotiques, a l'avantage d'être très spécifique et de n'atteindre que les agents pathogènes pour lesquels le phage a été administré (Fischetti 2008). Néanmoins, beaucoup de travail reste à faire afin de pouvoir prédire efficacement quel phage administrer à un patient étant donné une infection bactérienne (Edgar et al. 2012). Le but de ce projet est de contribuer à l'élaboration d'algorithmes permettant de prédire *in silico* les interactions phage-bactérie.

Plusieurs expériences utilisant cette technique de soins ont déjà été menées sur des animaux dans les années 1980, avec des résultats prometteurs (Matsuzaki et al. 2005). Plus récemment, dans les années 2000, un essai clinique a été conduit chez l'humain pour les infections au niveau de l'oreille (otite), avec d'excellents résultats (Lu & Koeris 2011).

2.3.1.5 Données génomiques

Au cours des dernières années, les recherches dans le domaine de la biologie, plus concrètement des bactériophages, ont permis de mettre en évidence les principaux mécanismes moléculaires en jeu lors des interactions entre les bactériophages et les bactéries (voir section 2.3.1.3, page 14). Il est intéressant de noter qu'une grande partie de cette information est encodée dans les génomes de la bactérie et du phage, au niveau de leurs protéines. Pour pouvoir prédire les interactions à l'aide d'apprentissage automatique, il est donc important d'avoir accès aux informations génomiques concernant un grand nombre de paires phages-bactéries connues pour interagir.

Un grand nombre de laboratoires dans le monde séquencent le génome humain, et également celui de divers organismes vivants et virus tels que les phages et les bactéries. Une grande partie des laboratoires se consacrant à ce domaine publient leurs séquences en ligne dans une base de données publique nommée GenBank (Benson 2004). Celle-ci est utilisée dans le projet afin d'obtenir les génomes de différents bactériophages et bactéries pour lesquels une interaction est connue. La Figure 8 permet d'avoir une vision globale concernant la quantité de séquences disponibles dans la base de données jusqu'en 2015. La courbe rouge représente le nombre de génomes entiers, et la bleue le nombre de séquences disponibles sur GenBank (GenBank 2015).

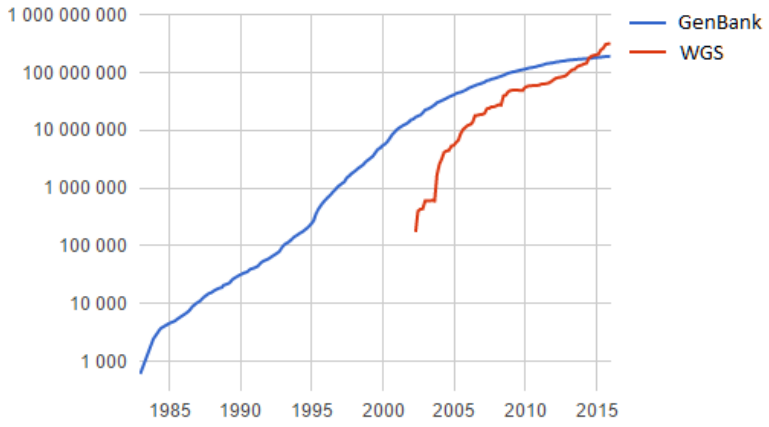


Figure 8 Données de GenBank (GenBank 2015)

Afin d'entraîner le modèle de prédiction, il sera nécessaire de disposer d'informations génomiques pour un grand nombre de paires bactériophages-bactéries. Grâce aux données collectées dans les bases de données publiques phagesdb.org et Genbank, l'équipe hôte de la HEIG-VD a pu réunir un set de données comprenant le génome, plus précisément les séquences d'acides aminés, de plus de mille paires bactériophages-bactéries connues pour interagir.

2.3.2 Apprentissage automatique

L'apprentissage automatique peut être vu comme un système informatique qui va s'améliorer avec l'expérience (Barnes 2015). Les ingénieurs spécialisés dans l'analyse de données ont développé plusieurs méthodologies, décrites dans cette section, pour permettre la création de modèles entrainés à partir d'un grand volume de données, qui sont utilisés pour la prédiction et/ou la classification de nouvelles données (Barnes 2015).

Il existe plusieurs types d'apprentissage automatique, appelées supervisées, semi-supervisées ou non supervisées, suivant que les exemples du set de données sont labellisés à priori ou non (Han et al. 2012; Kotsiantis et al. 2007). Dans le cadre du projet, des techniques non supervisées seront tout d'abord utilisées pour réduire la dimensionnalité des variables extraites à partir des génomes de phages et de bactéries, et supprimer les redondances. Par la suite, des techniques d'apprentissage supervisées seront utilisées pour prédire si oui ou non une paire bactériophage-bactérie interagit.

2.3.2.1 Apprentissage automatique non supervisé

Les algorithmes d'apprentissage automatique non supervisé ont trois principales fonctions : (i) *Clustering* – séparation, regroupement des données similaires ; (ii) Élaboration de règles d'association – détection de relations entre variables (*features*) ; (iii) Réduction de la dimensionnalité – réduction du nombre de variables utiles pour l'explorations des données (Guérif 2006; Han et al. 2012).

Les algorithmes de *clustering* les plus cités dans la littérature sont les algorithmes de partition et les algorithmes hiérarchiques (Han et al. 2012; Guérif 2006; Tan et al. 2006). Les algorithmes de partition ont pour objectif le partitionnement des objets en *n* groupes (*clusters*), de telle sorte à ce que les objets au sein de chaque *cluster* soient semblables. Afin de déterminer la ressemblance, on utilise une fonction permettant de calculer la distance entre deux objets, telle que la distance Euclidienne. L'objectif de ces algorithmes consiste en la minimisation de la distance entre tous les objets d'un même *cluster* et la maximisation de celle-ci face aux objets de tous les autres *clusters* (Fallis 2013; Witten et al. 2001). Une des méthodes permettant d'indiquer quel peut être le nombre de *clusters* idéal est la méthode « Critère d'information Bayésien ». Celle-ci part du principe que les objets ont été obtenus à partir d'un modèle et va donc, statistiquement, essayer plusieurs modèles prédéfinis en alternant la taille, la forme et la quantité de *clusters*. Le modèle avec le meilleur score permet de déterminer le nombre de clusters idéal (Fraley & Raftery 1998). Les algorithmes hiérarchiques sont divisés en deux types différents : les agglomératifs et les divisifs. Les deux nécessitent des matrices de similarité/distances. Les agglomératifs attribuent un cluster à chaque objet et successivement rassemblent les paires ayant les distances les plus courtes. Les divisifs, quant à eux, regroupent tous les objets dans un même cluster et vont successivement les diviser en clusters de plus en plus petits. Le résultat des diverses divisions ou jointures peut être représenté sous la forme d'un diagramme de Venn ou d'un dendrogramme (Witten et al. 2001; Fallis 2013).

Les algorithmes basés sur les règles d'association essayent de trouver des associations entre les divers objets. Une règle d'association est formée de la façon suivante : *X -> Y* signifie que si *X* apparaît, alors *Y* apparaît ; *X* et *Y* sont des ensembles d'une ou plusieurs entrées du set de données. Il existe trois types de règles pouvant être conclues : utiles, triviales et inexplicables. Les utiles correspondent aux règles pertinentes pour un problème ; les triviales sont celles dont l'utilisation par un algorithme n'est pas utile sachant qu'elles sont communes au problème ; les inexplicables correspondent aux règles qui n'apparaissent que peu de fois. Afin de s'assurer que l'algorithme n'identifie que des règles utiles, des mesures de support et de confiance sont utilisées. Ces dernières correspondent, pour l'exemple d'une règle du type *{A,B,F => O}*, au *nombre d'entrées contenues dans A, B, F et O* divisé par le *nombre total d'entrées* (*support*), et au *nombre d'entrées contenues dans A, B, F et O* divisé par le *nombre d'entrées dans A, B et F* (*confiance*). En imposant une confiance et un support minimum, il est possible de réduire la quantité de règles sans intérêt et celles n'apparaissant que rarement (Han et al. 2012; Tan et al. 2006).

2.3.2.2 Apprentissage automatique supervisé

Pour qu'un modèle issu d'un apprentissage automatique supervisé soit optimal, il est important que les entrées qui composent les sous-sets soient équitablement représentées par les différents attributs objectif (*label*) (Tan et al. 2006; Kotsiantis et al. 2007; Fallis 2013; Maletic & Marcus 2005). Par ailleurs, il est aussi important de définir dès le début du projet le set de test (*test set*) contenant des entrées avec lesquelles le modèle ne sera jamais entraîné, afin d'évaluer la performance du modèle une fois l'entraînement terminé. La performance sur le set de test peut être mesurée avec une matrice de confusion, qui permet notamment de calculer le rappel (*recall)*, la précision, le taux d'erreur et la mesure F1 (Chawla 2005; Kotsiantis et al. 2007), décrites plus en détail dans la section 2.3.2.7 (page 24). Dans le cas où le modèle ne répondrait pas aux exigences attendues, plusieurs points seront investigués: mauvais choix du set de données, mauvais choix dans les variables, mauvais set d'entraînement, algorithme d'entraînement pas adapté ou mauvaise configuration de celui-ci (Kotsiantis et al. 2007).

Ci-dessous sont décrits les algorithmes qui peuvent être utilisés pour entraîner un modèle à partir des données du set d'entraînement. Les algorithmes d'apprentissage supervisés les plus connus sont :

- Les arbres de décision : ceux-ci créent un modèle contenant une séquence de décisions arrangées en une structure semblable à un arbre où chaque feuille correspond à une classe – résultat possible de l'attribut objectif (*label*) –, chaque nœud à une décision à prendre - évaluation d'une variable - et chaque branche à un attribut (Han et al. 2012).
- Les réseaux de neurones : dans sa forme la plus simple, un réseau de neurones est composé de trois couches : une couche d'input, une couche « cachée » et une couche d'output. La couche cachée peut être composé d'un ou plusieurs niveaux (*deep learning*). Chaque neurone est décrit par une fonction dont les poids sont ajustés pendant l'entraînement (Han et al. 2012).
- Réseaux bayésiens : cette technique utilise des méthodes statistiques afin de déterminer quelles variables influencent d'autres variables (Tsai et al. 2009). Cet

algorithme a pour base deux hypothèses : Toutes les variables sont également importantes et les variables sont statistiquement indépendantes – la valeur d'une variable ne dit rien concernant une autre variable (Tan et al. 2006). Un exemple d'acquisition de connaissance que possède le modèle entraîné à partir de ce type d'algorithme peut être : « si quelqu'un fume, alors il aura probablement plus de chance d'avoir un cancer que celui qui ne fume pas »

- K-plus proches voisins : afin de prédire la classe d'un nouvel objet, cet algorithme commence par identifier les ***k*** entrées du set de données qui lui sont le plus semblables – voisins –, puis le classifie en accord avec la classe la plus nombreuse parmi ces voisins (Tsai et al. 2009). La valeur de ***k*** choisie ne devra pas être trop petite, pour éviter le sur-apprentissage et un manque de généralité du modèle (sensibilité au bruit). Cette valeur ne devra pas non plus être trop grande, notamment pour des raisons computationnelles (temps de calcul).
- Machines à vecteurs de support (MVS) : Cette technique cherche à maximiser la distance entre la frontière de séparation et les données les plus proches (notion de « marge maximale »). Grâce à l'utilisation de « kernels » (fonction noyau), ces méthodes peuvent également être utilisées pour identifier des frontières non-linéaires entre les classes (Han et al. 2012; Tan et al. 2006).

2.3.2.3 Techniques de réduction d'espace

Généralement, tout processus d'apprentissage automatique, indépendamment du domaine, part de grands volumes de données. La santé constitue un domaine dans lequel il existe de grands volumes de données concernant les patients. Par exemple, le système de santé américain a pratiquement atteint le zettabyte[5] (Raghupathi & Raghupathi 2014). L'institut Européen de bio-informatique (EBI) détient une base de données contenant plus de 20 pétabytes[6] d'information biologique tels que des gènes et des protéines. Indépendamment du type de données avec lesquelles on prétend travailler, il est souvent nécessaire de réduire la taille de données tout en essayant de maintenir un maximum d'informations (Han et al. 2012), sans quoi le traitement de telles quantités d'informations pourrait s'avérer extrêmement long, de l'ordre de plusieurs heures à plusieurs jours voir même des semaines selon la quantité d'informations à traiter. Quelques-unes des méthodologies vues au long de l'étude de l'état de l'art sont :

- L'analyse des composantes principales (ACP) est une technique qui partant de points dans un espace à ***N*** dimensions représentés dans un système axial prédéfini, par exemple : X, Y, Z, va chercher ***M (≤N)*** d'autres variables, appelées composantes principales (CP), qui visent à représenter au maximum l'information originale dans un système axial ayant subi une ou plusieurs transformations (Tuffery 2012). Les données sont alors projetées dans le nouveau système axial réduisant le nombre de dimensions et, par conséquent, l'espace. Les CP sont ordonnées de celle qui restitue le plus

[5] 1 Zettabite = 10^{12} gigabits (Raghupathi & Raghupathi 2014)
[6] 1 Petabite = 10^{6} gigabites (Marx 2013)

d'information à celle qui en restitue le moins où pour chacune d'elles il est possible de savoir le pourcentage d'information restitué (Han et al. 2012; Tuffery 2012). La Figure 9 illustre un exemple d'application d'ACP: à gauche sont représentées des données dans un système à trois dimensions sur lesquelles est appliqué une réduction avec deux CP illustrées sur la partie de droite de la figure. Toutes les données sont alors "écrasées" de sorte à être représentées dans ce nouveau système axial. La ACP fait partie des techniques de réductions de dimensionnalités non-supervisées, elle n'utilise donc pas l'attribut classe des données et part du principe que celles-ci n'appartiennent à aucune classe (Tollari 2006).

- L'analyse discriminante linéaire (LDA) vise à réduire le nombre de dimensions tout en maintenant un maximum d'informations. Contrairement à l'ACP, l'LDA tient en compte les classes auxquelles appartiennent les différents registres. Partant d'un set de données à ***N*** dimensions, celle-ci va chercher ***M*** (***≤N***) dimensions de sorte à ce que les projections de données permettent une grande séparation des classes (Tollari 2006). Sur la Figure 10 est illustrée la différence des axes choisis selon la technique utilisée. La LDA est utile particulièrement dans les cas où les fréquences interclasses sont inégales.

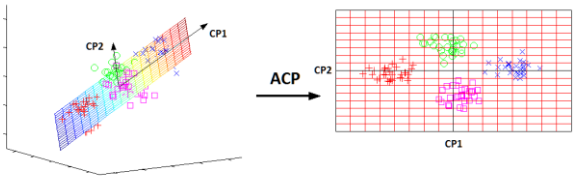


Figure 9 Exemple ACP (3 dimensions -> 2 dimensions) – adaptée de (Scholz 2006)

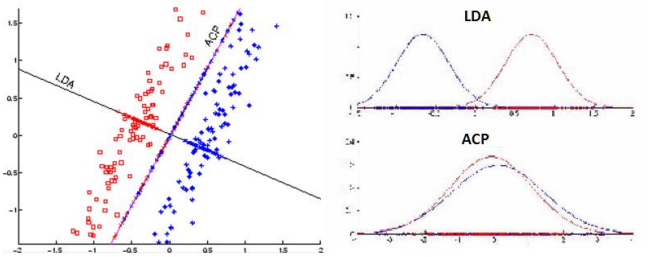


Figure 10 LDA et ACP – adaptées de (Tollari 2006)

2.3.2.4 Agrégation de modèles

Pour obtenir les meilleures performances possibles, une approche d'agrégation de modèles sera utilisée pour la création du modèle final du projet (voir section 4.5, page 55). Ce concept permet la combinaison de plusieurs algorithmes de classification, augmentant ainsi généralement les performances obtenues comparé à l'utilisation d'un unique algorithme (Polikar 2014). La Figure 11 illustre le procédé utilisé dans cette approche.

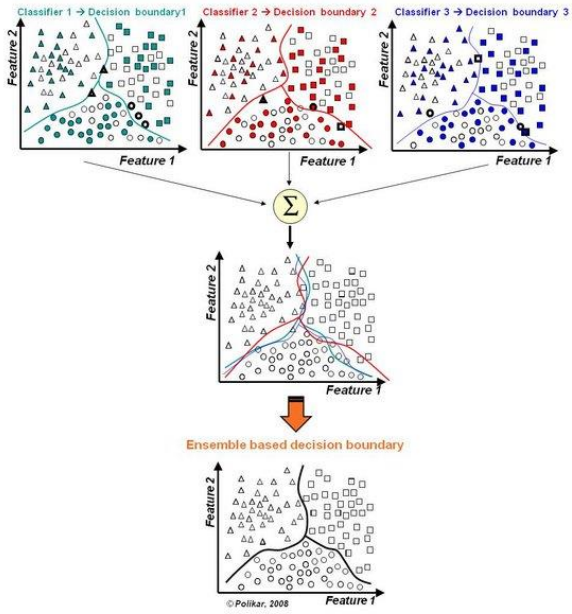


Figure 11 Combinaison de classificateurs (Rodrigues 2014)

Bagging, *Boosting* et Forêt aléatoires sont les trois méthodologies basées sur l'approche d'agrégation de modèles les plus citées dans la littérature (Han et al. 2012; Maletic & Marcus 2005; Fallis 2013) :

- L'approche *Bagging* consiste en la création de ***n*** sets d'entraînement élaborés avec l'approche *Bootstrap* décrite sous la section 2.3.2.5 (page 22), chacun de même taille que le set de données original. Chacun de ces sets d'entraînement est utilisé pour entraîner un modèle à l'aide d'algorithmes de classification. Lors du test, le *bagging* utilise chaque modèle pour prédire la classe d'une entrée, en procédant à un vote de poids égal entre tous les modèles (Han et al. 2012; Maletic & Marcus 2005).
- L'approche *Boosting* est très similaire au *bagging*, dans la mesure où un ensemble de classificateurs est généré en échantillonnant le set de données. Initialement, chaque entrée a les mêmes chances d'être choisie pour entraîner le modèle. Cependant, après qu'un modèle ***Mi*** ait été entraîné, le poids des entrées pour générer un nouveau set va être adapté de sorte à ce que le modèle ***Mi + 1*** donne plus d'importance aux entrées mal classifiées (Han et al. 2012). Contrairement au *Bagging*, le poids attribué à chaque modèle lors du vote est proportionnel à la confiance (*accuracy*) du modèle (Han et al. 2012; Maletic & Marcus 2005; Witten et al. 2001).
- Forêt aléatoire consiste en l'utilisation de multiples arbres de décision pour la classification. Les sets d'entraînement sont sélectionnés par *Bootstrap*. Par ailleurs, chaque arbre est créé en utilisant uniquement ***n*** variables choisies aléatoirement, tel que ***n*** soit beaucoup plus petit que le nombre total de variables du set de données. Lors de l'entraînement, la classe d'une entrée est attribuée par votation entre les arbres (Han et al. 2012; Witten et al. 2001).

2.3.2.5 Base de données et set de données

Avant de pouvoir passer à la création d'un modèle, la constitution d'un set de données est une étape cruciale qui doit être faite avec le plus grand soin pour que le futur modèle soit le plus général possible (Witten et al. 2001; Fayyad et al. 1996). Comme décrit plus haut sous la section 1.3 (page 3), le groupe hôte a déjà constitué une base de données qui pourra être utilisé pour le projet proposé pour générer des *sets* de données utilisables par des algorithmes d'apprentissage automatique. La base de données fournie est composée de +1000 exemples positifs de paires bactéries-bactériophages dont on est sûr qu'il existe une interaction. Pour chaque paire, les génomes de la bactérie et du phage (séquences de protéines) sont disponibles dans la base de données du groupe hôte.

Dans le domaine de l'apprentissage automatique, le set de données utilisé doit, si possible, être équitablement composé d'exemples de toutes les classes (Han et al. 2012). Pour ce projet, l'équipe hôte a donc rajouté à sa base de données « positifs » des paires phages-bactéries choisies au hasard, à condition que celles-ci ne se trouvent pas dans l'ensemble des entrées phages-bactéries positives. En effet, la probabilité de sélectionner, par erreur, une paire positive comme négative est faible et acceptable (Ben-Hur & Noble 2006).

A partir des séquences de protéines de cette base de données, un travail important a consisté en l'extraction de variables ("*features*") pour générer des sets de données quantitatifs utilisables pour de l'apprentissage automatique.

La sélection de la méthode pour partitionner le set de données en un set d'entraînement et de test est aussi cruciale, car c'est sur ces données que le modèle va apprendre et être testé. Quelques-unes des méthodes les plus utilisées sont le *hold-out*, la validation croisée, le *leave-one-out* et le *bootstrap*.

Le *hold-out* vise à ce que l'échantillonnage du set d'entraînement corresponde à ~70% du set de données, laissant ~30% de données pour le set de validation (ces pourcentages sont les plus cités dans la littérature (Maletic & Marcus 2005; Han et al. 2012; Fallis 2013)). Il est important de noter que même si les sets d'entraînement et de test sont créés de manière aléatoire, l'algorithme s'assure qu'ils soient stratifiés et donc représentatifs du set de données original (Maletic & Marcus 2005; Han et al. 2012; Fallis 2013). Dans notre cas, le set de données sera composé de 50% de paires ayant une interaction et de 50% de paires sans interaction connue.

L'approche de validation croisée consiste en la division du set de données original en ***k*** sous-ensembles mutuellement exclusifs, de taille égale, définis de façon aléatoire et, dans le cadre du projet, stratifiés. Le nombre d'expériences est ici dicté par le nombre ***k*** (la littérature fait référence à ***k = 10*** pour les set de données stratifiés (Han et al. 2012)). A chaque expérience, un des ***k*** sous-ensembles est utilisé pour valider le modèle entraîné à partir des ***k-1*** autres sous-ensembles. Il existe un cas particulier, appelé *leave-one-out*, dans lequel ***k*** = nombre d'entrées – 1. Dans ce cas, l'entrée restante étant utilisée pour valider le modèle, ce qui permet de couvrir au maximum les données utilisées pour l'entraînement du modèle (Han et al. 2012; Fallis 2013; Maletic & Marcus 2005).

La méthodologie *bootstrap* consiste en la création d'un set d'entraînement avec repositionnement, c'est-à-dire qu'il sélectionne ***n*** entrées du set de données original et que ceux-ci peuvent apparaître plus d'une fois dans le set d'entraînement. Statistiquement, (Han et al. 2012; Fallis 2013; Witten et al. 2001) ont démontré que la probabilité qu'une entrée soit sélectionnée est de ~63.2%. Le set de validation est composé des entrées qui n'ont pas été sélectionnées dans le set d'entraînement, correspondant aux ~36.8% restants. Le processus est répété ***n*** fois.

2.3.2.6 Sélection de variables

La sélection de variables extraites des entrées de la base de données est une étape importante dans l'apprentissage automatique. Il existe en effet plusieurs facteurs qui, s'ils ne sont pas pris en compte, peuvent conduire à la création d'un mauvais modèle ou augmenter le temps d'entraînement de ce dernier. Ces facteurs peuvent être la présence de valeurs manquantes, de valeurs inconsistantes, de valeurs à double ou de valeurs isolées (Tan et al. 2006). Un autre facteur, moins visible, mais qui peut également affecter la création de modèles, est le gain d'information que possède une variable (Piramuthu 2004). C'est tout l'enjeu du projet : identifier des variables qui contiennent suffisamment d'information pour pouvoir prédire les interactions phage-bactérie.

Afin d'y parvenir, nous étudierons des variables qui mettent en relation les protéines de la bactérie avec celles du phage (interactions protéine-protéine, IPP). En effet, lorsqu'un phage interagit avec une bactérie, les interactions entre ces paires se font au niveau des protéines, qui sont elles-mêmes encodées dans leurs génomes. Par conséquent, il est probable que le nombre d'IPP entre les protéines d'un phage et les protéines d'une bactérie infectée par ce phage soit élevé.

Afin de prédire si un phage infecte une bactérie, il faudra donc utiliser des variables présentes dans leurs génomes respectifs. En particulier, nous nous baserons sur les variables qui ont déjà été utilisées dans des modèles existants pour prédire les IPP (Coelho et al. 2014; Nourani et al. 2015; Dyer et al. 2007), puisque l'hypothèse de départ est que les IPPs seront un facteur déterminant pour découvrir des paires d'interaction phage-bactéries. Parmi ces variables, on trouve notamment la similarité des séquences protéiques (distance entre les IPP) et des mesures d'ontologie obtenues grâces aux ontologies génomiques tels que *Gene Ontology* (Coelho et al. 2014). Cependant, il est à noter que notre modèle ne sera pas en mesure de prédire les IPP, car notre base de données ne contient pas ce type d'information, mais uniquement si une paire phage-bactérie interagit ou non. La performance du modèle qui sera développé ne pourra donc pas être comparée à ces modèles existants de prédiction d'IPP entre des organismes dont on sait qu'ils interagissent, puisque le but de notre modèle est précisément de découvrir si une paire d'organismes (phage-bactérie) interagit ou non. Il s'agira donc de prédictions d'interactions au niveau des organismes, et non de leurs protéines, même si les variables utilisées se baseront sur les séquences de protéines.

Pour une paire phage-bactérie, les IPPs seront cependant très nombreuses, du fait du grand nombre d'IPPs potentielles : nombre de protéines bactérie *x* nombre de protéines phage[7] ≈ $+2.6*10^5$. Il faudra donc commencer par réduire la dimensionnalité pour obtenir un plus petit nombre de variables informatives pour chaque paire phage-bactérie. Une grande partie du projet de Master a été consacré à cet aspect.

2.3.2.7 Mesures de performances

Indépendamment du type d'algorithme utilisé pour entraîner un modèle, il est nécessaire que celui-ci soit évalué afin de déterminer sa performance. L'évaluation d'un modèle est basée sur la capacité que celui-ci a à prédire un registre face à l'attribut objectif, qui dans le cadre du projet consiste en prédire l'existence ou pas d'une interaction entre un phage et une bactérie.

Les mesures de performances ont pour base la matrice de confusion qui fournit l'information concernant les prédictions faites par le modèle (Maletic & Marcus 2005). Le Tableau 2 représente la matrice de confusion. Un **vrai positif** (VP) signifie que le modèle a bien classé une vraie interaction comme étant vraie. Le **vrai négatif** (VN) indique que le modèle arrive à bien classer une interaction qui n'existe pas comme n'existant pas. Le **faux positif** (FP) indique le nombre de fois que le modèle a classé une interaction existante alors que celle-ci n'existe pas. Les **faux négatifs** (FN) donnent connaissance du nombre de paires bactéries-bactériophage que le modèle a indiqué ne pas interagir alors que celles-ci interagissent. Il est possible d'associer des coûts aux mauvaises prédictions (Maletic & Marcus 2005) indiquant, parmi les FN et FP, lesquelles sont considérées comme les plus graves. Dans le cadre du projet, un faux positif sera considéré moins grave qu'un faux négatif. En effet, les phages prédits seront ensuite validés expérimentalement afin d'être administrés chez un patient et il convient donc de ne pas être trop restrictif sur les candidats à tester.

Tableau 2 Matrice de confusion – Adapté de (Maletic & Marcus 2005)

		Classe prévue	
		Interaction	*Pas interaction*
Classe réel	*Interaction*	Vrais positifs (VP)	Faux négatifs (FN)
	Pas interaction	Faux positifs (FP)	Vrais négatifs (VN)

En se basant sur la matrice de confusion, la performance du classificateur peut être évaluée à l'aide des mesures présentées dans le Tableau 3.

[7] Dans la base de données fournie, chaque bactérie et bactériophage comporte, en moyenne, 3'500 et 75 protéines, respectivement.

Tableau 3 Mesures d'évaluations – Adapté de (Han et al. 2012; Fallis 2013; Maletic & Marcus 2005)

Mesure	Formule	Description
Accuracy	$\frac{VP + VN}{VP + VN + FP + FN}$	Précision élevée permet de dire que le modèle arrive à bien classifier les exemples
Taux d'erreur	$1 - \frac{VP + VN}{VP + VN + FP + FN}$	Taux d'erreur bas permet de dire que le modèle arrive à bien classifier les exemples. Opposé de la précision
Sensibilité, Rappel, taux de vrais positifs	$\frac{VP}{VP + FN}$	Capacité du modèle à prédire les vrais positifs
Spécificité, taux de vrais négatifs	$\frac{VN}{VN + FP}$	Capacité du modèle à prédire les vrais négatifs
Précision	$\frac{VP}{VP + FN}$	Résultat élevé permet d'affirmer que le modèle arrive à bien classer les vrais positifs
Taux de faux positifs	$\frac{FP}{FP + VN}$	Erreur de prédiction pour la classe de vrais négatifs
Taux de faux négatifs	$\frac{FN}{FN + VP}$	Erreur de prédiction pour la classe de vrais positifs
Mesure F1	$\frac{2 * VP}{2 * VP + FP + FN}$	Moyenne harmonique entre le rappel et la précision

Un taux de vrais positifs élevé permet de dire que les objets de la classe « interaction » ont été bien classifiés, mais ne dit rien concernant les objets qui ont été mal classifiés. Le rappel donne une indication similaire, sans permettre de savoir combien d'exemples ont été mal classifiés. La mesure F1 permet d'obtenir une moyenne harmonique de ces deux valeurs (Han et al. 2012).

La courbe *Received Operating Characteristic* (ROC) est une mesure largement utilisée dans le domaine, qui consiste à élaborer une courbe entre le taux de vrais positifs et le taux de faux positifs suivant le seuil de discrimination choisi pour classifier les exemples. Elle est essentiellement utilisée pour la comparaison de modèles, le meilleur étant celui qui possède l'aire sous la courbe (ASC) la plus grande (Han et al. 2012; Maletic & Marcus 2005; Fallis 2013). La Figure 12 illustre un exemple de courbe ROC. La ligne bleue représente un classificateur aléatoire, tandis que le point rouge correspond à un modèle idéal.

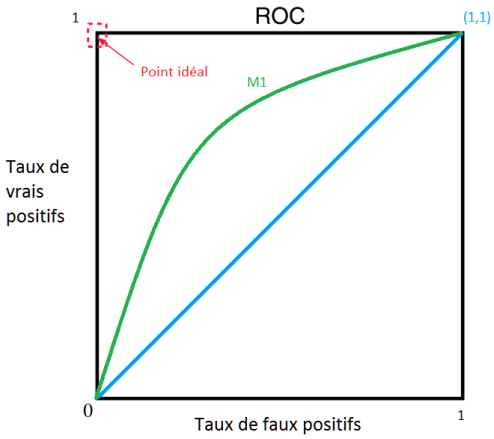


Figure 12 Exemple de courbe ROC – Adapté de (Maletic & Marcus 2005)

2.3.2.8 Niveau d'abstraction des algorithmes d'apprentissage automatique

Lors de la sélection d'algorithmes d'apprentissage automatique, un facteur à prendre en considération est le degré d'abstraction de l'information qui indique comment est-ce que les *inputs* arrivent aux *outputs*. Il existe plusieurs niveaux d'abstraction. Dans le long terme, pour pouvoir valider l'hypothèse que l'information contenue dans les IPP est suffisante pour prédire les interactions phage-bactéries, seront utilisés tout d'abord des algorithmes possédant un niveau élevé d'abstraction tels que les réseaux de neurones et les MVS. Une deuxième étape du projet consistera en l'utilisation d'algorithmes ayant un niveau d'abstraction plus bas, tels que les arbres de décisions ou les réseaux bayésiens (Kotsiantis et al. 2007), afin de mieux comprendre quelles sont les variables importantes pour la prédiction. Cette information est particulièrement importante lorsque l'on travaille avec des médecins, puisqu'ils ne peuvent pas se permettre de prendre des décisions « à l'aveugle ». Le Tableau 4 illustre les caractéristiques des divers types d'algorithmes supervisés sous forme d'un tableau de comparaison où il est possible d'observer leurs performances face à divers critères.

Tableau 4 Comparaison de divers algorithmes d'apprentissage – Traduit et adapté de (Kotsiantis et al. 2007)

	Arbre de décision	**Réseaux de neurones**	**Réseaux Bayésiens**	**K plus proches voisins**	**Machine à vecteurs de supports**
Précision générale	**	***	*	**	****
Vitesse d'entrainement pour un même set de données	***	*	***	****	*
Vitesse de classification	****	****	****	*	****
Tolérance aux valeurs manquantes	***	*	****	*	**

	Arbre de décision	Réseaux de neurones	Réseaux Bayésiens	K plus proches voisins	Machine à vecteurs de supports
Tolérance aux attributs interdépendants	***	*	**	**	****
Tolérance aux attributs redondants	**	**	*	**	***
Tolérance au bruit	**	***	***	****	**
Capacité explicative	***	*	****	**	*

2.3.2.9 Utilisation d'apprentissage automatique en biologie

L'apprentissage automatique a déjà fait ses preuves dans les plus divers domaines dont la biologie, notamment dans la résolution de problèmes spécifiques au monde de la génomique (étude du génome), de la protéomique (étude des protéines), et de l'analyse d'images de cellules ou de tissus (Larranaga 2006). La quantité grandissante d'informations biologiques requiert en effet l'utilisation de ce type de technologies afin de trouver des réponses aux questions scientifiques. L'apprentissage automatique a été particulièrement utilisé pour accélérer la découverte de biomarqueurs basés sur des données génomiques et protéomiques. Ceci a notamment permis de découvrir les mécanismes biologiques liés aux maladies neurodégénératives telles que Alzheimer (Phan et al. 2006), Parkinson (Zeng et al. 2009) ou Huntington[8] (Underwood et al. 2006), pour citer quelques exemples parmi tant d'autres. Dans le cadre de ce projet, l'apprentissage automatique est utilisé pour identifier quel bactériophage est capable d'infecter quelle(s) bactérie(s).

[8] Maladie qui se caractérise par la perte de coordination moteur et altération psychiatriques (Gil-Mohapel & Rego 2011)

3 Analyse de valeur

Ce chapitre vise à définir ce qu'est une proposition de valeur, son importance et son utilité. Le projet développé faisant partie d'un service qui pourrait être utilisé à des fins commerciales, l'auteur a élaboré une analyse de valeur et une méthodologie de négociation, créé un modèle CANVAS et décrit un possible scénario de négociation.

3.1 Proposition de valeur - définition

Tout projet, produit ou service développé doit avoir une certaine valeur pour les clients finaux, sur laquelle le produit ou service repose (Gava 2001). Les organisations ont fini par accepter le fait que ce soit le client qui fixe la valeur du produit ou service. La valeur n'est plus seulement considérée comme une somme que le client est prêt à mettre pour obtenir un bien, mais est perçue comme la différence entre les bénéfices et sacrifices tangibles ou intangibles, c'est-à-dire la valeur perçue par le client (Brei & Rossi 2005; Dominguez 2000). La tangibilité, c'est ce qui différencie principalement un service d'un produit. Dans le cadre de ce projet, c'est donc d'un service qu'il est question. La valeur que perçoit un client lorsqu'il considère acquérir un nouveau service est influencée par les bénéfices suivants (Gabriel 2010): fonctionnels, sociaux, expérimentaux et personnels ; ainsi que par les sacrifices suivants : monétaires, temporels, psychologiques et comportementaux (Campo 2004). La proposition de valeur que procure une organisation doit renvoyer l'idée que les bénéfices dont le client va tirer parti sont supérieurs à ce qu'il est prêt à investir et à ses sacrifices (Dauzat 2016).

3.2 Valeur du marché actuel

L'intérêt pour le domaine de la phagothérapie est de plus en plus grand. Le nombre d'articles publiés sur cette thérapie est en constante augmentation et à même dupliqué ces 10 dernières

années (Ravat et al. 2015). Trouver une alternative face à la problématique de la résistance aux antibiotiques (et aux antimicrobiens en général) devient de plus en plus urgent. Il est estimé que, si d'ici 2050 aucune solution n'est trouvée, plus de 10 millions de vies pourraient être perdues des suites d'infections non traitées (O'Neill 2014). Le marché des antibiotiques vaut actuellement plus de 40 milliards de dollars (Bohineust 2011). Sachant que le problème de résistance est en augmentation, il existe donc un grand engouement pour trouver un substitut (Bohineust 2011).

Les clients directement intéressés par le service qui sera développé sont les médecins et biologistes. Ce projet est donc développé en collaboration avec un biologiste et un médecin, afin de s'assurer du bon déroulement et maximiser l'impact final. De potentiels autres clients directs peuvent être les firmes pharmaceutiques, même si celles-ci attendent une modification de la règlementation (*Food and Drug Administrations* aux Etats-Unis, *European Medicines Agency* en Europe) afin de pouvoir déposer des patentes sur les phages et les destiner à la phagothérapie (Conti 2013). Les industries liées à la protection de l'eau constituent également un segment de clients intéressants, puisque les phages peuvent être utilisés afin d'identifier des eaux contaminées et ainsi protéger les populations concernées (Vinay et al. 2015). Les industries de l'alimentation qui font appel aux procédés alimentaires nécessitant des bactéries tels que la fermentation du lait, ou l'industrie liée aux produit chimiques tels que les solvants ou insecticides qui travaillent également avec des bactéries, sont autant de clients potentiels (Rakhuba et al. 2010). Pour conclure, un dernier éventail de clients intéressants se trouve dans l'agroalimentaire, pour protéger les consommateurs d'infections par les plus diverses bactéries existantes, telle que la *Listeria*[9] (Ducoeurjoly 2010).

Il est nécessaire de tenir en compte que la phagothérapie, même si elle est exploitée depuis des décennies en Europe de l'Est, n'en est qu'à ses débuts du côté Occidental (Rakhuba et al. 2010), notamment en termes de règlementations. Cette thérapie est encore récente et le prix pour découvrir de nouvelles paires d'interactions bactériophage-bactérie *in vivo* est élevé et nécessite de longues expériences de validation (Coulon 2015). C'est sur ce dernier point que repose la proposition de valeur du projet – découvrir de nouvelles paires *in silico* permettant ainsi la réduction des coûts de cette thérapie –, en identifiant de meilleurs candidats potentiels dès le départ.

3.3 Scénarios d'affaire

Une fois la proposition de valeur et les clients définis, débutera le processus de négociation avec ces derniers. D'après (Dupont & Audebert 1994), « La négociation est l'activité mettant en présence deux ou plusieurs parties (individus, groupes, délégations) qui, en raison de leur interdépendance, veulent trouver une issue satisfaisante et non violente à une situation exigeant, de la part de chacun, la prise en compte de la réalité de l'autre ». Pour qu'une

[9] Listériose : bactérie pathogène capable, chez les femmes enceintes, de traverser le placenta infectant ainsi le fœtus pouvant le tuer. Peut également être fatal pour tout être humain. (Denis 2002).

négociation ait lieu, les critères suivants doivent être observés : l'existence de dialogue, la présence d'un problème à résoudre et la recherche d'un accord mutuellement acceptable par toutes les parties (Stimec 2011). Au moment du dialogue, plusieurs facteurs font que chacune des parties s'influence mutuellement. D'après une étude menée par (Audebert 2005) auprès de dirigeants et cadres d'entreprise ainsi que d'étudiants, quelques-uns des facteurs cités sont : le bluff, la rapidité de réaction et la personnalité des intervenants. La liste complète peut être consultée dans le document d'(Audebert 2005).

Lors de la négociation, tout intervenant doit avoir un esprit de vainqueur. Il existe deux manières de gagner une négociation : soit tous les intervenants en sortent gagnant-gagnant et les intérêts de tous ont été satisfaits, soit l'une partie a satisfait ses intérêts au détriment des autres (Nicola et al. 2014). Selon l'adaptation de (Nicola et al. 2014), une négociation à quatre sorties possibles : perdant-perdant et donc pas d'accord ; gagnant-perdant – l'une des deux parties en ressort perdante ; gagnant-gagnant – l'échange de bénéfices entre les deux parties est tel qu'elles en ressortent toutes gagnantes. Une négociation peut être classifié d'intégrative dans le cas où il s'agit d'un échange gagnant-gagnant et distributive si le résultat est gagnant-perdant (Alfredson & Hopkins 2007).

Dans le cadre du projet, le type de négociation prévue est du type gagnant-gagnant – intégrative. Le groupe de recherche hôte apportera les connaissances informatiques pour la création d'un modèle de prédiction, et les médecins et biologistes contribueront avec leurs connaissances en biologie et clinique nécessaires à la création d'un modèle avec une forte valeur ajoutée. Actuellement, plusieurs idées sont en discussion concernant la mise en œuvre du service, dont la création d'une plateforme web où les clients paieraient pour accéder au service, ou la vente de celui-ci à une entité professionnelle tel qu'un laboratoire ou hôpital.

3.4 Modèle d'affaire CANVAS

Afin de comprendre la logique du modèle CANVAS, il est nécessaire de savoir ce qu'est un modèle d'affaire. D'après (Osterwalder & Pigneur 2011)**,** « Un modèle d'affaire décrit la logique de création, d'offre et de capture de valeur de la part d'une organisation ». Le modèle CANVAS aide l'entrepreneur à la création d'un modèle d'affaire pour tout type de projet auquel il prétend se lancer. Ce modèle est l'un des plus utilisés dans le monde, ayant été élaboré avec plus de 470 personnes de 45 nationalités différentes (Osterwalder & Pigneur 2011).

Le modèle CANVAS, dans sa version actuelle élaborée par (Osterwalder & Pigneur 2011), est divisé en neuf blocs répartis sur quatre perspectives illustrées dans le Tableau 5. Des études par les auteurs du modèle ont démontré que remplir le modèle à la main en recourant aux *Post-it* permet une meilleure satisfaction relativement aux données fournies.

Tableau 5 Composants du modèle d'affaire CANVAS – Adapté de (Pigneur & Fritscher 2014)

Perspective	Question	Blocs
Offre	Quoi ?	• Proposition de valeur
Client (côté droit)	Qui ?	• Segmentation des clients • Canaux de distributions • Relation avec les clients
Activité	Comment ?	• Ressources clé • Activité clé • Partenaires clé
Financement	Combien ?	• Moyen de rendement • Coût des structures

Le design du modèle CANVAS, illustré sur la Figure 13, représente au centre l'offre, à droite les blocs liés au client et revenu et à gauches les blocs liés à l'activité et à la structure de coûts.

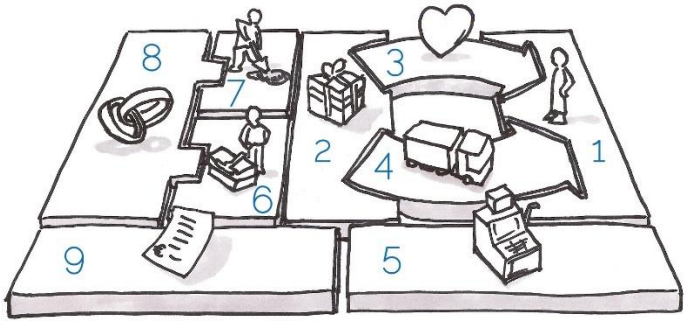


Figure 13 Modèle CANVAS (Gay 2015)

Chaque bloc nécessite des informations spécifiques (Osterwalder & Pigneur 2011) :

- Segmentation des clients : L'information relative à ce bloc fait référence aux clients pour lesquels le produit/service va être proposé ;
- Proposition de valeur : Description du produit/service, information relative à la valeur qui va être fournie aux clients ;
- Canaux de distribution : Savoir comment est-ce que le contact avec le client va être établi. Décrire comment est-ce que l'organisation va contacter les clients afin de leur transmettre leur valeur. Les canaux de distribution se réfèrent, normalement, aux cinq concepts suivants : connaissance, évaluation, achat, distribution et service après-vente ;
- Relation clients : Savoir comment est-ce que l'organisation va être à l'écoute de ses clients et va communiquer avec eux. Les pages sur les réseaux sociaux sont un bon exemple de ceci ;
- Sources de revenus : Déterminer comment est-ce que l'organisation va rentabiliser son activité ;
- Ressources clé : Décrire quels sont les ressources qui sont importantes pour le lancement du projet. Ceux-ci peuvent-être, entre autres : physiques, financières, intellectuelles/humaines, matérielles ;

- Activités principales : Déterminer quelles sont les activités les plus importantes pour que l'organisation puisse fonctionner correctement. Par exemple, développement logiciel, marketing ;
- Partenaires clés : Informations relatives aux partenaires nécessaires pour que l'organisation puisse bien fonctionner ;
- Structure de coûts : identifier les différents coûts du projet.

Le modèle d'affaire CANVAS pour le projet de cette étude est représenté dans annexe nº 3.

3.5 Modélisation et quantification du processus de négociation

Lors d'une négociation, il peut s'avérer difficile d'arriver à un accord en raison de critères, intérêts et points de vue divergents de la part des intervenants. À cela vient s'ajouter le fait qu'au long de la négociation, certaines des alternatives qui pouvaient être envisageables au début ne le sont plus, ou que de nouvelles alternatives apparaissent (Gomes & Gomes 2004). Des outils facilitant le choix de décisions ou des supports d'aide à la décision peuvent être utilisés afin de réduire les divergences et aider les parties à trouver un consensus. Cette sous-section vise à décrire quelques-unes des théories pouvant être utilisées lors d'une négociation telle que la théorie des ensembles approximatifs, la théorie des ensembles flous et la théorie des jeux. Pour que ces méthodes puissent être appliquées il est nécessaire que chaque intervenant possède une base de connaissance[10] sans laquelle il n'est pas possible d'appliquer les diverses théories (Gomes & Gomes 2004).

La théorie des ensembles approximatifs vise à réduire l'indiscernabilité qui peut exister au sein d'un problème. Elle permet de déterminer, de façon statistique, quel sont les critères qui ont plus de poids et pour chacun d'eux quel est le négociateur ayant le plus de crédibilité. Par exemple, lorsqu'il est question de problèmes de programmation, il est plus probable qu'un groupe composé d'ingénieurs informaticiens et logiciel soit plus crédible qu'un groupe composé uniquement d'ingénieurs informaticiens (Gomes & Gomes 2004).

La théorie des ensembles flous indique qu'une alternative peut être vue comme une solution intermédiaire, éliminant ainsi le fait qu'une alternative soit complètement ou pas du tout viable. Les ensembles flous sont généralement associés à l'inconnu ou à l'imprécision des données. L'application de cette théorie vise à donner une réponse à la question « quelle serait la possibilité d'arriver à un résultat de consensus » (Ambapour 2009; Gomes & Gomes 2004).

La théorie des jeux consiste en la participation de deux négociateurs au moins, qui, à tour de rôle, maximisent leurs bénéfices en minimisant celui des autres. Pour ce faire, chaque joueur/négociateur doit connaitre les règles du jeu et savoir (i) comment chaque joueur peut se comporter, (ii) l'ensemble d'actions possibles (correspondant aux séquences d'actions possibles, c'est-à-dire à la façon dont avance la négociation à chaque tour), (iii) les résultats

[10] Composé de règles de base servant à la modélisation du problème et de faits de base représentant les informations concernant le problème (Denis & Miclet 2006).

après chaque tour, (iv) le *payoff* (correspondant aux résultats possible après plusieurs tours), et (v) les stratégies (savoir comment est-ce que chaque négociateur peut répondre à chaque tour et à tout moment) (Dias 2004).

Comme discuté dans la section 3.3 (page 30), les négociations qui s'appliquent dans le cadre de ce projet sont du type gagnant-gagnant, en recourant à la théorie des ensembles approximatifs. En effet, le projet incorporant une diversité de domaines d'études – médecins, biologistes et informaticiens – ce choix est largement recommandé. Chacun des spécialistes saura quels sont les aspects importants face à leur domaine de compétence, même si cela n'est pas le cas dans d'autres domaines en dehors de leurs compétences. La théorie sélectionnée permet la réduction de l'indiscernabilité existante grâce à l'application de poids sur les divers critères.

4 Design - approche théorique

Selon (Fayyad et al. 1996), « l'apprentissage automatique est un processus non-trivial d'identification de structures inconnues, valides et potentiellement exploitables dans les bases de données. » Actuellement, il existe de nombreuses méthodologies indiquant l'ordre des opérations à suivre pour la création de systèmes d'apprentissage automatique. L'intérêt pour trouver un standard est aussi grandissant, puisque les techniques d'apprentissage automatique sont de plus en plus utilisées dans les plus divers domaines tels que l'économie, le marketing et la biologie (Oprean 2010). Parmi les méthodologies existantes, celle de (Fayyad et al. 1996) sera adoptée tout au long du développement du projet puisqu'elle est conçue pour le domaine de la recherche, tandis que d'autres méthodologies, telles que CIRSP-DM et SEMMA, sont plus orientée vers les domaines de l'industrie (Oprean 2010).

Ce chapitre vise à décrire (i) quels sont les méthodologies existantes qui ressemblent le plus à ce qui a été développé, dans la section 4.1 (page 36)**,** et (ii) la méthodologie proposée par (Fayyad et al. 1996) utilisée tout au long du développement du projet, dans la section 4.2 (page 37). Sachant que tout processus d'apprentissage automatique nécessite des données à partir desquelles il est possible de créer les sets de données, la section 4.3 (page 39) présente l'information contenue dans les deux bases de données, Phage_Bact et "*DataBase of Protein Domain Interactions*" (DOMINE), qui ont servi à construire les différents sets de données. La section 4.4 (page 45) explique les deux stratégies entreprises pour l'élaboration de deux types de sets de données basés (i) sur les domaines des protéines et (ii) sur la composition chimique des séquences protéiques. La section 4.5 (page 55) vise à expliquer comment les sets de données ont été utilisés ainsi que la description des techniques d'apprentissage automatique utilisées, les tests et résultats étant discutés dans le chapitre "Évaluation et tests" (page 91). Ce chapitre vise à expliquer ce qui a été fait d'une façon schématisée laissant les détails concernant l'implémentation pour le chapitre 5 (page 59).

4.1 Évaluation et tests des méthodes étudiées

Dans cette section sont décrites des approches existantes similaires à ce qui a été développé, avec pour chacune une description des mesures de performances, des méthodologies utilisées et du set de données. Actuellement, il n'a été trouvé aucune approche identique au projet en cours de développement. Cependant, plusieurs méthodes d'apprentissage automatique ont été utilisées pour prédire les interactions entre les protéines de virus et d'humains, afin de mieux comprendre les mécanismes d'infection (Cui et al. 2012; Dyer et al. 2007; Coelho et al. 2014). Ces modèles serviront de base au développement de notre méthode de prédiction des interactions phage-bactérie.

(Cui et al. 2012) proposent une approche d'apprentissage automatique basée sur un modèle de MVS pour la prédiction d'interactions entre protéines humaines et protéines virales. Ce dernier a utilisé son modèle pour la prédiction d'interactions entre protéines humaines et des virus de l'Hépatite C (VHC) et Papillome[11] (VPH). Les résultats qu'il a obtenus peuvent être consultés dans le Tableau 6. La sensibilité, spécificité et *accuracy* ont été les mesures utilisées pour l'évaluation de la performance du modèle. Les résultats ont été obtenus en moyennant trois exécutions du modèle avec un set de données différent de celui utilisé pour l'entraînement. Les résultats obtenus pour la sensibilité, spécificité et *accuracy* ont été respectivement de 78% ; 85% et 82% pour le VHC et de 79%, 88%, et 83% pour le VPH.

(Dyer et al. 2007) propose une approche par MVS pour la prédiction IPP entre humain et Virus de l'Immunodéficience Humaine. Les mesures de performance utilisées sont la précision et le rappel avec validation-croisée (*k=4*). Avec cette configuration, les auteurs arrivent à une performance de 70% pour la précision et de 40% pour la sensibilité.

(Coelho et al. 2014) propose une méthode permettant de prédire quelles protéines sont impliquées dans les infections par des pathogènes parodontaux [12] , en recourant à l'apprentissage supervisé avec réseaux bayésiens. Les auteurs ont utilisés la validation croisée avec *k=5* et l'*accuracy*, le score F1, la précision et le rappel comme mesure de performance. Le Tableau 6 résume l'information concernant les conclusions et résultats obtenus par les divers auteurs. Il est à noter qu'aucun de ceux-ci n'a réalisé de tests statistiques d'hypothèses pour soutenir leurs affirmations et résultats.

[11] Virus responsable entre autres, pour l'apparition de verrues cutanée et cancer du col de l'utérus (Morin et al. 2006)

[12] Pathogènes attaquant tout l'environnement de fixation dentaire (Guthmiller & Novak 2002)

Tableau 6 Précision classifications IPP. EN : set de validation; TE : set de test.

Nom	Accuracy	F1	Précision	Sensibilité	Spécificité	Echantillon-nage	Apprentiss-age	Set de données
(Cui et al. 2012) VHC	81.6%	---	---	77.8%	85.4%	---	MVS	EN 1'000 TE 390
(Cui et al. 2012) VPH	83.3%	---	---	78.8%	87.8%	---	MVS	EN 1'000 TE 390
(Dyer et al. 2007) **Malad. Inféct.**	---	---	70%	40%	---	Validation croisée (4)	MVS	EN 800 TE 78000
(Coelho et al. 2014) **Inféct. Orale**	85%	85%	86%	28%	---	Validation croisée (5)	Réseaux bayésiens	EN 36'968 TE 6'716'792

Aucun des auteurs n'ayant mis à disposition sa méthodologie, le Tableau 6 représente la seule source d'informations disponible pour comparer les méthodes entre elles (pas de benchmark). Dans notre cas, la méthodologie développée ne permet pas de prédire les IPP, mais se base sur celles-ci pour prédire les interactions entre phage et bactéries. La performance de notre modèle est tout de même comparée à la performance des modèles présentés dans le Tableau 6 à titre indicatif.

4.2 Processus de découverte d'informations

Pour le développement du projet, la méthodologie de « découverte de connaissances » proposée par (Fayyad et al. 1996) sera adoptée. Celle-ci, schématisée dans la Figure 14, est composées de neuf étapes :

1. **Acquisition de connaissances relatives au domaine d'étude et compréhension des objectifs visés**: Une étude concernant les domaines biologiques nécessaires à la compréhension du problème a été réalisée dans la section 2.3.1 (page 11). L'objectif du projet, décrit dans la section 1.3 (page 3), consiste en l'élaboration d'un modèle

prédictif capable de prédire, étant donnée une bactérie, quels phages peuvent interagir avec celle-ci et l'infecter **;**

2. **Sélection et création de l'ensemble de données utilisée tout au long du développement** : Pour ce faire, la HEIG-VD a fourni une base de données contenant plus de 2'000 paires bactérie-bactériophage pour lesquelles les séquences protéiques sont également disponibles. Il a ensuite été nécessaire de télécharger une deuxième base de données, DOMINE, contenant des interactions entre domaines de protéines. Pour chacune de ces bases de données, il a été nécessaire de :
    a. Analyser les contenus des bases de données, étudier l'information qui y était insérée et comprendre sa signification (section 4.3, page 39) ;
    b. Sélectionner l'information jugée utile pour l'élaboration de deux types de sets de données (section 4.4, page 45) :
        i. Set de données basé sur les domaines d'interactions des protéines (section 4.4.1, page 46) ;
        ii. Set de données basé sur les séquences d'acides aminés de chaque protéine (section 4.4.2, page 53) ;
    c. Générer les sets de données (chapitre 5.2, page 60).
3. **Prétraitement et nettoyage des données :** Traitement de données manquantes, élimination de données inconsistantes, valeurs isolées, conversion de données catégoriques en numériques. Éliminations de variables corrélées ou ayant un bas gain d'information. Généralisation d'attributs catégoriques, normalisations de données et discrétisation des attributs continus (Section 4.4.1.4, page 51) ;
4. **Transformation des données** : Réduction et sélection de variables (Section 5.2.2, page 75) ;
5. **Sélection du type d'apprentissage automatique en accord avec l'objectif du projet** : Dans le cadre du projet, c'est du supervisé (présence/absence d'interaction phage-bactérie) (Section 2.3.2, page 17) ;
6. **Sélection d'un algorithme d'apprentissage automatique :** Choisir le modèle en accord avec l'objectif visé et définir les paramètres (section 4.5, page 55)
7. **Implémentation de l'algorithme :** Implémenter l'algorithme sélectionné à l'étape 6. Dans le cadre du projet il s'agit d'utiliser les bibliothèques de *python* contenant les divers algorithmes (chapitre 5, page 59) ;
8. **Évaluation** : Évaluer les performances de l'algorithme sélectionné. C'est au cours de cette étape qu'il sera possible de retourner en arrière et de choisir d'autres algorithmes pour pouvoir comparer la performance de divers algorithmes. Selon la capacité explicative de l'algorithme, c'est également au cours de cette étape que les données explicatives seront utilisées afin de donner une explication à l'utilisateur de comment l'algorithme est arrivé à un résultat (chapitre 6, page 91) ;
9. **Utiliser les connaissances découvertes** : Cette étape consiste en l'intégration des modèles créés avec d'autres systèmes. Cette dernière étape ne sera pas abordée au cours du projet par manque de temps (chapitre 7, page 111).

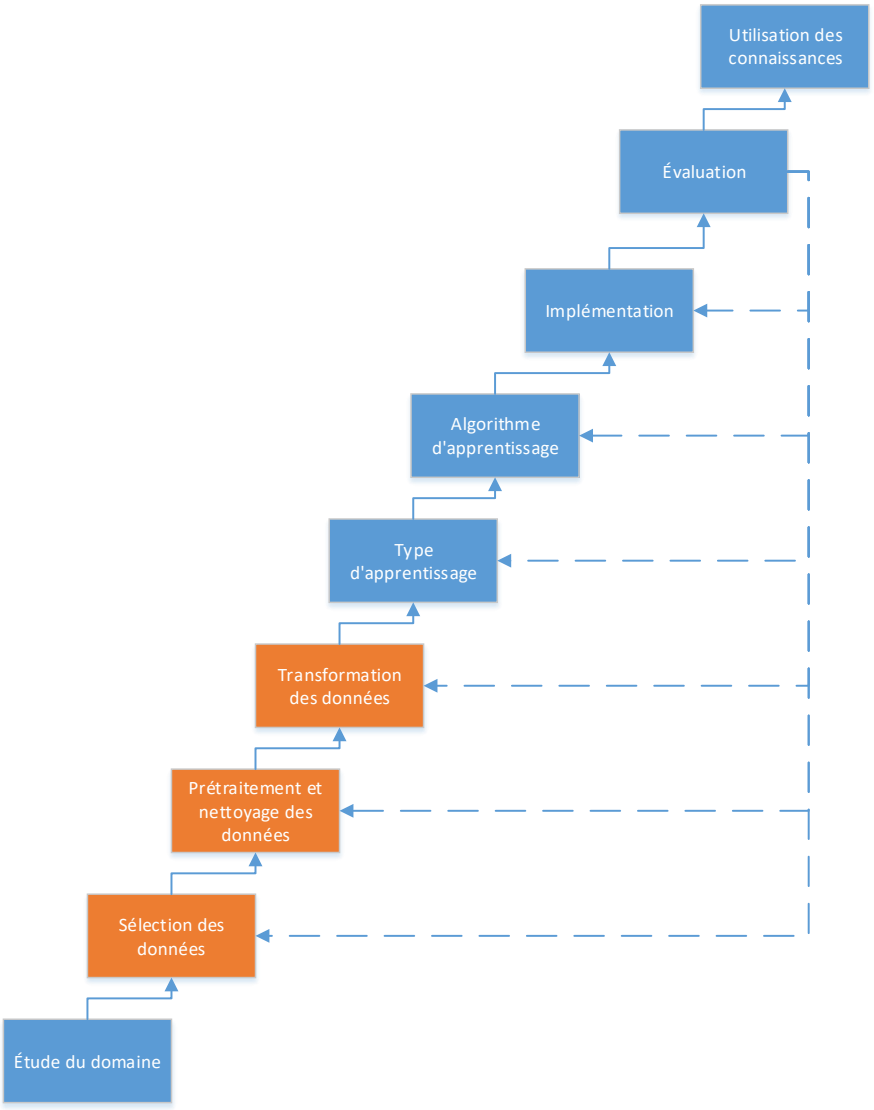


Figure 14 Méthodologie « Découverte de connaissances » – Adapté de (Fayyad et al. 1996)

La plus grande partie du projet se concentre sur les étapes 2 à 4 – en orange sur la Figure 14. Ces étapes sont en effet considérées comme les plus importantes et peuvent occuper jusqu'à 80% du temps total d'un projet de *machine-learning* (Piramuthu 2004; Zhang et al. 2003; Ville 2001). En effet, une mauvaise sélection et traitement des données mène à la création d'un mauvais modèle ou d'un modèle qui ne peut répondre aux objectifs du projet (Piramuthu 2004).

4.3 Données fournies

Après être entré dans le domaine d'étude et avoir compris le problème à résoudre, la deuxième étape du processus de découverte de connaissances consiste en l'analyse et en la compréhension des données fournies. Pour le projet, deux bases de données ont été fournies. La première, décrite dans la section 4.3.1 (page 40), nommée Phage_bact et élaborée par Aitana

Neves à la HEIG-VD, contient l'information concernant 1'065 paires positives et 1'065 paires négatives d'interactions entre une bactérie et un bactériophage. Phage_bact contient également diverses informations biologiques concernant chaque organisme comme leurs séquences ADN, les séquences protéiques et la quantité de protéines. La deuxième base de données décrite dans la section 4.3.2 (page 42), dont la version 2.0 de 2010 a été téléchargée sur le site DOMINE[13] (Raghavachari et al. 2008), contient diverses informations concernant les domaines d'une protéine tels que leurs fonctions, une brève description les concernant et les paires de domaines qui sont prédits pour interagir. Les séquences ADN et protéiques disponibles dans la base de données Phage_Bact étant au format FASTA ou multi-FASTA, la section 4.3.3 (page 44) explique brièvement ce format spécifique au domaine bio-informatique.

4.3.1 Base de données Phage_bact

Phage_bact, représentée sur la Figure 15, contient l'information concernant les paires bactéries-bactériophages positives et négatives ainsi que les séquences ADN et protéiques au format FASTA et multi-FASTA, décrit dans la section 4.3.3 (page 44). Cette dernière est composée de quatre tableaux décrits ci-dessous :

- *Bacteria* : décrit 43 bactéries où pour chacune d'elles est fourni un identificateur unique (id), son espèce (*species*), sa souche (*strain*), son numéro GI (*GI*), le nombre de protéines (*Nb_proteins*), la séquence génomique complète dans le format FASTA (*whole_genome*), la séquence génomique codante au format multi-FASTA (*dna_cod_seq*) et protéomique dans le même format (*prot_seq*).
- *Phages* : contient l'information de 1065 bactériophages où pour chacun d'eux est fourni un identificateur unique (*id*), un nom (*name*), son GI (*GI*), le nombre de protéines (*Nb_proteins*), son génome complet en FASTA (*whole_genome*), la séquence génomique codante au format multi-FASTA (*dna_cod_seq*) ainsi que sa séquence protéomique dans le même format (*prot_seq*).
- *Interactions* et *Negative_Interactions* : contient l'information concernant 1065 interactions positives et 1065 interactions négatives. Ces tableaux sont composés d'un identificateur (*interaction_id*), de l'identificateur de la bactérie (*Bacterium_id*) et du bactériophage (*Phage_id*) impliqués dans l'interaction ainsi que le type d'interaction (*class*).

Il aurait été possible de joindre le tableau des interactions positives et négatives en y joignant l'information des tableaux *Interactions* et *negative_interactions*, cependant il m'a été demandé de ne pas modifier la base de données, car le tableau *negative_interactions* peut être généré de plusieurs manières différentes et le fait d'avoir plusieurs tableaux permet de mieux retracer ce qui a été fait.

[13] Site DOMINE : http://domine.utdallas.edu/cgi-bin/Domine?page=download

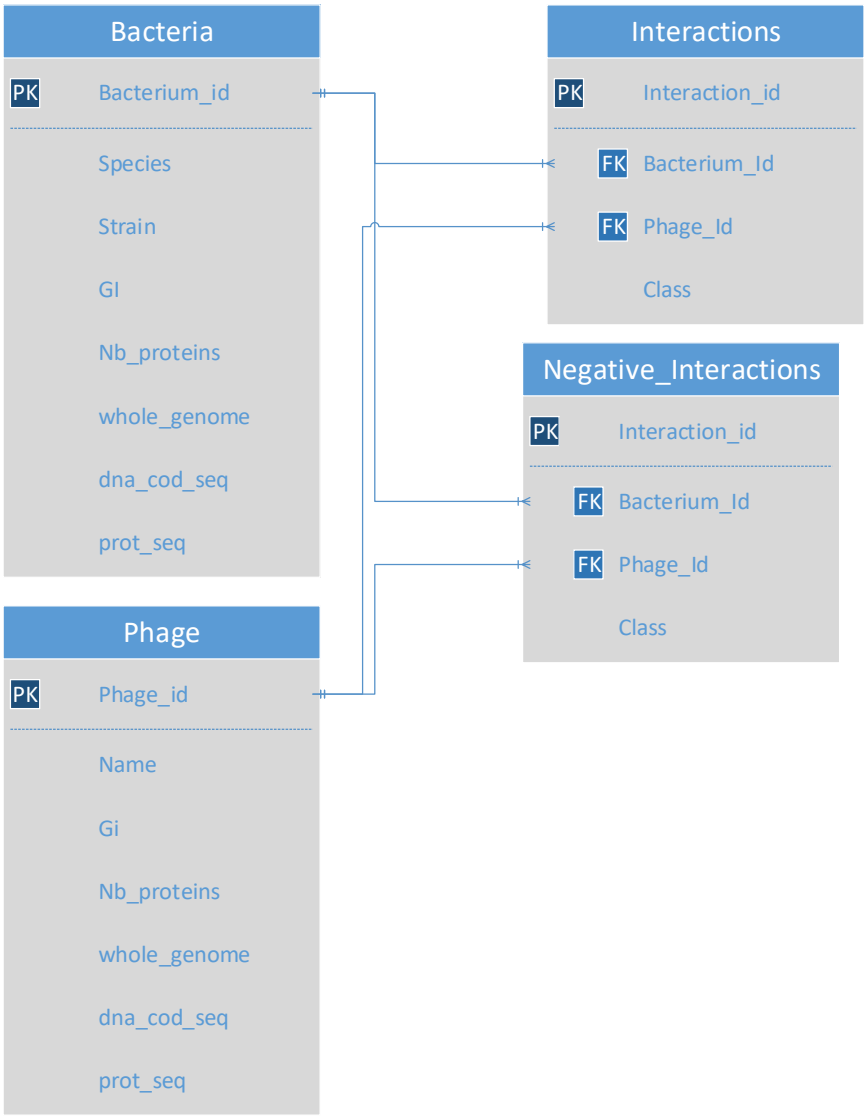


Figure 15 Diagramme de base de données Phage_bact

Le type des champs comportant les séquences ADN et protéiques des cellules ont été modifié de « TEXT » en « MEDIUMTEXT » par le fait qu'une base de données MySql fut utilisée pour le développement du projet. Ce changement est dû au fait que le nombre maximum de caractères autorisé dans le type « TEXT » est de 65'535 et dans « MEDIUMTEXT » est de 16'777'215 (Sheldon & Moes 2005). Le Tableau 7 contient l'information concernant le nombre de caractères pour la plus petite et plus grande séquence d'acides aminés du champ « Wholde_genome » et de même pour les séquences protéiques des champs « dna_cod_seq » et « prot_seq » de tous les bactériophages et bactéries. Cette erreur fut détectée grâce aux messages d'attention renvoyés par MySql indiquant que ces champs furent tronqués lors de l'importation des données dans la base de données.

Tableau 7 Plus petite et grande taille des séquences ADN et protéiques

Champs	Bactérie		Bactériophage	
	Minimum	Maximum	Minimum	Maximum
« Whole_genome »	49’581	9’154’641	1’220	236’276
« Dna_cod_seq »	54’570	9’025’479	2	268’551
« Prot_seq »	24’310	3’993’877	490	117’178

4.3.2 Base de données DOMINE

DOMINE, la deuxième base de données représentée dans la Figure 16, fournit l’information concernant les domaines de protéines complémentaires (voir section 4.4.1, page 46). Un domaine d’une protéine peut être vu comme un fragment de celle-ci destiné à une fonction biologique bien particulière. Deux protéines peuvent interagir physiquement par le biais de leurs domaines, si elles possèdent des domaines « complémentaires ». La base de données DOMINE répertorie justement les domaines connus ou prédits pour interagir. Elle est composée de 4 tableaux, décrits ci-dessous :

- *PFAM* : chaque domaine est composé d’un numéro d’accès (*id -DomainAcc*), d’un identifiant de domaine (DomainId), d’une description (DomainDesc) ainsi que du numéro du domaine équivalent dans la base de données *interpro*[14] *;*
- *GO* : *Gene Ontology* (GO) est une ressource informatique regroupant diverses informations concernant un gène, telles que sa fonction et sa localisation cellulaire. L’ontologie décrit également diverses relations entre les registres permettant de savoir, par exemple, si une fonction fait partie d’une autre fonction. Chaque registre est composé d’un identifiant unique (*GoTerm*), de sa fonction (*ontologie*) et de la description de la fonctionnalité (*GoDesc*) (Blake et al. 2013) ;
- *PGMAP* : Ce tableau établit la relation entre les domaines (*DomainAcc*) et leurs correspondances dans la GO (*GoTerm*) ;
- *Interactions* : Regroupe les paires d’interactions entre domaines obtenus grâce à 15 sources d’informations différentes (*iPfam, 3did, ME, RCDP, Interdom, P-value, DPEA, PE, GPE, DIPD, RDFF, K-GIDDI, INSITE, DomainGA, DIMA*), un niveau de fiabilité de l’interaction (*PredictionConfidence*) ainsi qu’un *boolean* indiquant si la paire de domaines appartient ou pas au même processus biologique (*SameGO*) (Raghavachari et al. 2008).

[14] InterPro est une base de données réunissant les informations de plusieurs autres bases de données, constamment mise à jour et contenant diverses informations concernant chaque protéine, notamment sa séquence protéique, sa fonction ainsi que les domaines qui la composent (Mitchell et al. 2015).

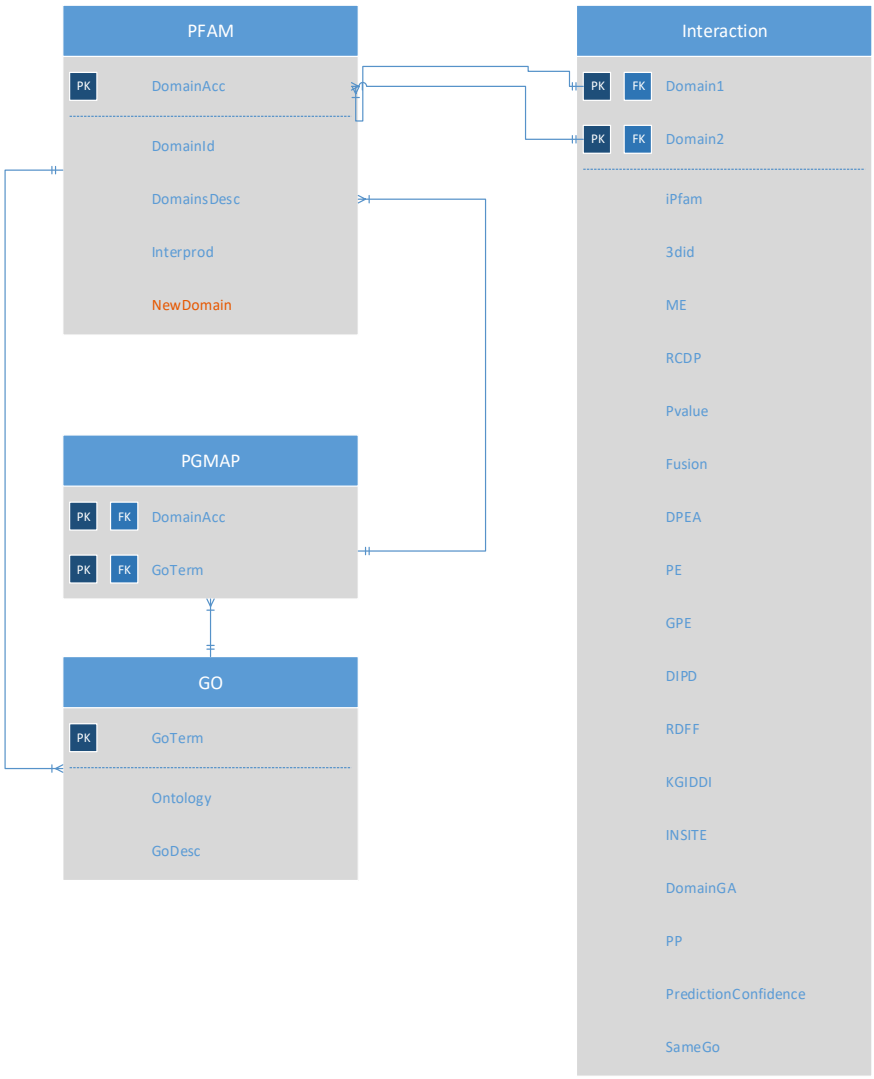


Figure 16 Diagramme de la base de données DOMINE

Deux problèmes ont été mis en évidence dans la base de données DOMINE. Le premier vient du fait que certains domaines présents dans le tableau PGMAP ne le sont pas sous PFAM alors que ceux-ci possèdent une connexion du type clé secondaire comme il est possible de le voir sur la Figure 16. Le deuxième est que DOMINE n'est pas actualisée depuis septembre 2010 alors que les APIs utilisées sous le chapitre 5 (page 59) pour la création des sets de données ont recours à des bases de données de l'EBI constamment mises à jour. Afin de résoudre les deux problèmes, il a été créé un script en *python* qui va, pour chaque domaine présent dans PGAMT, vérifier si celui-ci est présent dans PFAM, et dans le cas contraire, fait une requête à l'API de l'EBI afin d'en obtenir les informations préconisées par le tableau PFAM et les y additionner. Afin d'actualiser les domaines dans le tableau PFAM, un champ « NewDomain » contenant le nouveau domaine obtenu par l'EBI a été ajouté. Ne pouvant actualiser le tableau INTERACTIONS, il a été important de garder l'ancien et le nouveau domaine qui seront utilisés pour la création des sets de données décrits dans la section 4.4 (page 45).

4.3.3 Format Fasta et multi-Fasta

Les bases de données utilisées pour le développement du projet contiennent des registres de séquences au format multi-FASTA. Ce format est spécifique aux données biologiques et composé d'informations génomiques qui, pour le projet, correspondent à la séquence d'acides aminés de chaque protéine (Mougenot 2012). Chaque séquence est composée d'un minimum de deux lignes : la première, servant d'identification et débutant avec le caractère « > », contient des informations relatives à la protéine ; la deuxième ligne et les suivantes correspondent aux séquences d'acides aminés de la protéine (voir représentation Figure 17).

L'information correspondant à l'identification de la protéine est la suivante :

1. Base de données qui a permis l'extraction de la séquence ainsi qu'identifiant unique au sein de la base de données (ID). Le Tableau 8 illustre les formats de bases de données les plus utilisées ;
2. L'identifiant de la séquence d'acides aminés ;
3. Le type de protéine ;
4. ID de la protéine ;
5. Localisation de la séquence au sein du génome de la bactérie ou du bactériophage.

>**(1)**lcl|NC_027352.1_prot_YP_009149053.1_15 **(2)**[gene=JBP901_gp015] **(3)**[protein=hypothetical protein] **(4)**[protein_id=YP_009149053.1] **(5)**[location=10247..10561]
METARKTLTAQLKDTVAFFKDRKTVGRMKTEHSIFIPSKLMLEKEGETAEFILVARHIGRQPAWRCPVAA
EQVNIVKTAGVEFVFSFQTDTDYDAIMDYISSKI
>lcl|NC_027352.1_prot_YP_009149054.1_16 [gene=JBP901_gp016] [protein=putative PhoH family protein] [protein_id=YP_009149054.1] [location=10755..11474]
MSFFGITARNVGQKALAEACKSRVPIIIGTGRAGTGKSLIAQAVGFDTVFEEKRWSTNTGKFVYTRLQVD
VGKEVGFLPGDLDEKSNPYFRPFFDNLTLLDKGGYMRNYISQGKIVLDHIQTIRGGTYHDTYLVVDEAQN
LDNATMTAIGTRLALGSKLILLGNFAQIDVDKLKNPEHNGFYKLLKGLHESGRTDMFTHVHLTQGERGEI
ADLVEGIMVPQSEVNPSFVKLEEKGLLNY

Figure 17 Extrait de la base de données – *Bacillus Cereus* ATCC 14579[15]

La différence entre un format FASTA et multi-FASTA réside dans le fait que le premier n'est composé que d'une séquence alors que le deuxième en comporte deux ou plus. La librairie *BioPython*, qui permet d'analyser et traiter ce format de données (Bassi 2009), a été utilisée dans les divers scripts développés. A noter qu'il est possible qu'à certains emplacements de la séquence d'acides aminés se trouve un « X », voir Figure 3 pour la liste des acides aminés. Ce caractère signifie qu'il existe un acide aminé à cet emplacement mais que celui-ci n'a pas été correctement séquencé (Zahoorullah 2015).

[15] Bactérie qui contamine divers aliments tels que le riz ou lait cru qui, si ingérée, peut provoquer des vomissement et colite (Griess 2013)

Tableau 8 Format d'identification des bases de données génomiques (Wills 2015)

Base de données	Format de la ligne d'identification
GenBank	gi|numéro gi|gb|numéro d'accession|locus
EMBL	gi|numéro gi|emb|numéro d'accession|locus
DNA Data Bank of Japan	gi|numéro gi|dbj|numéro d'accession|locus
NBRF PIR	pir||entrée
Protein Research Foundation	prf||nom
Swiss-Prot	sp|numéro d'accession|nom
Brookhaven Protein Data Bank (1)	pdb|entrée|chaîne
Brookhaven Protein Data Bank (2)	entrée:chaîne|PDBID|CHAÎNE|SÉQUENCE
Brevets	pat|brevet|numéro
GenInfo Backbone Id	bbs|numéro
General database identifier	gnl|base de données|identifiant
NCBI Reference Sequence	ref|numéro d'accession|locus
Local Sequence identifier	lcl|identifiant

4.4 Création des sets de données

Comme décrit au chapitre précédent, après avoir effectué une recherche concernant le domaine d'études et analysé les données fournies, il est nécessaire de créer des sets de données qui puissent être lus et interprétés par les algorithmes d'apprentissage automatique. Dans le cadre du projet, deux types de sets de données ont été créés : le premier, décrit dans la section 4.4.1 (page 46), se base sur les domaines présents sur les protéines. Le deuxième, décrit dans la section 4.4.2 (page 53), se base sur la composition chimique des séquences protéiques. Pour chaque cas, un set de données "normalisé" a également été créé en divisant chaque score par le nombre d'IPPs de la paire bactérie-bactériophage. La Figure 18 illustre les sets de données qui ont été créés.

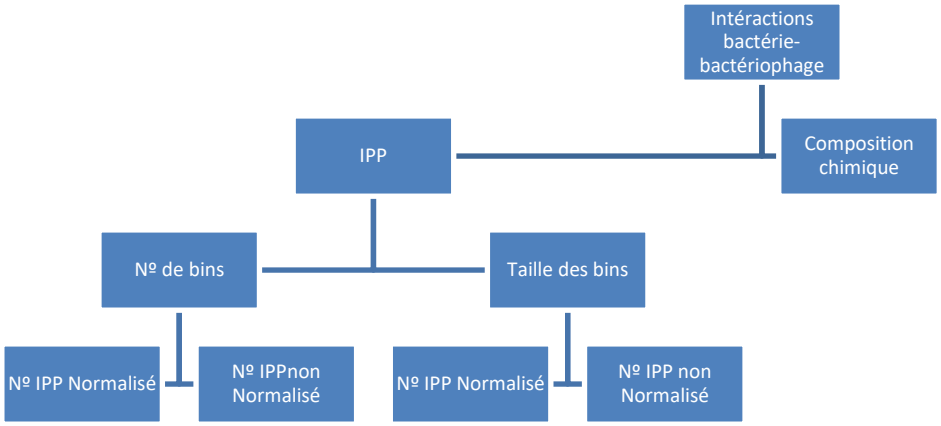


Figure 18 Diagramme set de données

4.4.1 Variable « domaines de protéines »

Comme décrit plus haut dans la section « Mécanismes d'infection (attaque) et de défense » (page 14), il est connu que les phages arrivent à infecter certaines bactéries et que cela se passe, entre autres, au niveau des interactions entre leurs protéines. Nous partons donc de l'hypothèse que les paires bactérie-bactériophage qui interagissent seront caractérisées par un grand nombre d'IPPs, contrairement aux paires bactérie-bactériophage qui n'interagissent pas.

Comment est-ce que les protéines interagissent pour donner lieu à des IPPs ? Une protéine est composée d'un ou plusieurs domaines qui, dans la perspective biologique, peuvent être vus comme une fonction pour laquelle la protéine a été créée (par exemple, rôle dans la production d'anticorps) (Parham 2003; Terrapon 2010). C'est notamment par l'intermédiaire de ces domaines que des IPPs peuvent avoir lieu, par l'interaction d'un domaine avec un domaine d'une autre protéine. La présence de domaines "complémentaires" sur deux protéines permet donc, en partie, de prédire si une IPP est probable ou non (ce genre de *feature* a notamment été utilisé dans (Coelho et al. 2014) pour prédire des IPPs).

Le premier type de set de données que nous allons créer sera basé sur les scores d'IPPs obtenus pour chaque paire bactérie-bactériophage (en testant toutes les interactions protéine-protéine possibles pour cette paire). Il est important de noter que le nombre d'IPPs sera propre à chaque paire bactérie-bactériophage, puisque le nombre d'IPPs est égal au nombre de protéines de la bactérie multiplié par le nombre de protéines du bactériophage. Afin de déterminer les scores d'IPPs pour chaque paire bactérie-bactériophage, la procédure suivante a été suivie : (1) identification des domaines se trouvant sur les protéines de chaque organisme ; (2) identification des domaines "complémentaires" (grâce à la base de données DOMINE) et calcul d'un score pour chaque IPP ; (3) histogramme des scores IPPs pour chaque paire bactérie-bactériophage et création du set de données "domaines d'interaction".

4.4.1.1 Recherche de domaines dans une séquence protéique

L'interface de programmation (API) *Biological sequence analysis using profile hidden Markov models* (HMMER) par (Eddy & Wheeler 2015) a été utilisée afin de prédire les domaines dans une séquence protéique. Celle-ci reçoit en entrée une séquence protéique et renvoie les domaines *PFAM* [16]qui s'y trouvent est consultée à travers l'architecture *Representational State Transfer* (REST). L'API utilisée permet de retourner l'information sous plusieurs formats tels que le *javaScript Object Notation* (Json) ou *Extensible Markup Language* (XML).

Pour la réalisation de cette première phase, un tableau nommé « PROTDOM » a été créé dans la base de données « Phage_bact » où chaque registre contient : un identificateur unique du registre (ProtDomId), l'identification de la séquence (ProtId), le domaine (DomainAcc), l'identificateur interne de la cellule (Cell_id), un booléen indiquant s'il s'agit d'un phage ou d'une bactérie (Bacteria_Cell) et la séquence de la protéine dans le cas où le script n'aurait reçu aucune réponse de l'API HMMER (ProtSeq).

La Figure 19 correspond à un extrait du tableau PROTDOM représentant les trois sorties possibles après avoir consulté l'API. Pour chaque domaine, il peut y avoir :

- Plusieurs domaines qui sont retournés pour une séquence et pour chacun d'eux un registre est inséré ;
- Aucun domaine n'est retourné indiquant que la séquence analysée n'en possède aucun et dans ce cas est inséré un registre contenant « --NP-- » dans le champ des domaines ;
- Aucune réponse n'est retournée dû au *timeOut* ou au fait que le serveur HMMER soit indisponible, auquel cas est inséré un registre contenant « --PN-- » dans le champ des domaines et la séquence protéomique dans le champs ProtSeq de sorte à ce que celle-ci puisse être ré-analysée.

ProtDomId	ProtId	DomainAcc	Cell_id	Bacteria_Cell	ProtSeq
238915	lcl|NC_012660.1_prot_WP_015886097.1_5291	PF02779	26	1	--
238914	lcl|NC_012660.1_prot_WP_015886097.1_5291	PF13292	26	1	--
238913	lcl|NC_012660.1_prot_WP_015886097.1_5291	PF02780	26	1	--
238907	lcl|NC_012660.1_prot_WP_015886092.1_5285	--NA--	26	1	--
238906	lcl|NC_012660.1_prot_WP_043206336.1_5284	PF02518	26	1	--
238875	lcl|NC_012660.1_prot_WP_043205960.1_5260	--PN--	26	1	MIRLAHSEQAVWDGFLISVTAQLIQQDANRGPEYIAE RAAEIADEMLLERRERCVERRAASVDWIGPAKGQ

Figure 19 Extraits tableau PROTDOM

Un premier script a été créé afin de parcourir toutes les protéines de tous les organismes présents dans la base de données Phage_bact et d'en déterminer les domaines. La Figure 20 illustre le procédé effectué tout au long de cette étape. La base de données Phage_Bact, représentée en vert, contient toutes les bactéries, représentées en brun, et tous les phages, représentés en violet. Chaque séquence protéique, représentée par les traits noirs, est analysée afin de déterminer les domaines qui la composent, représentés par les formes de couleurs. Après exécution du script, nous avons pu observer que les bactéries possèdent plus de domaines que les bactériophages, en moyenne 81 [17] fois plus de domaines. Ceci est

[16] Les *PFAM* sont les fonctions connues des protéines (Sonnhammer et al. 1998).

[17] Nombre obtenu en divisant le nombre de domaines par le nombre de bactéries et de même pour les bactériophage.

certainement dû au fait que les protéines des bactéries sont composées de plus d'acides aminés et sont donc plus susceptibles de contenir des domaines.

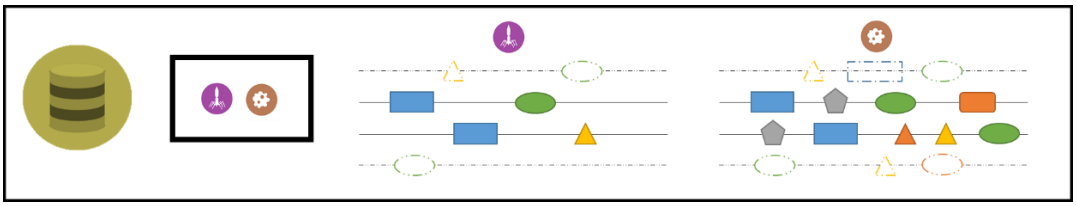


Figure 20 Recherche de domaines dans les séquences protéiques (Leite et al. 2016)

Un deuxième script, décrit dans la section 5.2.1.2 (page 63) a été créé afin de parcourir toutes les protéines du tableau PROTDOM n'ayant pas pu être analysées dû aux erreurs mentionnées ci-dessus (29'429 des 226'219 séquences, représentant un peu plus de 13% des séquences).

4.4.1.2 Détection des domaines connus pour interagir et calcul des scores d'IPPs

Après avoir obtenu tous les domaines de toutes les protéines de chaque organisme, la deuxième étape a consisté, pour chaque paire bactérie-bactériophage, à combiner toutes les protéines de la bactérie avec toutes les protéines du bactériophage de sorte à savoir quels sont les domaines "complémentaires" et obtenir ainsi un score pour chaque IPP (grâce à la base de données DOMINE). La section 5.2.1.3 (page 65) décrit le script responsable de cette étape.

Le score de deux domaines "complémentaires" est égal au nombre de fois que cette complémentarité est prédite parmi les 15 différentes sources d'informations de la base DOMINE. Si un des deux ou les deux domaines ont dû être actualisés avec l'EBI, voir section 4.3.2 (page 42), on effectue la moyenne des scores obtenus avec la combinaison des anciens et nouveaux scores. La somme de tous les scores obtenus en tenant compte de tous les domaines "complémentaires" présents sur deux protéines représente le score d'IPP qui est enregistré dans un tableau nommé « Score_interactions ».

La Figure 21 représente un extrait du tableau « Score_interactions » où sont gardés les scores de toutes les IPPs. Celui-ci est composé des six colonnes suivantes :

- *Score_Inter_id* : un identifiant unique ;
- *ProtBactId* : identifiant unique de la protéine d'une bactérie ;
- *ProtPhageId* : identifiant unique de la protéine d'un bactériophage ;
- *Positive_Interaction* : *boolean* informant s'il s'agit d'une interaction positive (1) ou négative (0) ;
- *Score_result* : la somme de tous les scores des domaines "complémentaires" entre ces deux protéines.

Score_Inter_Id	ProtBactId	ProtPhageId	Positiv_Interaction	Interaction_Id	Score_result
1000969	lcl|NC_002696.2_prot_NP_418854.1_34	gene_48|GeneMark.hmm|112_aa|+|34322|34660	0	515	1
728311	lcl|NC_018289.1_prot_WP_036462788.1_6608	gene_224|GeneMark.hmm|376_aa|+|152849|153979	1	913	2
413167	lcl|NC_018289.1_prot_WP_003893908.1_2496	gene_47|GeneMark.hmm|114_aa|+|36165|36509	1	458	2
784235	lcl|NC_022808.2_prot_WP_003110454.1_5633	lcl|NC_019913.1_prot_YP_007236540.1_138	1	994	1
948098	lcl|CP012023.1_prot_IMCC12053_2319_2319	gene_16|GeneMark.hmm|111_aa|+|9514|9849	0	334	1
100601	lcl|NC_018289.1_prot_WP_003893047.1_1613	gene_120|GeneMark.hmm|160_aa|+|70900|71382	1	17	1
1168516	lcl|NZ_CP009628.1_prot_WP_000929160.1_3047	lcl|NC_015284.1_prot_YP_004323543.1_174	0	1011	1
575001	lcl|NC_018289.1_prot_WP_011731354.1_6359	gene_66|GeneMark.hmm|195_aa|+|59450|60037	1	691	13
705265	lcl|NC_018289.1_prot_WP_011730107.1_4671	gene_22|GeneMark.hmm|37_aa|-|18840|18953	1	871	1
181042	lcl|NC_018289.1_prot_WP_011727552.1_1247	gene_91|GeneMark.hmm|91_aa|-|65175|65450	1	148	2

Figure 21 Extraits tableau Score_interactions

La Figure 22 représente le procédé effectué au long de ce script. La partie de gauche représente la base de données DOMINE contenant l'information sur les domaines "complémentaires" ainsi que leur score. La partie de droite illustre une protéine d'une bactérie (en haut) et une protéine d'un bactériophage (en bas) ainsi que les domaines complémentaires (lignes rouges). La somme des résultats de toutes ces "complémentarités" constitue un score d'IPP, enregistré dans le tableau « PROTDOM ».

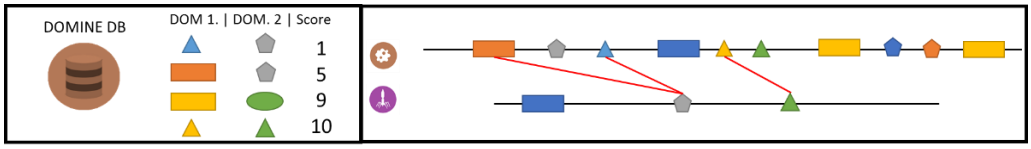


Figure 22 Recherche et calculs des scores d'IPP. (Leite et al. 2016)

Dans l'exemple de la Figure 22, la protéine de la bactéries (en haut) possède 10 domaines, et celle du bactériophage (en bas) 3 domaines. D'après DOMINE, il existe trois paires de domaines complémentaires, représentées par les lignes rouges. En sommant le score DOMINE de chaque paire de domaines, on obtient un score IPP de 5+1+10=16. Cette procédure est répétée pour toutes les interactions protéine-protéine entre une bactérie et un bactériophage.

Sur la Figure 23**,** il est possible de voir que le nombre d'IPPs est différent d'une paire bactérie-bactériophage à une autre. Chaque barre bleue représente en effet les scores IPPs d'une paire bactérie-bactériophage (P-B). Puisque le nombre d'IPPs est différent d'une paire bactérie-bactériophage à une autre, le script décrit dans la section 4.4.1.4 (page 51) et qui génère les sets de données, propose une option permettant de normaliser les scores en les divisant par le nombre d'IPPs de la paire bactérie-bactériophage.

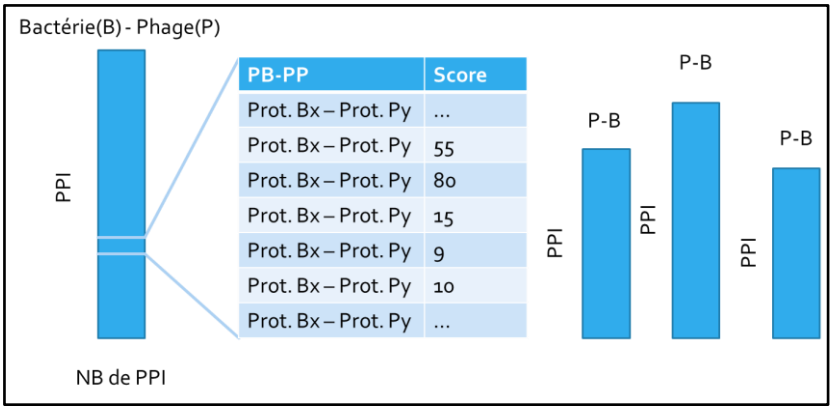


Figure 23 Représentation du nombre d'IPPs pour diverses paires bactérie-bactériophage.

4.4.1.3 Calcul de la fréquence des scores (histogramme)

Les scores IPPs contiennent beaucoup d'informations, mais le problème est que chaque vecteur de scores IPPs a une taille différente selon la paire bactérie-bactériophage choisie (Figure 23). Ceci n'est pas adapté pour entraîner un modèle d'apprentissage automatique, puisqu'avec ces données, on ne peut pas construire un set de données qui soit une matrice rectangulaire. Afin de remédier à ce problème, l'approche choisie a été de représenter les scores IPPs sous forme d'histogramme, en déterminant à l'avance le nombre de *bins* pour toutes les paires bactérie-bactériophage. L'hypothèse sous-jacente est que l'histogramme des scores IPPs sera différent pour une paire bactérie-bactériophage qui interagit vs. une paire qui n'interagit pas (voir Figure 25).

Après avoir calculé le score de chaque IPP, il a donc été nécessaire de déterminer, pour chaque paire bactérie-bactériophage, la fréquence d'apparition des divers scores IPPs ("histogramme"). Pour ce faire, un script faisant appel à l'information contenue dans le tableau « Scores_interaction » a été créé (décrit dans la section 5.2.1.4, page 69)

Les fréquences trouvées sont enregistrées dans un tableau nommé « QtdScores », illustré sur la Figure 24, composé de 5 colonnes décrites ci-dessous :

- *QS_id* : identifiant unique du registre ;
- *Interaction_Id* : identifiant unique de l'interaction ;
- *Positiv_interaction* : *boolean* informant s'il s'agit d'une interaction positive (1) ou négative (0) ;
- *ScoreNumber* : valeur du score
- *QuantityScore* : fréquence avec laquelle le score apparaît

QS_id	Interaction_Id	Positiv_Interaction	ScoreNumber	QuantityScore
1	0	1	1	197
2	0	1	2	156
3	0	1	3	65
4	0	1	4	56
5	0	1	5	24
6	0	1	6	3
7	0	1	7	2
8	0	1	8	1
9	0	1	9	1
10	0	1	10	1
11	0	1	12	1
12	0	1	15	2
13	0	1	17	4
14	0	1	18	2
15	0	1	19	6
16	0	1	0	276024

Figure 24 Extraits tableau « QtdScores »

Un autre script, décrit dans la section 5.2.1.4 (page 69), a été créé pour automatiser ce processus. Ce dernier a été nécessaire car l'une des restrictions lors du processus de création des sets de données, décrit dans la section 1.3 (page 3), était que ce ne soit pas dépendant de la de base de données. Calculer ceci dynamiquement n'a pas été possible dû à la grande quantité de mémoire allouée[18].

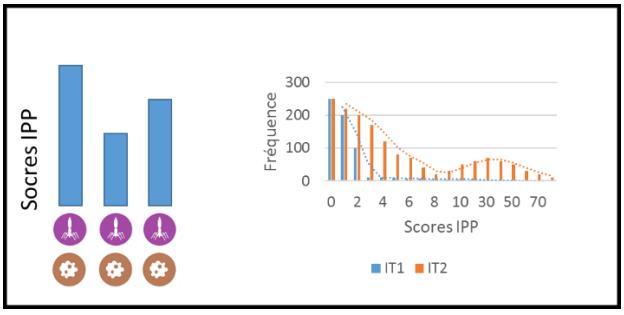


Figure 25 Score IPP et Histogrammes des scores (Leite et al. 2016)

A gauche, vecteur de scores IPPs pour trois paires bactérie-bactériophage. A droite, histogramme (hypothétique) des scores IPPs pour une paire bactérie-bactériophage qui interagit (en rouge, IT2) vs. une paire qui n'interagit pas (en bleu, IT1).

4.4.1.4 Création des sets de données

La création du script, décrit dans la section 5.2.1.6 (page 71), permettant de générer les différents sets de données a nécessité une sauvegarde des données contenues dans le tableau « Qtd_Scores » de sorte à ce qu'il n'existe aucune dépendance entre ce dernier et le script. Le *pickling*, fonctionnalité de *python* permettant la sérialisation et postérieure désérialisation de structures telles que des objets et vecteurs dans des fichiers, a été utilisé afin de garder l'information. Contrairement au simple fichier, ce mécanisme élimine toute nécessité de créer une fonction d'écriture et lecture et ainsi tout problème d'organisation de l'information que l'on souhaite enregistrer. Face au Json et XML, *pickle* ne nécessite aucun convertisseur de données pour pouvoir les sérialiser (Lubanovic 2015). Dans le *pickle* créé sont sérialisés quatre

[18] Lors des expérimentations, l'ordinateur c'est arrêté à 8 GB de mémoire vive

vecteurs correspondant aux attributs id (Qs_id), Classe (Positiv_interaction), score (ScoreNumber) et la fréquence à laquelle celui-ci apparaît (QuantityScore) à partir du tableau « QtdScores ». Un script à part a été créé afin de créer le fichier en *pickling*.

Lorsque le script est lancé, plusieurs paramètres sont demandés à l'utilisateur :

- Le chemin du fichier *pickling* ;
- Le chemin où il prétend enregistrer le set de données généré ;
- Le type de set de données qu'il prétend générer (voir ci-dessous) ;
- S'il souhaite normaliser les données.

L'utilisateur à le choix entre indiquer le nombre de *bins* présents dans l'histogramme ou la taille des *bins,* auquel cas le nombre de *bins* est calculé automatiquement. Pour chacune des configurations, l'utilisateur a le choix entre utiliser les scores normalisés ou non. S'il opte pour les données normalisées, le script va diviser le score de chaque IPP par le nombre d'IPPs de la paire bactérie-bactériophage. La Figure 26 illustre les quatre types de set de données qu'il est possible de générer avec le script.

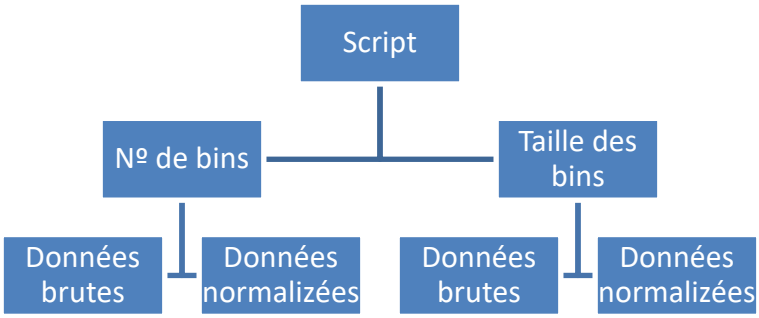


Figure 26 Types de set de données basés sur les domaines d'interactions

La Figure 27 illustre les quatre types de sets de données qu'il est possible de générer (données sont fictives). Les deux premiers sets de données qui sont représentés ont été générés en ayant pour base le nombre de *bins* défini par l'utilisateur. Dans le cadre du projet, la taille des *bins* pour chaque registre n'a pas été enregistrée car ceci ne s'est pas avéré utile. Cependant le script a été fait de sorte à ce que cette altération soit possible n'impliquant que quelques changements au niveau des retours des fonctions. La taille des *bins* est choisie automatiquement par la librairie *histogramme* de *numPy* qui calcule le résultat. Les deux derniers sets de données sont basés sur un vecteur spécifiant la taille des *bins.* Dans ce cas, le script calcule automatiquement le nombre de *bins* nécessaire pour couvrir l'ensemble des scores IPPs de toutes les paires bactérie-bactériophage de la base de données. Il est à noter que les sets de données basés sur la taille des *bins* peuvent contenir des zéros puisque le score IPP maximum varie d'une paire bactérie-bactériophage à une autre.

NºBarres/ B-P	Id	0	1	2	3	4	Classe
Ba-Pa	0	125	14	12	10	9	1
Ba-Pb	1	145	13	10	5	4	1
Bb-Po	2	210	185	57	40	19	0

NºBarres/ B-P	Id	0	1	2	3	4	Classe
Ba-Pa	0	120	19	11	10	10	1
Ba-Pb	1	150	12	11	6	4	1
Bb-Po	2	225	170	55	43	19	0

$\frac{Score}{Nº\ de\ IPP}$

NºBarres/ B-P	Id	0-5	6-10	11-15	16-20	21-25	Classe
Ba-Pa	0	110	20	15	15	0	1
Ba-Pb	1	155	31	15	0	0	1
Bb-Po	2	190	170	80	50	34	0

NºBarres/ B-P	Id	0-2	2-4	4-6	7-8	9-10	Classe
Ba-Pa	0	120	10	5	4	3	1
Ba-Pb	1	130	45	40	10	0	1
Bb-Po	2	220	176	0	0	0	0

$\frac{Score}{Nº\ de\ IPP}$

Figure 27 Exemple des quatre types de sets de données basés sur les domaines des protéines (données fictives) (Leite et al. 2016).

4.4.2 Variables « composition chimique »

Ce type de set de données est composé de 54 variables correspondant à l'information obtenue lors de l'analyse de la séquence protéique des IPPs. Sachant que chaque IPP correspond à une interaction entre une protéine d'une bactérie et une protéine d'un bactériophage, chaque protéine fournit l'information pour 27 variables dont 21 correspondent au pourcentage d'acides aminés, 5 au pourcentage d'éléments chimiques et une dernière représente la somme des poids de tous les acides aminés qui composent chaque protéine en *Dalton*[19]. Pour chaque registre sont additionnés un identificateur et l'information concernant la classe, positive ou négative.

La taille du set de données est calculée en accord avec la formule suivante :

$$\left(\sum_{i,j \in \{set\ positif\}} NP_i * NP_j + \sum_{i,j \in \{set\ negatif\}} NP_i * NP_j \right) * (54 + 2) \qquad (1)$$

, où *i* et *j* correspondent à une bactérie et bactériophage et NP au nombre de protéines de chaque organisme. Il y a 54 variables, en plus de l'id et de la classe, d'où le 54+2.

[19] 1 Dalton ≈ $1.6605*10^{-27}$kg (Rusconi 2011)

La base de données PHAGE_BACT ayant 43 bactéries avec une moyenne de 3'417 protéines chacune, et 1065 phages avec une moyenne de 74 protéines chacun, cela résulterait en un set de données avec plus de 538’587'540 registres avec 54 variables.

Un set de données de cette dimension a trop de variables pour être analysé par les algorithmes d’apprentissage automatique (risque d'*overfitting*, puisque le nombre de variables est bien supérieur au nombre d'exemples dans le set d’entrainement). Par ailleurs, puisque le nombre d'IPPs dépend de chaque paire bactérie-bactériophage, il est à nouveau nécessaire de transformer les données afin que chaque paire bactérie-bactériophage soit représentée par le même nombre de variables.

La méthodologie ACP, décrite dans la section 2.3.2.3 (page 19)**,** a été utilisée pour la réduction de la dimensionnalité du set de données. Le nombre de composantes principales (CP) a été choisi de sorte à ce que le pourcentage de variance représente plus de 90%. Après analyse des données, il a donc été décidé de ne garder que 2 CP, ce qui signifie que chaque paire bactérie-bactériophage est finalement représentée par un total de 54*2=108 variables. La Figure 28 schématise un extrait du set de données où il est possible de voir les deux premières CP composées chacune de 54 variables, dont 27 contiennent l’information des protéines de la bactérie et 27 celles du bactériophage.

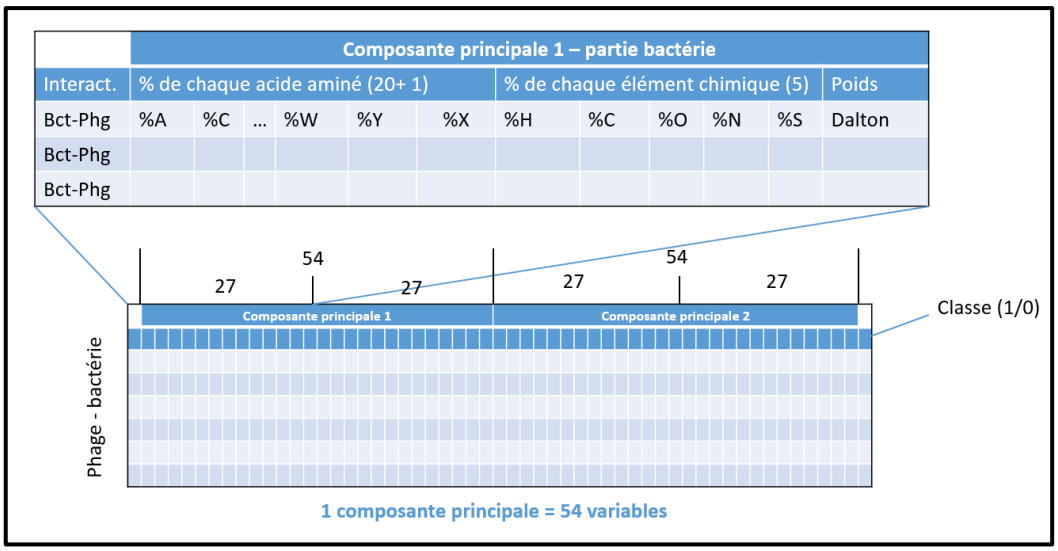


Figure 28 Schéma set de données – séquence protéiques adapté de (Leite et al. 2016)

Deux scripts, décrits dans la section 5.2.2 (page 75), ont été créés afin de générer le set de données basé sur les séquences protéiques. Le premier, expliqué dans la section 5.2.2.1 (page 76), va, pour chaque séquence protéique de toutes les bactéries et bactériophages, calculer leurs poids ainsi que le pourcentage d’acides aminés et d’éléments chimiques Le deuxième, décrit sous la section 5.2.2.2 (page 79), applique la réduction de variables et génère le set de données. Toutes les données calculées sont enregistrées dans un tableau, illustré sur la Figure 29, créé dans la base de données « Phage_bact » et nommé « AciAmin_C_WEIGHT ». Ce dernier est composé de 31 colonnes décrites ci-dessous :

- *AACW_Id* : identifiant du registre ;
- *idCell* : identifiant de la cellule ;
- *bool_bacteria* : *boolean* indiquant s'il d'agit d'une bactérie (1) ou d'un bactériophage (0) ;
- *idProtein* : identifiant de la protéine ;
- 21 colonnes faisant références aux pourcentages des 21 acides aminés (*Perc_A, Perc_C, Perc_D, …*) ;
- 5 colonnes faisant référence aux pourcentages des 5 éléments chimiques (*Perc_C_C, Perc_C_H, Perc_C_O*, …) ;
- *Weight* : poids de la protéine.

AACW_Id	idCell	bool_Bacteria	idProtein	Perc_A	...	Perc_X	Perc_C_C	...	Perc_C_S	Perc_Weight
282983	0	1	lcl|CP014196.1_prot_AUT26_00005_1	0.0887949	...	0	0.252081	...	0.00128972	61489
282984	0	1	lcl|CP014196.1_prot_AUT26_00010_2	0.0989305	...	0	0.251596	...	0.00152022	46989
282985	0	1	lcl|CP014196.1_prot_AUT26_00015_3	0.122449	...	0	0.250896	...	0.00258861	36708
282986	0	1	lcl|CP014196.1_prot_AUT26_00020_4	0.110831	...	0	0.250141	...	0.00126618	50370
282987	0	1	lcl|CP014196.1_prot_AUT26_00025_5	0.0978261	...	0	0.246891	...	0.00155473	22927
282988	0	1	lcl|CP014196.1_prot_AUT26_00030_6	0.089725	...	0	0.25043	...	0.00172032	88817
282989	0	1	lcl|CP014196.1_prot_AUT26_00040_8	0.144144	...	0	0.248414	...	0.000792812	26222
282990	0	1	lcl|CP014196.1_prot_AUT26_00055_9	0.127193	...	0	0.251225	...	0.000845023	86163
282991	0	1	lcl|CP014196.1_prot_AUT26_00060_10	0.178808	...	0	0.248868	...	0.00255956	36257
282992	0	1	lcl|CP014196.1_prot_AUT26_00065_11	0.0993151	...	0	0.251747	...	0.002644	38569

Figure 29 Extrait tableau « AciAmin_C_WEIGHT»

Sur la Figure 29, la colonne « Perc_X » est composée de valeurs « 0 » parce qu'il n'existe que 71 protéines pour lesquelles un ou plusieurs acides aminés n'ont pu être identifiés. Ce script a permis de traiter les 228'374 protéines qui composent les 42 bactéries et 1'065 bactériophages parmi les 2'130 paires bactérie-bactériophage.

4.5 Apprentissage automatique

Jusqu'à présent, nous avons analysé les données fournies et créé les sets de données qui sont maintenant prêts à être utilisés pour l'apprentissage automatique. Á présent, suivant la méthodologie de (Fayyad et al. 1996), nous allons déterminer le type d'algorithme à utiliser, choisir les algorithmes à implémenter et déterminer les paramètres avec lesquels lancer les algorithmes. La configuration des sets de données générés et l'analyse des résultats obtenus sont décrits dans le chapitre 6 (page 91).

Dans le domaine de l'apprentissage automatique, il n'existe aucune formule pour déterminer à l'avance l'algorithme et les configurations permettant d'obtenir le meilleur résultat. C'est pourquoi, il a été nécessaire d'élaborer un script qui puisse exécuter plusieurs algorithmes d'apprentissage automatique avec diverses configurations afin de trouver celles qui permettent d'obtenir les meilleures performances. Le script développé incorpore la validation croisée stratifiée de 10 *fold* qui, comme décrit dans la section 2.3.2.5 (page 22), permet d'utiliser toutes les paires bactérie-bactériophage pour l'entraînement et garantit que toutes passent par le set de validation en couvrant ainsi toutes les données.

Parmi les algorithmes d'apprentissage étudiés dans l'état de l'art, les algorithmes implémentés furent : le K-plus proches voisins, les forêts aléatoires, le MVS et le NN. Les trois premiers algorithmes ont été implémenté avec l'API scikit-learn dans sa dernière version stable, 0.17.0. Réseaux de neurones (NN) est disponible mais dans une version encore en développement, 0.18.0 (Scikit Learn 2016), c'est pourquoi il a été utilisé une API développée à l'HEIG-VD ne contenant que l'algorithme NN (Satizabal & Perez-Uribe 2016). De la même façon qu'il est possible de paramétrer plusieurs configurations pour chaque algorithme d'apprentissage, le script peut également traiter plusieurs sets de données.

Avant d'entraîner les divers modèles, toutes les données passent par une normalisation, afin d'éviter que certaines caractéristiques puissent en dominer d'autres et affecter l'entraînement. L'une des plus grandes différences de valeurs se voit entre les pourcentages, variant entre [0-1] et le poids des protéines, variant entre [10'505-1'084'455] Dalton. Tous les registres sont également réordonnés aléatoirement avant que ne soient créés les différents *folds,* éliminant ainsi un quelconque type de suite logique pouvant exister au moment de la génération de la base de données « phage_bact » comme par exemple, l'ordonnancement des registres par id de bactérie.

Le Tableau 9 décrit les paramètres qui doivent être configurés pour chacun des algorithmes programmés.

Tableau 9 Description des paramètres des algorithmes d'apprentissage automatique

Algorithme	Paramètre	Objectif
k-plus proches voisins	Nombre de voisins	Nombre de voisins à considérer pour la classification. En cas d'égalité, l'algorithme considère la classe du premier voisin pour lequel la distance a été calculée
	Distance	Permet d'indiquer le type de distance à utiliser parmi : la Minkowski, Euclidienne ou Manhattan (2 – page 82)
Forêts aléatoires	Nombre d'arbres	Nombre d'arbres à créer pour la classification
	Nombre de registre minimum par feuille	Nombre minimum de registres par feuille pour arrêter l'entraînement de l'arbre
	Profondeur maximum	Profondeur maximum des arbres avant d'arrêter l'entrainement de l'arbre
Machine à vecteur de support	Valeur de penalty	Tolérance à l'erreur de classification
	Valeur de l'élan	Vitesse d'apprentissage
Réseaux de neurones	Nombre de neurones	Nombre de neurones dans la couche cachée
	Époques	Nombre d'itérations
	Taux d'apprentissage	Vitesse d'apprentissage de la fonction
	Élan	Élan utilisé pour éviter les minima locaux

Pour chaque set de paramètres avec chaque algorithme d'apprentissage et chaque set de données, les résultats suivants ont été enregistrés :

- Á chaque itération de la validation croisée ont été enregistrés :
    - Les résultats espérés et prédits par le modèle pour les paires bactérie-bactériophage du set de validation et d'entrainement ;
    - Le nombre de vrais positifs, vrais négatifs, faux positifs et faux négatifs ;
    - L'*accuracy*, la précision, le *recall*, la spécificité, la sensibilité et la mesure F1 ;
- Les moyenne, médianes, variances et écart-type des résultats de la validation croisée pour :
    - Les vrais positifs, vrais négatifs, faux positifs et faux négatifs ;
    - L'*accuracy*, la précision, le *recall*, la spécificité, la sensibilité et la mesure F1 ;

Tous les résultats des expériences (*runs*) sont enregistrés dans des fichiers CSV qui ont permis de créer les graphiques illustrés dans le chapitre 6 (page 91).

L'erreur de la validation croisée est calculée à travers la moyenne des erreurs des ***k*** itérations. En ayant les prédictions faites sur les sets de validation et celui de test pour chacun des *folds* et des quatre algorithmes, il a été créer un script[20] qui :

- Implémente le système de vote et détermine la classe pour chacune des prédictions ;
- Calcule la spécificité et sensibilité moyennes des 10 *folds* de chacune des bactéries ;
- Génère les graphiques de résultats présentés dans la section 6.2 (page 104).

[20] Ce dernier script et tous ceux permettant la génération des graphiques n'ont pas été décrits dans ce document mais peuvent être demandés par e-mail.

5 Implémentation – approche technique

Dans ce chapitre sont décrits tous les aspects techniques et technologiques adoptés tout au long du projet, les aspects conceptuels étant décrits dans le chapitre 4 (page 35). Dans la section 5.1 (page 59) sont décrits les composants utilisés, les configurations qui ont dû être effectuées ainsi que les protocoles de communications utilisés. La section 5.2 (page 60) décrit la façon dont ont été développés les scripts nécessaires à l'extraction des variables, dont les aspects conceptuels sont décrits dans la section 4.4 (page 45). Pour terminer, la section 5.3 (page 82) décrit comment ont été implémentés les scripts contenant l'algorithme de prédiction dont le design est décrit dans la section 4.5 (page 55).

Pour faciliter la compréhension de l'implémentation des scripts, des diagrammes de flux qui représentent le fonctionnement de ceux-ci, ont été utilisés. Pour tous les scripts sont donnés : une explication des cycles est des fonctions développées, des bases de données utilisées et des librairies jugées importantes.

5.1 Composants

Pour la réalisation du projet, une machine virtuelle avec Ubuntu 14.04 installée dans les serveurs de la HEIG-VD a été mise à disposition. La machine fournie contient 8GB de mémoire vive, un processeur de 8 cœurs avec une vitesse de 2.7 Ghz et 18 Gb de mémoire sur le disque dur principal. La connexion à la machine virtuelle s'est effectuée via SSH et via FTP pour le transfert des fichiers. Dans la Figure 30 est représenté le diagramme de composants utilisé tout au long du projet.

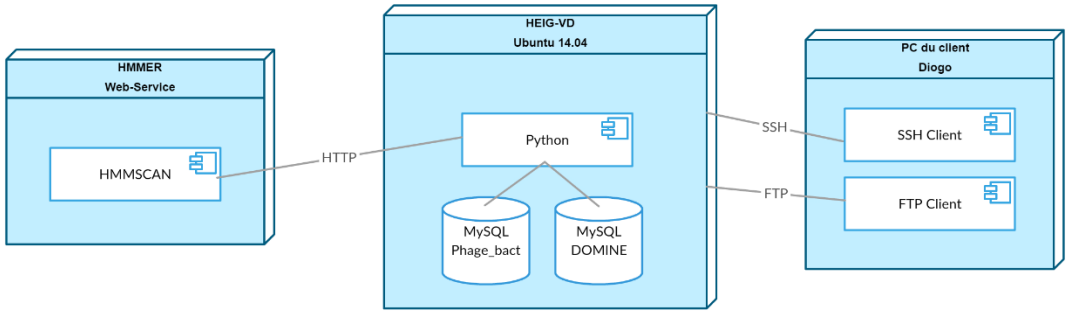


Figure 30 Diagramme de composants

Au début du projet, il a été nécessaire d'installer le système de gestion de base de données « MySql » dans sa version gratuite afin d'y importer les deux bases de données, « Phage_bact et DOMINE », fournies et décrites sous la section 4.3 (page 39). L'utilisation du service web « HMMER » est utilisé par le script décrit dans la section 5.2.1.1 (page 61) et a servi à obtenir les domaines qui composent les protéines. Les protocoles de communications SSH et FTP ainsi que les configurations nécessaires à la connexion à la machine étaient déjà tous configurés, étant uniquement nécessaire d'installer un client FTP et SSH pour la communication du côté client.

La machine virtuelle a été nécessaire dû au fait que certains des scripts ont pris plusieurs jours à tourner, notamment le script de recherche de domaines, décrit dans la section 5.2.1.1 (page 61), et celui qui contient l'algorithme de prédiction, décrit dans la section 5.3 (page 82).

5.2 Extractions des variables

Sous cette section est expliqué comment ont été développé les scripts permettant d'extraire les deux types de variables décrites sous la section 4.4 (page 45). La première section décrite ci-dessous, 5.2.1 (page 60), explique les procédés de développement qui ont dû être programmés pour l'extraction des variables basées sur les domaines d'interactions entre protéines. La section 5.2.2 (page 75) explique les scripts développés pour les variables basées sur les séquences protéiques.

5.2.1 Variables basées sur les domaines

Dans cette section sont décrits les 6 scripts qui ont dû être développés pour extraire les variables se basant sur les séquences protéiques. Le premier script, 5.2.1.1 (page 61), montre le procédé qui a permis d'analyser toutes les séquences protéiques de toutes les bactéries et bactériophages afin de trouver les domaines. Cette recherche a nécessité de recourir à un service-web dont le serveur était souvent indisponible. Un deuxième script qui ré-analyse les séquences n'ayant pas pu être analysée précédemment a donc dû être créé, 5.2.1.2 (page 63). Après avoir déterminé les domaines, un script capable d'indiquer, parmi ceux-ci, quels sont ceux qui interagissent et calculant le score IPP a été développé, 5.2.1.3 (page 65). Le script suivant,

5.2.1.4 (page 69), s'occupe de compter la fréquence des scores IPP pour chaque paire bactérie-bactériophage. L'un des objectifs proposés fut d'élaborer un script permettant la construction de sets de données qui ne soit dépendant d'aucun accès à internet ou une quelconque base de données. Pour la réalisation de cette tâche, un script qui sauvegarde les fréquences des scores IPP a donc été conçu, 5.2.1.5 (page 71). Le dernier script permet de générer les sets de données basé sur les domaines des séquences protéique et ne nécessite que le fichier enregistré précédemment, 5.2.1.6 (page 71).

5.2.1.1 Script de recherche de domaines

Ce premier script va, comme décrit sous la section 4.4.1.1 (page 47), chercher tous les domaines disponibles dans toutes les séquences protéiques de chaque bactérie et bactériophage. Dans la Figure 31 est illustré le diagramme de flux représentant le fonctionnement du script développé.

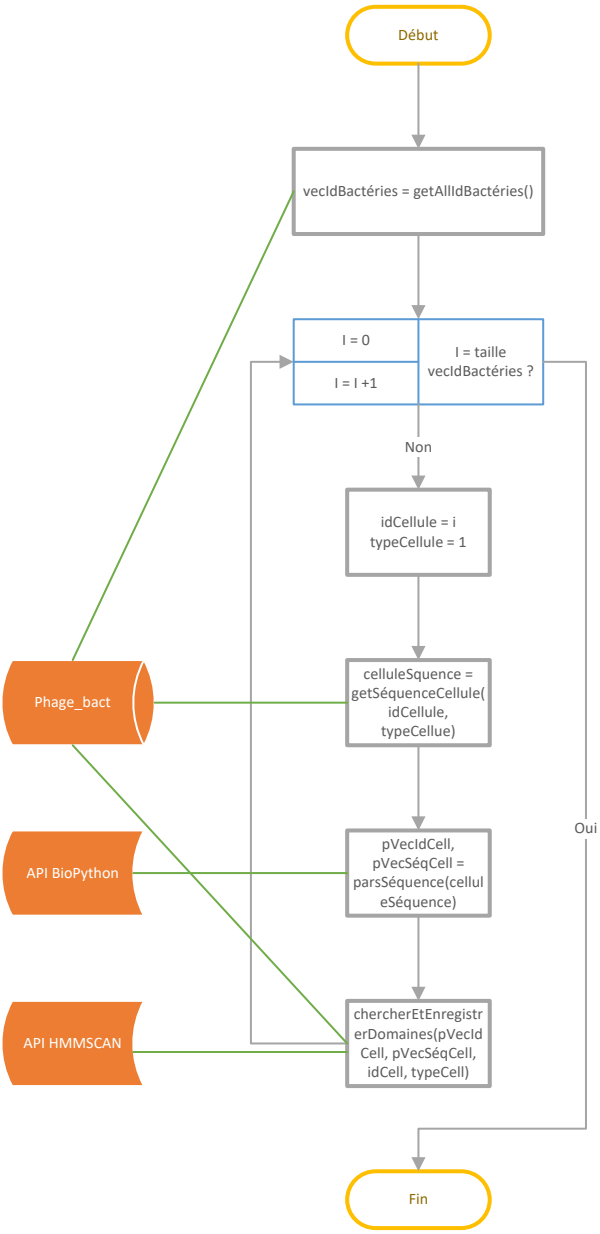


Figure 31 Diagramme de fluxe du script de recherche de domaines

Lorsque le script est lancé la première fonction appelée, « getAllBacteries() », va consulter la base de données « Phage_bact » et retourner un vecteur contenant tous les « ids » de toutes les bactéries. De suite, le vecteur contenant les « ids » est parcouru et pour chaque id il va :

1. Consulter la base de données « phage_bact » afin d'obtenir la séquence protéique de la cellule avec l'id « idCellule » du type « typeCellule » en appelant la fonction « getSequenceCellule(idCellue, typeCellule) ». Celle-ci retourne la séquence au format multi-fasta (CelluleSequence) ;
2. Appeler la fonction « parsSequence(CelluleSéquence) » qui reçoit comme paramètre la séquence protéique multi-fasta et, utilisant l'API bioPython, va la *parser* pour en

retourner deux vecteurs : le premier « pVecIdCell » contenant les id des protéines et un deuxième « pVecSéqCell » avec uniquement les séquences protéiques de celles-ci.

3. La dernière fonction appelée, « chercherEtEnregistrerDomaines(pVecIdCell, pVecSeqCell, idCell, tYpeCell) », reçoit comme paramètres le vecteur contenant les Ids des protéines « pVecIdCell », le vecteur contenant les séquences protéiques « pVecSeqCell », l'id de la cellule « idCell », un booléen indiquant s'il s'agit d'une bactérie (1) ou d'un bactériophage (0). Pour chaque id de séquence protéique celui-ci va :
    - Vérifier qu'elle ne se trouve pas déjà dans le tableau « PROTDOM » base de données « Phage_bact » ;
    - Si celle-ci ne s'y trouves pas, l'API HMMSCAN est consultée à travers la technologie REST afin de retourner un fichier *Json* contenant l'information sur la séquence, effectuer le parsage du fichier et insérer les domaines trouvés dans le tableau « PROTDOM » de la base de données « Phage_bact », pour chaque domaine trouvé un nouveau registre est inséré. Dans le cas où la protéine ne contiendrai pas de domaines, le script insert un registre avec, dans le champ « DomainAcc », l'information « --NA-- ».

Avant de lancer le script sur la machine mise à disposition, celui-ci a été lancé en local pendant une journée. Il s'est avéré que parfois l'API HMMSCAN ne répondait pas dû au temps que le serveur prenait a calculé la séquence, ceci pouvait prendre jusqu'à 8 heures[21], ou parce que celui-ci se trouvait en manutention. Afin de résoudre ces problèmes, la bibliothèque utilisée pour consulté l'API HMMSCAN, urllib2, permet de définir un temps maximum d'attente « *time-out* » qui a été définit à 60 secondes. Dans le cas où le serveur est en manutention, il est alors retourné une erreur indiquant qu'il n'a pas été possible d'accéder à l'API. Lorsqu'une de ces deux exceptions est détectée, la séquence est enregistrée dans le tableau « PROTDOM » de la base de données « Phage_bact » avec, dans le champs « DomainAcc », l'information « --PN -- » et la séquence dans le champs « ProtSeq ».

Le même script a été lancé pour les bactériophages, la seule différence réside dans la première méthode qui va chercher les id de tous les bactériophages et non ceux des bactéries au lieu de ceux correspondants aux bactéries.

5.2.1.2 Script de recherche de domaines - actualisations des protéines non analysées

Comme décrit sous la section du chapitre précédant, 4.4.1.1 (page 47), lors de la recherche des domaines il se pouvait que le serveur de l'API HMMSCAN soit indisponible ou que celle-ci prenne énormément de temps à répondre. Le script décrit ci-dessous a pour objectif d'aller chercher les protéines qui n'ont pu être analysées afin de les ré-analyser. Le diagramme de flux représenté sur la Figure 32 permet de voir la façon dont fonctionne le script développé.

[21] Il s'est avéré que, parfois, en lançant une deuxième requête, celle-ci obtenait directement une réponse.

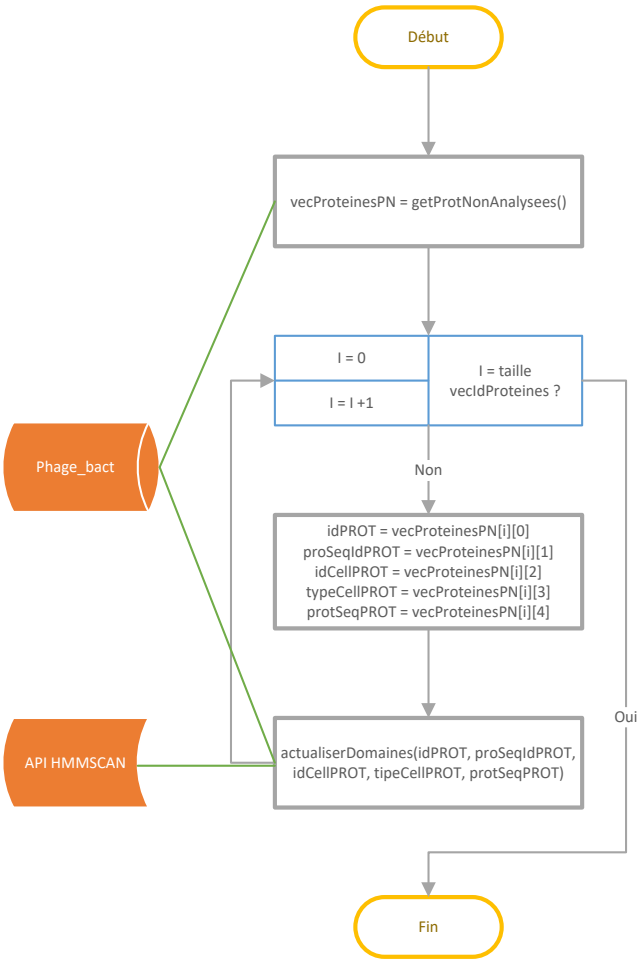


Figure 32 Diagramme de flux du script d'actualisation des séquences non analysées

Lorsque celui-ci est exécuté, la fonction « getProtNonAnalysees() », consulte le tableau « PROTDOM » de la base de données « Phage_bact » afin de retourner tous les registres contenant l'information « --PN --» dans le champs « DomainAcc » sous forme d'une matrice composée de 5 colonnes qui représentent tous les champs du tableau « PROTDOM » sauf DomainAcc dû au fait que la protéine ne contient pas de domaine. Pour chaque ligne de la matrice le script va :

1. Copier les valeurs de toutes les colonnes de la matrice dans une variable séparée. L'id du registre dans la variable « idPROT », l'id de la protéine dans « protSeqIdPROT », l'id de la cellule figurant dans le tableau « Bacteria » ou « Phages » selon il s'agit d'une bactérie ou d'un bactériophage dans « idCellPROT », le type de la cellule dans « typeCellPROT » et la séquence protéique de la protéine dans « protSeqPROT » ;
2. Appeler la méthode « actualiserDomaines(idPROT, proSeqIdPROT, idCellPROT, typeCellPROT, protSeqPROT) » qui va, tout comme le script de recherche de domaines, consulter l'API HMMSCAN pour déterminer les domaines existant sur la protéine :
    a. Dans le cas où l'API répond correctement, c'est-à-dire envoie le fichier *Json* avant le *time-out* ou que le serveur ne soit inopérationnel, il va supprimer le registre de la protéine grâce à l'ID de celui-ci et soit :

i. La protéine contient un ou plusieurs domaines et pour chacun d'eux y insérer un registre dans le tableau « PROTDOM » ;
ii. La protéine ne contient aucun domaine et est inséré un registre contenant l'information « --NA-- » dans le champs « DomainAcc » dans le tableau « PROTDOM » ;

b. Dans le cas où l'API ne répond pas, alors le script passe cette protéine et n'effectue aucune modification des données.

Ce script a dû être lancé plus de 7 fois consécutivement dû aux mêmes problèmes qui ont surgit avec le premier script de recherche de domaines, le temps de réponse excède le *time-out* ou l'inoperabilité de l'API HMMSCAN afin que toutes les protéines puissent être analysées.

5.2.1.3 Script de détection des domaines connus pour interagir

Comme décrit sous la section 4.4.1.2 (page 48), après avoir obtenus tous les domaines de chaque protéine de toutes les bactéries et bactériophages, ce script va combiner, pour chaque pair d'interaction, toutes les protéines de la bactérie avec celles du bactériophage y déterminer quels sont les domaines qui interagissent, calculer leurs scores et enregistrer ces derniers dans le tableau « Score_interactions ». La Figure 33 illustre le fonctionnement du script élaboré pour cette tâche.

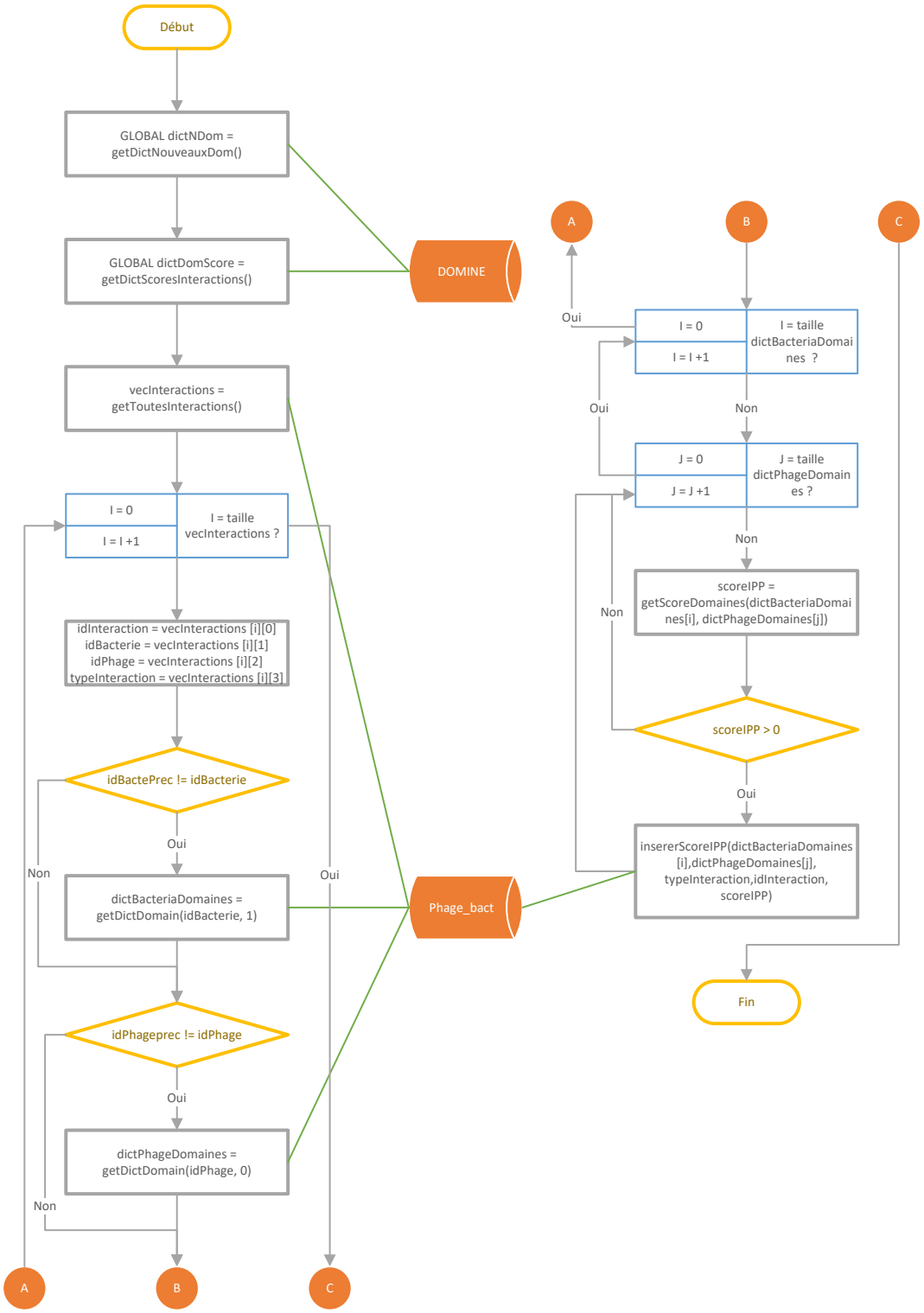


Figure 33 Diagramme de flux du script qui calcule les scores d'interactions pour chaque IPP

La fonction « getDictNouveauxDoms() » consulte le tableau « PFAM » de la base de données « DOMINE » et retourne tous les domaines qui ont été actualisés sous forme d'un dictionnaire. Sous la Figure 34 il est possible de voir un extrait des informations qui sont retournées. Un dictionnaire est composé de plusieurs tuples composés, chacun, d'une clé, élément à gauche

des « : » et d'une valeur, élément à droite. Dans le cas du script, la clé correspond à l'ancien domaine et la valeur à qui est venu le substituer.

```
{
  'PF00138': 'PF00139',
  'PF00442': 'PF00443',
  'PF00461': 'PF00717',
  'PF00597': 'PF06808',
  'PF00785': 'PF00989',
  '… : …' ,
}
```

Figure 34 Dictionnaire contenant les domaines actualisés

La fonction « getDictScoresInteractions() » retourne la somme des scores de chaque interaction entre deux domaines sous forme d'un dictionnaire où il est possible de consulter un extrait sous la Figure 35. Lorsqu'elle est appelée, la fonction consulte le tableau « Interactions » de la base de données « DOMINE » et y retourne tous les registres. Pour chaque registre la fonction somme les scores obtenus par les 15 sources d'informations et crée un dictionnaire où à chaque clé correspond deux domaines connus pour interagir et chaque valeur le score total obtenu pour cette interaction.

```
{
  'PF00017|PF03372': 1,
  'PF01327|PF02224': 0,
  'PF01565|PF09129': 2,
  'PF03945|PF09131': 2,
  'PF00076|PF00389': 1,
  '… : …' ,
}
```

Figure 35 Dictionnaire contenant les scores des interactions entre deux domaines

La fonction « getToutesInteractions » retourne une matrice contenant toutes les interactions positives et négatives présentes dans les tableaux « Interactions » et « Negativ_interactions » de la base de données « Phage_bact », c'est-à-dire les paires d'interaction bactérie-bactériophage ainsi que la classe[22]. Pour chaque ligne de la matrice le script va :

1. Copier les valeurs de chaque position dans un variable séparée. L'id du registre dans la variable « idInteraction », l'id de la bactérie dans « idBacterie », l'id du bactériophage dans « idPhage », le type d'interaction dans « typeInteraction » ;

[22] Classe : information concernant l'interaction (1 -> positive – 0-> négative)

2. Vérifier si l'id de la bactérie actuelle est le même que celui de la bactérie précédente et s'il est différent alors appeler la fonction « getDictDomain(idBacterie , 1), recevant comme paramètre l'id de la cellule et la valeur 1 indiquant qu'il s'agit d'une bactérie, qui va chercher toutes les protéines de la bactérie et retourner un dictionnaire contenant les protéines ayant au moins un domaine. Sous la Figure 36 il est possible de voir le type d'information contenue dans le dictionnaire retourné où les clés représentent les id des protéines et les valeurs un vecteur contenant tous les domaines de la protéine. Cette information est obtenue à travers la consultation du tableau « PROTDOM » de la base de données « Phage_bact » ;
3. Faire de même que l'étape précédente (2) mais pour les bactériophages ;
4. Combiner toutes les protéines de la bactérie avec celles du bactériophage et pour chaque possibilité :
    a. Vérifier pour chaque paire de domaines s'ils disposent de domaines actualisés, en consultant le dictionnaire « dictNDom » et calculer la somme des scores des interactions entre tous les domaines des deux protéines grâce au dictionnaire « dictDomScore ». S'il s'avère qu'un des deux ou les deux domaines disposent d'actualisation, il est alors fait la moyenne des scores d'interactions entre ces derniers ;
    b. Additionner les scores de l'interaction entre les deux protéines au tableau « Scores_interactions » dans la base de données « Phage_bact » si ce dernier est supérieur à 0.

```
{
  'lcl|CP014196.1_prot_AUT26_09140_1789': ['PF00392', 'PF07729'],
  'lcl|CP014196.1_prot_AUT26_04685_915': ['PF00413'],
  'lcl|CP014196.1_prot_AUT26_02180_426': ['PF00441', 'PF02771', 'PF02770'],
  'lcl|CP014196.1_prot_AUT26_01770_347': ['PF06262'],
  'lcl|CP014196.1_prot_AUT26_01355_264': ['PF00494'],
  … : …
}
```

Figure 36 Dictionnaire contenant les protéines et respectifs domaines d'une cellule

Garder l'information dans les dictionnaires évite au script de consulter la base de données pour vérifier si chaque domaine dispose d'une actualisation et obtenir le score d'interaction entre deux domaines. Cependant, ceci n'est possible que lorsqu'il n'existe qu'une petite quantité de données, dans ce cas 62 domaines actualisés et un peu plus de 26'000 interactions de domaines, existent. Consulter une base de données est une opération longue qui passe par plusieurs étapes dont : créer et ouvrir une connexion, lancer la requête, fermer la connexion et traiter les

données reçues. Toutes ces étapes rendent le script lent sachant qu'il aurait fallu exécuter ces instructions pour chaque IPP et qu'il y en existe plus de $5{,}38*10^{8}$ [23].

5.2.1.4 Script d'élaboration des histogrammes

Comme décrit sous la section 4.4.1.3 (page 50), ce script va, pour chaque interaction entre une bactérie et un bactériophage consulter le tableau « Scores_interaction » afin de trouver les différents scores enregistrés et compter leurs fréquences. Le résultat est enregistré dans le tableau « QtdScores ». La quantité d'interactions ayant un score 0 est la différence entre le nombre d'IPP d'une interaction et la somme du nombre d'IPP ayant un score supérieur ou égal à 1. La Figure 37 décrit le fluxogramme du script élaborer pour cette fonction.

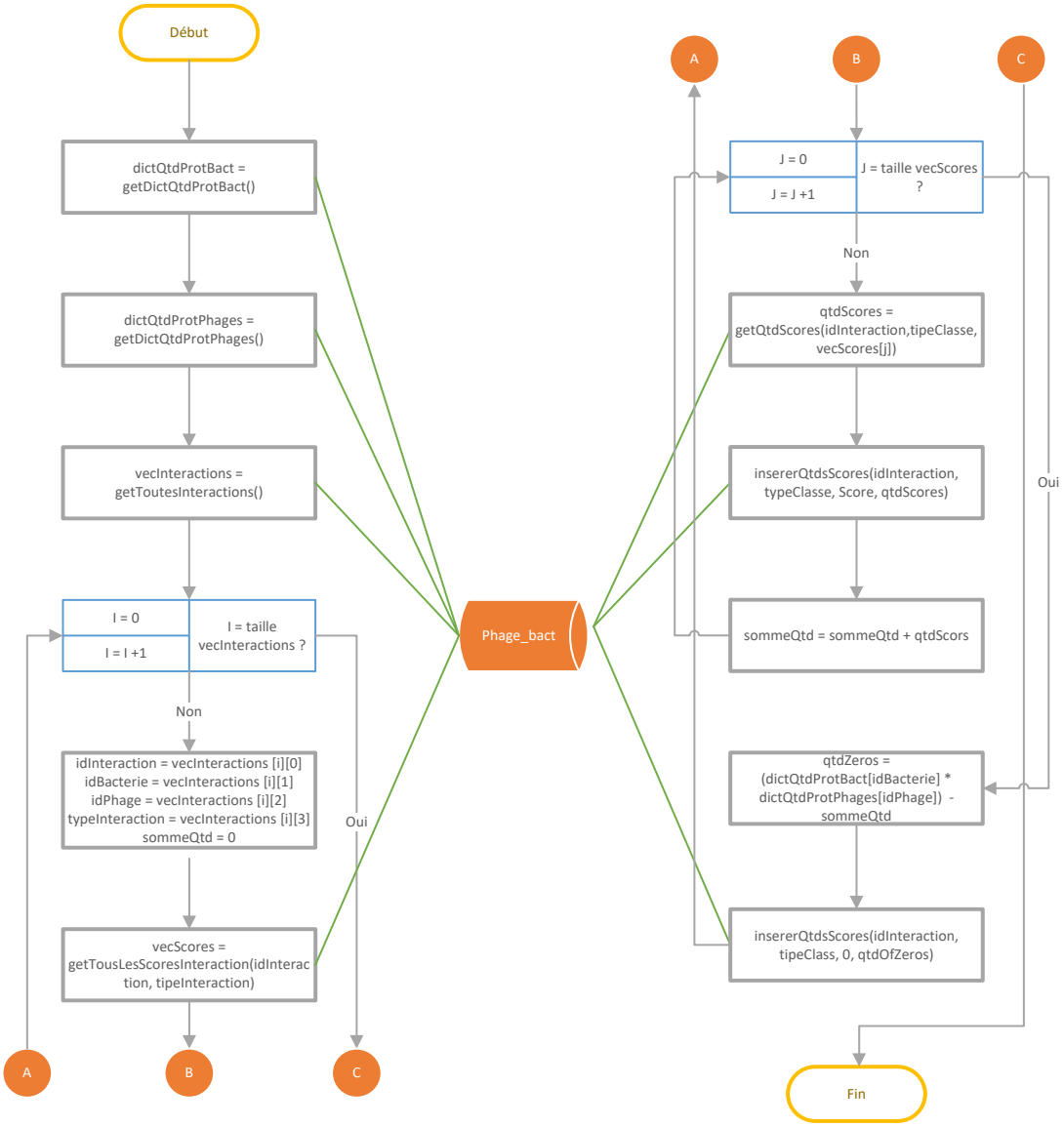


Figure 37 Diagramme de flux du script qui compte la fréquence des scores

[23] Corresponds au même nombre d'IPP indiqué sous la section 4.4.2 (page 84)

Les fonctions « getDictQtdProtBact() » et « getDictQtdProtPhage() » consultent la base de donnée « Phage_bact » retournent les séquences protéiques de chaque cellule au format multi-fasta, comptent le nombre de protéines et retournent, chacune d'elles, un dictionnaire, illustré sous la Figure 38, où les clés représentent les IDs des cellules et les valeurs le nombre de protéines correspondant.

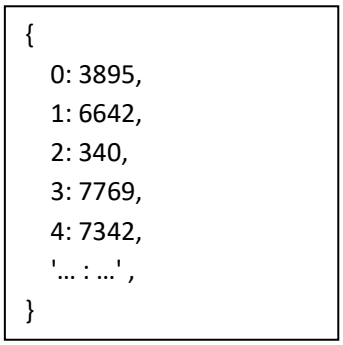


Figure 38 Dictionnaire contenant le nombre d'IPP pour chaque cellule

La fonction « getToutesInteractions », comme pour le script décrit sous la section 5.2.1.3 (page 65), retournes les informations concernant les interactions sous forme d'une matrice. Cette dernière contient l'ID de l'interaction, l'ID de la bactérie, l'ID du bactériophage et le type d'interaction, 1 pour les interactions positives et 0 pour les négatives. Pour chaque ligne de la matrice le script va :

1. Copier les valeurs de chaque position dans un variable séparée. L'id du registre dans la variable « idInteraction », l'id de la bactérie dans « idBacterie », l'id du bactériophage dans « idPhage », le type d'interaction dans « typeInteraction ». Initialiser la variable sommeQtd à 0, celle-ci ira servir á calculer le nombre d'IPP ayant un score de 0 ;
2. La fonction « getTousLesSocresinteractions(idInteraction, typeInteraction), ayant comme paramètres un ID d'interaction ainsi que son type, retourne un vecteur contenant toutes les valeurs de score trouvé pour l'interaction *i* dans le tableau « Score_interactions ». Le score zéro est calculé en soustrayant la somme de tous les scores au nombre total d'IPP de l'interaction i.
3. Pour chaque score le script va :
    a. Appeler la méthode « getQtdScores(idInteraction, typeInteraction, vecScores[j]) qui reçoit comme paramètre l'ID, le classe et le score en cours de traitement, retourne le nombre de fois que le score apparaît au sein de l'interaction ;
    b. Exécuter la méthode « insererQtdsScores(idInteraction, typeClasse, vecScore[j], qtdScore) » recevant comme paramètres l'ID de l'interaction, le type d'interaction, le score actuel et le nombre de fois qu'il apparait, va insérer le nombre d'IPP dans le tableau « QtdScores » ;
    c. Additionner la quantité de scores trouvé à la variable sommeQtd ;
4. Calculer le nombre d'IPP ayant un score de 0 ;

5. Insérer le nombre d'IPP avec un score de 0 dans le tableau « QtdScores » en appelant le méthode « insererQtdsScores(idInteraction, typeClasse, 0, qtdScore) » passant en paramètre le score 0 et sa quantité calculé au point 4.

Après avoir exécuté le script, le tableau « QtdScore » contient les fréquences d'apparitions pour chaque score de chaque interaction entre une bactérie et bactériophage.

5.2.1.5 Script générant le fichier *pickle*

L'une des restrictions du projet consiste à élaborer un script permettant de générer des sets de données avec plusieurs types de configurations. Le script décrit sous la Figure 39 génère le fichier *pickle* qui est utilisé pour la génération des sets de données décrit sous la section 5.2.1.5 (page 71).

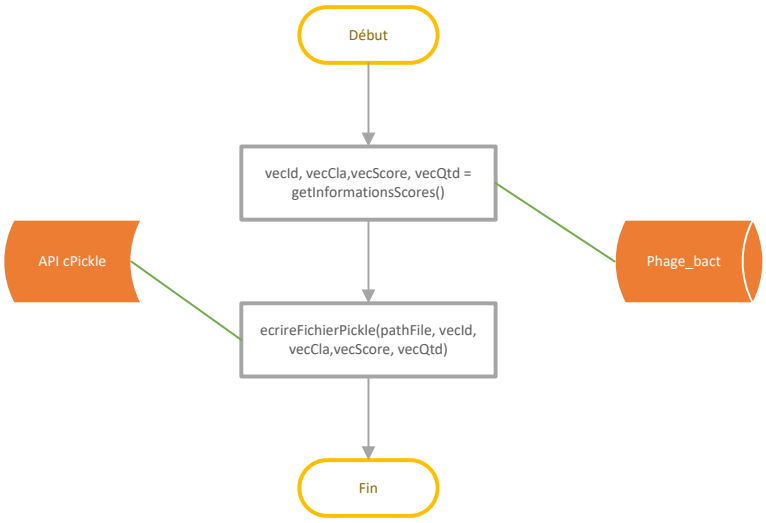


Figure 39 Diagramme de flux du script qui génère le fichier *pickle*

Lorsque le script est lancé, la fonction « getInformationsScores() » retourne toutes les informations du tableau « « qtdScores » de la base de données « Phage_bac » sous forme de quatre vecteurs correspondants aux colonnes comportant l'ID (vecId), le type (vecCla), la valeur du score (vecScore) et la quantité de fois que celui-ci apparait (vecQtd). La deuxième fonction « ecrireFichierPickle », qui reçoit comme paramètres le chemin où va être enregistre le fichier (pathFile) et les quatre vecteurs, va sérialiser l'information et créer le fichier *pickle*.

5.2.1.6 Script permettant de générer les sets de données

Comme décrit sous la section 4.4.1.4 (page 51), le script permettant de générer les sets de données ne doit dépendre d'aucune base de données ou connexion à internet. Le script créé, décrit sous la Figure 40, ne nécessite que du fichier *pickle* contenant les scores et respectifs fréquence de chaque interaction entre une bactérie et un bactériophage. Lorsque celui-ci est exécuté, il va demander à l'utilisateur de lui fournir les informations suivantes avant de générer les sets de données :

- Le chemin du fichier *pickle* ;

- Le chemin et nom du fichier où l'utilisateur prétend enregistrer le set de données
- S'il prétend normaliser les données ;
- S'il prétend indiquer le nombre de séparations ou indiquer la taille de celles-ci
- Indiquer le nombre ou taille des séparations. Il est indiqué le plus petit et plus grand score afin qu'il n'introduise pas de valeurs aléatoires.

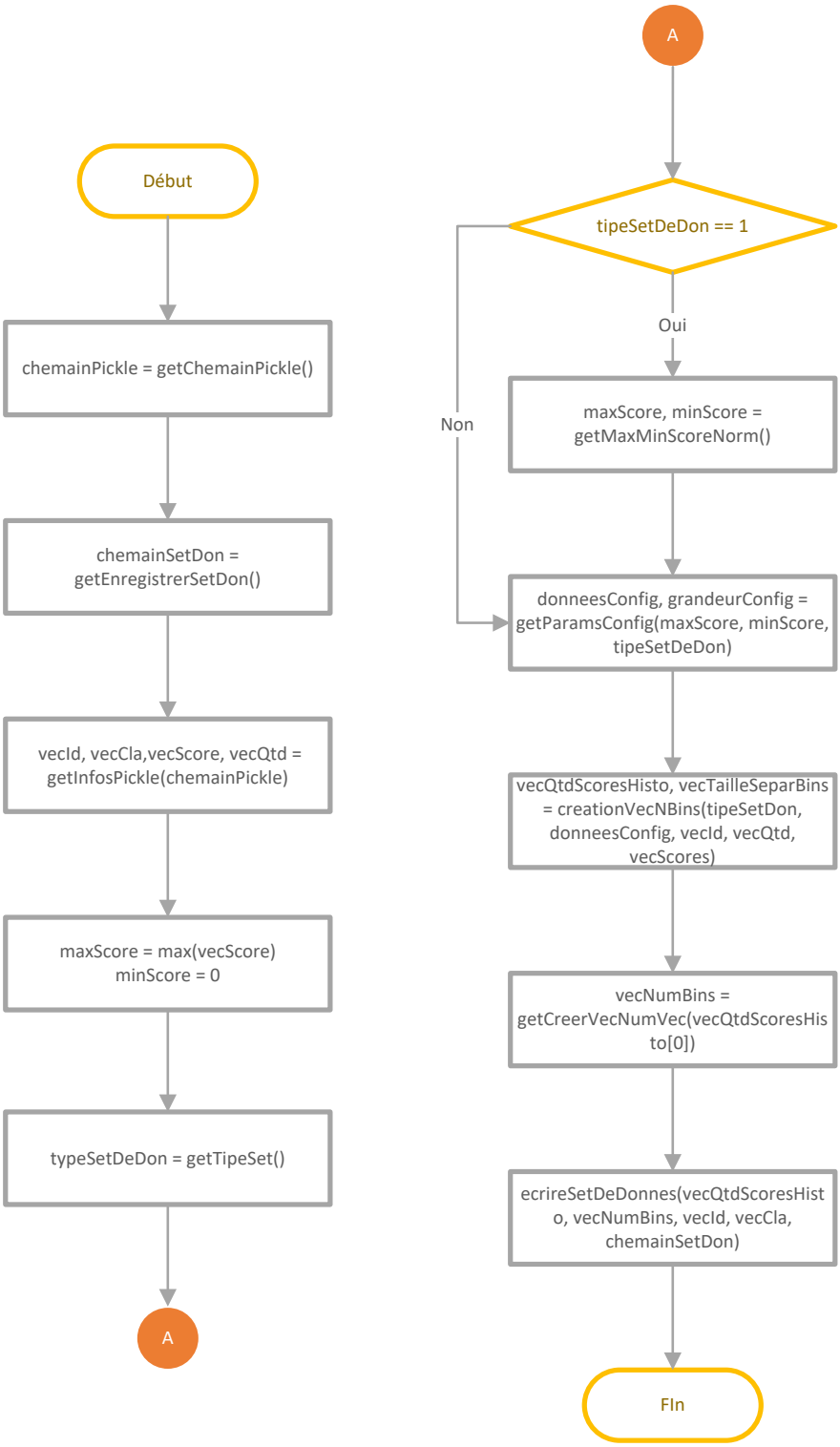


Figure 40 Diagramme de flux du script qui permet de générer les sets de données

La première partie du script sert à demander toutes les informations, citées ci-dessus, à l'utilisateur avant de, dans une deuxième partie, générer le set de données. Les deux premières fonctions, « getChemainPickle() » (A) et « chemainSetDon() » (B), demandent, à l'utilisateur d'insérer le chemin où se trouve le fichier *pickle* et où il veut enregistrer le set de données après

qu'il soit généré. Pour garantir que le chemin inséré correspond bien à un fichier *pickle*, le script vérifie son existence et seulement après appelle la deuxième fonction. La fonction « getInfosPicke(chemainPickle) » va charger les informations contenues dans le fichier et retourne quatre vecteurs contenants : l'id (vecId), le type (vecClass), le score (vecScore) et la fréquence de chaque score (vecQtd) de chaque interaction. Deux variables, maxScore et minScore, sont initialisées avec le plus grand et plus petit score. Le score de zéro est attribué aux protéines n'ayant aucune interaction, c'est pourquoi la variable minScore et initialisée avec 0. La fonction « getTypeSet() » (C) vise à demander à l'utilisateur s'il prétend, ou pas, normaliser les données avant de générer le set de données. Si l'utilisateur prétend utiliser les données normalisées, le script va alors calculer le score le plus petit mais supérieur à 0 ainsi que le plus grand. Pour terminer, la fonction « getParamsConfig(maxScore, minScore, typeSetDeDon) » demande à l'utilisateur quel type de set de donnée celui-ci prétend qu'il soit généré (D), tel comme illustré sur le diagramme de la Figure 25, ainsi que le nombre de bins ou taille de ceux-ci (E) selon s'il veut générer un set de données basé sur l'un ou sur l'autre. Si l'utilisateur prétend générer un set de données en se basant sur la taille des bins, juste après que celui-ci ait inséré la taille le script créé un vecteur contenant les tailles de chacun deux. Sur la Figure 41 sont illustrées les demandes d'informations à l'utilisateur avant le début de la génération du set de données.

```
A File path: /home/diogo/dataSetsC/gradesdictC2.p
B Save file: /home/diogo/dataSetsC/NB10.csv
C Use normal dataSet : (Yes = 1/ No = 0): 0
D Number of Bins (1) or Vector of Bins (2) :1
E Number of Bins (1-50) :10
```

Figure 41 Extrait de configuration pour la génération de set de données

La seconde partie du script consiste en la génération du set de données en accord avec les informations précédemment demandées à l'utilisateur. La fonction « creationVecNBins(typeSetDon, donneesConfig, vecId, vecQtd, vecScores) » va, pour chaque interaction :

1. Créer un vecteur avec autant de position que d'IPP et y insérer les respectifs scores ;
2. Si nécessaire, normaliser les données en divisant toutes les positions par le nombre d'IPP ;
3. Envoyer, à l'*Application Programming Interface* (API) *NumPi* un vecteur contenant les scores de l'interaction et un autre contenant la configuration des *bins* :
    a. Un vecteur avec une seule position signifiant la taille des *bins* ou
    b. Un vecteur contenant la taille de chaque *bins* ;

L'API *Numpi* retourne deux vecteurs, le premier contenant dans chaque position le nombre d'IPP, et dans le deuxième l'intervalle de chaque bin ou son numéro selon la génération se serait basée le nombre ou taille des *bins*. Les deux vecteurs retournés sont additionnés dans deux matrices qui seront retournées et utilisées dans les fonctions suivantes. Sous la Figure 42, il est possible de voir un extrait des valeurs retournées par chacun des vecteurs en ayant pour base la configuration de la Figure 41. Les vecteurs du dessus comportent 11 positions faisant références aux dix *bins*, le premier vecteur indique que le premier *bins* à une taille de [0 - 1.4[,

le deuxième [1.4-2.8[, …, le dernier [12.6-14[. Les vecteurs du dessous donnent l'information concernant la quantité des scores appartenant à chacun des intervalles.

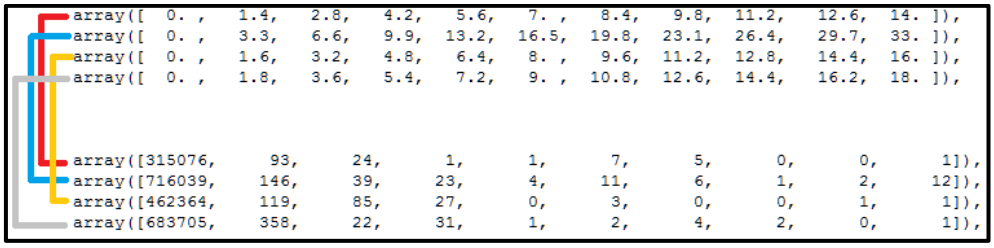


Figure 42 Exemple de données avec 10 *bins*

La fonction « getCreerVecNumVec(vecQtdScoresHisto[0]) » reçoit en paramètre la première ligne de la matrice contenant les fréquences d'apparitions des scores et retourne un vecteur composé d'autant de positions que de nombre de *bins* où pour chacune d'elles y est inséré le numéro du bin. Ce dernier est utilisé pour la création de la première ligne du set de données. La dernière fonction « ecrireSetDeDonnes(vecQtdScoresHisto, vecNumBins, vecId, vecCla, chemainSetDon) » génère le set de données et reçoit comme paramètres :

- La matrice contenant la fréquence de chaque *bin* ;
- Le vecteur contenant le numéro des *bins* ;
- Un vecteur avec tous les *ids* des interactions ;
- Un vecteur contenant le type de chaque classe (1 = interaction positive ; 0 = interaction négative) ;
- Le chemin où va être enregistré le set de données.

La Figure 43 illustre un extrait de set de données généré avec la configuration de la Figure 41

```
ID_interaction;0.0;1.0;2.0;3.0;4.0;5.0;6.0;7.0;8.0;9.0;Class_interaction
0;276221;221;80;5;2;1;1;2;4;8;1
1;276401;121;2;11;3;5;1;0;0;1;1
2;365261;40;1;3;1;2;1;0;0;1;1
3;371849;64;10;21;1;2;2;2;0;1;1
…
1051;1080746;287;22;45;6;12;15;0;18;6;0
1052;1442828;222;18;36;40;7;5;2;9;3;0
…
```

Figure 43 Extrait d'un set de données généré

5.2.2 Variables ayant pour base la composition chimique des séquences protéiques

Dans cette section sont décrits les 2 scripts qui ont permis de générer le set de données basé sur la composition chimique des séquences protéiques. Le premier, 5.2.2.1 (page 76), décrit comment ont été calculés les pourcentages d'acides aminés, composants chimiques et poids de

chaque protéine de toutes les bactéries et bactériophages. Le second, 5.2.2.2 (page 79), permet de combiner ces données pour toutes les IPP de chaque paire bactérie-bactériophage, et d'appliquer l'ACP afin de finalement générer le set de données.

5.2.2.1 Script qui calcule le poids, pourcentage d'acides aminés et de composants chimiques

Ce script va, pour chaque séquence protéique de toutes les bactéries et bactériophages :

- Compter le nombre d'acides aminés afin d'y calculer le pourcentage ;
- Compter le nombre de composants chimiques afin d'y calculer le pourcentage de chacun d'eux ;
- Calculer le poids de la protéine en calculant la somme des poids de tous les acides aminés.

Pour déterminer quels sont les composants chimiques et poids de tous les acides aminés, il a été utilisé l'information contenue dans un document technique fournit par les laboratoires *promega* (Promega 2010). Après avoir calculé tous les pourcentages d'acides aminés, composants chimiques et poids d'une protéine, les résultats sont enregistrés sous le tableau « AciAmin_C_WEIGHT » de la base de données « Phage_bact » afin d'être utilisés par le deuxième script décrit sous la section 5.2.2.2 (page 79). Dans la Figure 44 est illustré le fonctionnement du script développé.

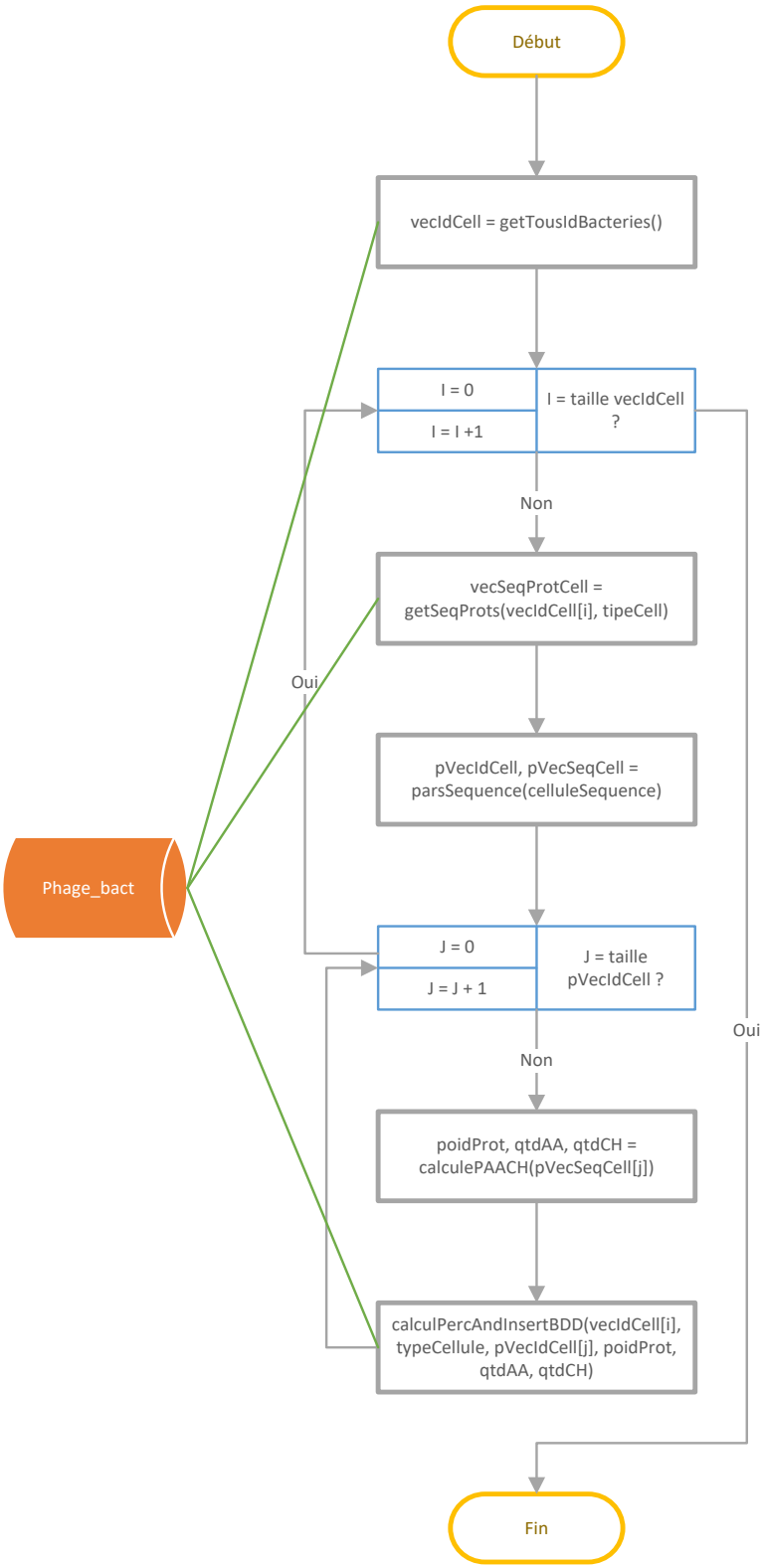


Figure 44 Diagramme de fluxe du script qui analyse les séquences protéiques

La première méthode « getTousIdBacteries() », consulte la base de données « Phage_bact » afin d'obtenir tous les ID de toutes les bactéries et retourne un vecteur contenant cette information. Pour chaque ID de bactérie le script va :

1. Consulter la base de données « Phage_bact » afin d'y retourner les séquences protéiques en appelant la fonction « getSeqProts(vecIdCell[i], typeCell) » et reçoit comme paramètres l'ID de la bactérie ainsi que le type de cellule (1 = bactérie, 2 = bactériophage). Celle-ci retourne les séquences protéiques de la cellule au format multi-fasta ;
2. Appeler la fonction « pVecIdCell, pVecSeqCell = parsSequence(celluleSequence) » qui reçoit comme paramètre les séquences protéiques, effectue le *parse* et retourne un vecteur contenant l'id de chaque protéine « pVecIdCell » et un deuxième vecteur « pVecSeqCell » avec les séquences protéiques. Pour chaque séquence le script va appeler la fonction :
    a. « calculePAACH(pVecSeqCell[j]) » qui va compter le nombre d'acide aminés, pour chacun d'eux, le nombre de composant chimique et calculer le poids de la protéine correspond à la somme du poids de tout ses acides aminés. Lorsque la fonction est appelée sont alors créés deux vecteurs, un premier de 21 positions correspondantes au nombre d'acides aminés (qtdAA) et un deuxième contenant 5 positions correspondantes aux 5 éléments chimiques qui composants les acides aminés (vecPercCaract). Le cycle *switch-case* n'existant pas dans le langage *python*, il a été créé 21 *ifs* correspondant aux 21 acides aminés ou chacun d'eux va incrémenter la position du vecteur correspondant à l'acide aminé d'un point et du nombre de chaque élément chimique dans le deuxième vecteur. Sous la Figure 45 est illustré le *if* correspondant à l'acide aminé A « Alanine ». La fonction retourne :
        i. Le poids ;
        ii. Un vecteur composé de 21 positions correspondant aux 20 acides aminés plus une lorsqu'il apparaît un X, voir section 4.3.3 (page 44) ;
        iii. Un vecteur de six positions correspondant à chaque composant chimique faisant parti des acides aminés [24];
    b. « calculPercAndInsertBDD(vecIdCell[i], typeCellule, pVecIdCell[j], poidProt, qtdAA, qtdCH) » qui reçoit en paramètre l'Id de la cellule (vecIdCell[i]), le type de cellule (typeCellule : 1-> Bactérie – 0 -> bactériophage), l'ID de la protéine (pVecIdCell[j]), le poids de la protéine (poidsProt), le vecteur contenant la quantité de chaque acide aminé (qtdAA) et le vecteur contenant la quantité d'éléments chimiques de la protéine. Cette fonction va :
        i. Calculer le pourcentage de chaque acide aminé en divisant toutes les positions du vecteur « qtdAA » par le nombre d'acides aminés qui composent la protéine ;
        ii. Calculer le pourcentage de composants chimique en divisant chaque position du vecteur « qtdCH » par le nombre total de composante chimique ;

[24] Pour effectuer le comptage des composants chimiques, ont été considérées les informations contenues dans le document (Promega 2010)

iii. Créer et insérer un registre dans le tableau « AciAmin_C_WEIGHT » (voir Figure 29) de la base de données « Phage_bact » contenant le poids, pourcentage de chaque acide aminé et composant chimique.

```
if acidAmin == 'A':
    qtdAA[0] = qtdAA[0] + 1
    vecPercCaract[0] = vecPercCaract[0] + 3
    vecPercCaract[1] = vecPercCaract[1] + 7
    vecPercCaract[2] = vecPercCaract[2] + 2
    vecPercCaract[3] = vecPercCaract[3] + 1
    vecPercCaract[4] = vecPercCaract[4] + 0
    valPoids = valPoids + 89
    isAcidAmin = True
```

Figure 45 Extrait du code *python* élaboré pour le comptage des acides aminés

Le vecteur « qtdAA », contenant 21 positions correspondantes aux 20 acides aminés plus *X*, est incrémenté de 1 à la position 0 correspondant à l'acide aminé Alanine. Le vecteur « vecPercCaract » composé de 5 positions correspondantes au nombre d'éléments de carbone (C – 0), d'hydrogène (H – 1), d'oxygène (O – 2), d'azote (N – 3) et de souffre (S – 4). Dans le cas où l'acide aminé actuel correspond à une lettre X, la variable « isAcidAmin » reste à « *false* » et le vecteur « qtdAA » est incrémenté à la position 21.

5.2.2.2 Script qui génère le set de donnée basé sur les séquences protéiques

Le script décrit sous la section précédente, 5.2.2.1 (page 76), s'occupe de calculer le poids, pourcentage d'acides aminés et de composants chimique de chaque protéine de toutes les bactéries et bactériophages et enregistré ceci dans le tableau « AciAmin_C_WEIGHT » de la base de données « Phage_bact ». Ce deuxième script va, pour chaque interaction entre une bactérie et bactériophage :

- Consulter le tableau « AciAmin_C_WEIGHT » afin d'obtenir les informations précédemment calculées concernant toutes les protéines de chacune des cellules ;
- Combiner toutes les protéines de la bactérie avec celles du bactériophage ;
- Appliqué l'ACP ;

Après avoir appliqué l'ACP à toutes les interactions, et comme décrit sous la section 4.4.2 (page 53), le script créer un set de données contenant 2130 registres et 108 variables. La Figure 46 illustre le fonctionnement du script qui génère ce dernier type de set de données.

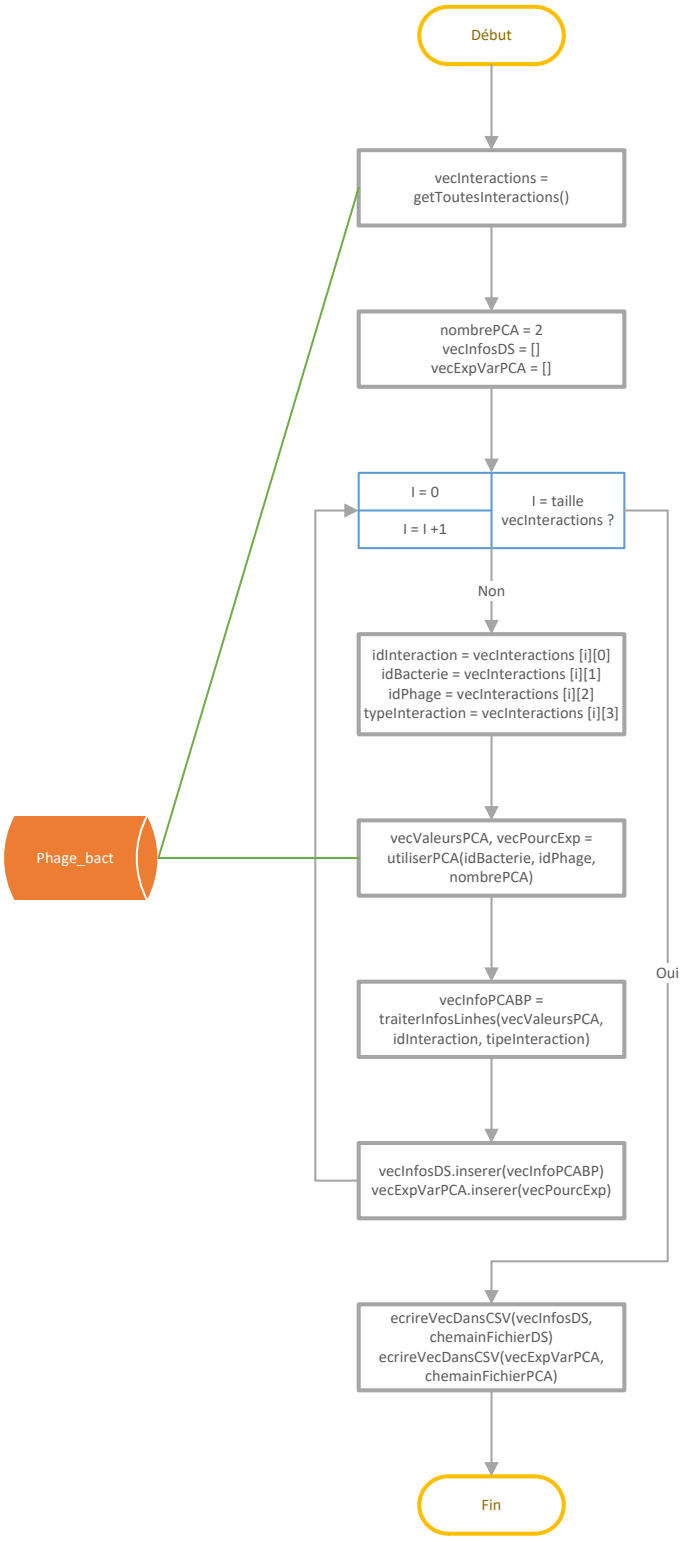


Figure 46 Diagramme de flux du script générant le set de données basé sur les séquences protéiques

La première fonction appelée « getToutesInteractions() » retourne une matrice contenant toutes les interactions positives et négatives dans la base de données « phage_bact ». Chaque ligne de la matrice est composée de l'ID de l'interaction, l'ID de la bactérie, l'ID du bactériophage et du type d'interaction (1-> positive ; 0-> négative). Après cela, le script crée une variable indiquant qu'il va générer un set de données avec deux composantes principales (nombrePCA), un vecteur où seront stockés les résultats de la réduction de chaque interaction (vecInfosDS) et un vecteur ou va être inséré la variance des données représentées par chaque PC (vecExpVarPCA). Pour chaque ligne de la matrice le script va :

1. Copier l'ID de l'interaction, l'ID de la bactérie, l'ID du bactériophage et le type d'interaction dans les variables suivantes : « idInteraction », « idBacterie », « idPhage » et « typeInteraction » ;
2. Appeler la fonction « utiliserPCA(idBacterie, idPhage, nombrePCA) » qui va :
    a. Consulter le tableau « AciAmin_C_WEIGHT » et retourner le pourcentage de tous les acides aminés, composants chimiques et poids de toutes les protéines appartenant à la bactérie, dans une matrice, et au bactériophage, dans une autre matrice, de l'interaction en cours de traitement ;
    b. Créer une matrice résultant de la combinaison des protéines de la bactérie et du bactériophage. Ceci donne une matrice composée de 54 colonnes et *X* lignes ou *X* corresponds aux paires entre les protéines des deux cellules ;
    c. Appliquer la réduction de dimensionnalité par l'analyse de composantes principales (ACP) avec 2 composantes principales (CP) en recourant à l'API « sklearn.decomposition ». Cette dernière retourne une matrice de 54 lignes ou chacune d'elle est composée de deux valeurs représentant les deux CP ;
    d. Consulter la variance des données expliquée par chaque CP ;
    e. Retourner la matrice qui contient les CP ainsi que le vecteur contenant la variance explicative de chaque variable ;
3. Traiter la matrice en appelant la fonction « traiterInfosLinhes(vecValeursPCA, idInteraction, typeInteraction) ». Cette méthode retourne un vecteur qui contient :
    a. Dans la première position, l'ID de l'interaction ;
    b. Chaque groupe de 54 variables suivantes corresponde aux valeurs d'une CP ;
    c. La dernière position contient le type d'interaction (1-> positive – 0-> négative) ;
4. Insérer chaque ligne retournée par la méthode précédante dans le vecteur « vecExpVarPCA » et la variance dans le vecteur « vecInfosDS » ;
5. Utiliser la fonction « ecrireVecDansCSV(vecInfosDS, chemainFichierDS) » pour écrire le contenu de la matrice « vecExpVarPCA » contenant les données des PC de chaque interaction entre une bactérie et un bactériophage dans fichier du type CSV. La même méthode est appelée afin d'écrire le contenu de la matrice, « vecInfosDS », possédant la variance d'information représentée pour chaque PC.

5.3 Algorithme d'apprentissage automatique

Cette section vise à expliquer le script contenant l'algorithme d'apprentissage automatique utilisé pour la prédiction des paires bactérie-bactériophage. Dû à la complexité du script, il a été décidé de diviser l'explication en trois sous-sections :

1. Description générale du script d'apprentissage automatique - Montre ce que fait le script pour chacun des sets de données ;
2. Exemple d'appel d'une fonction d'apprentissage automatique – Montre ce que fait le script pour chacune des configurations des quatre techniques d'apprentissage automatique (seul SVM est expliqué, mais la démarche est identique pour les autres ;
3. Fonctionnement du processus d'apprentissage automatique – explique l'implémentation de la partie d'entrainement, de validation et de test des modèles.

Ce script, existant en deux versions, a été utilisé tout au long du processus qui a servi à identifier les meilleures configurations pour chacun des quatre algorithmes implémentés, dont les résultats peuvent être consultés dans la section 6.1 (page 91), mais également pour le modèle final. Dans sa première version, le script ne contenait aucune fonction permettant de lancer les données de tests et fut utilisé pour l'analyse de plusieurs configurations. Dans la deuxième version de celui-ci, les fonctions permettant l'utilisation du set de données de test et l'enregistrement des résultats de prédiction obtenus pour chacun des *folds* ont été implémentées. Dans cette dernière version, les vecteurs de configuration de chacun des quatre algorithmes ne contiennent que les paramètres avec lesquels ont été obtenus les meilleurs résultats de performance, en accord avec les sets de données retenus, voir section 6.2 (page 104).

5.3.1 Script d'apprentissage automatique – description générale

Avant d'exécuter le script il est nécessaire d'initialiser les variables décrites sous le Tableau 10 afin que celui-ci puisse lancer toutes les expériences pour tous les algorithmes d'apprentissage automatique programmés. Dans la Figure 47 est illustré le fonctionnement du script qui test les diverses configurations pour les quatre algorithmes d'apprentissage automatique implémenté et qui, pour chacun d'eux, enregistre les performances obtenues.

Tableau 10 Variables de configuration du script d'apprentissage automatique

Variable	Informations
qtdKFolds	Nombre de *folds* désiré pour effectuer la validation croisée.
cheminDosDS	Chemin du dossier contenant les sets de données qui seront utilisés. Ce dossier ne doit contenir que les sets de données au format CSV dont les lignes corresponds aux différents registres et les colonnes aux variables.

Variable	Informations
cheminIdPosi	Chemin du fichier CSV contenant les identifiants de registres avec des interactions positives qui seront utilisée pour la validation de modèle final
cheminIdNeg	Chemin du fichier CSV contenant les identifiants de registres avec des interactions négatives qui seront utilisée pour la validation de modèle final
chemainDosierPerfs	Chemin du dossier ou vont être enregistrés les différents résultats
vecQtdKVoisins	Vecteur où chaque position correspond au nombre de voisins le plus proche à expérimenter
valeurDistance	Dans l'API *sklearn*, utilisée pour le développement du script, la distance est représentée par la formule de *minkowski* sous la formule 2. Dans quel cas, si : • q = 1 alors cela corresponds à la distance Manhattan ; • q = 2 alors cela corresponds à la distance Euclidienne.
vecQtdArbres	Nombre d'arbres qui vont être développé
vecQtdArbresMin	Nombre d'arbres minimum par feuille
vecProfMax	Profondeur maximum de l'arbre
vecValPenalty	Tolérance aux interactions mal classifiées
vecGamma	Paramètre gaussien. Plus la valeur est petite, moins le modèle sera apte à capter la complexité des données. Si elle est trop grande, il y a un risque de sur-apprentissage.
vecQtdNeurones	Nombre de neurones dans la couche intermédiaire
vecNbEpoques	Nombre de fois que NN est lancé pendant l'entrainement
vecTauxApprentissage	Poids avec lesquels le réseau va s'ajuster de forme à réduire l'erreur de prédiction.
vecElan	Paramètre permettant d'éviter les minimums locaux

$$dist(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{in} - x_{jn}|^q} \quad (2)$$

, où *i et j* sont deux registre et *Xin – Xjn* correspondent aux cordonnées des registres sur la dimension *n*

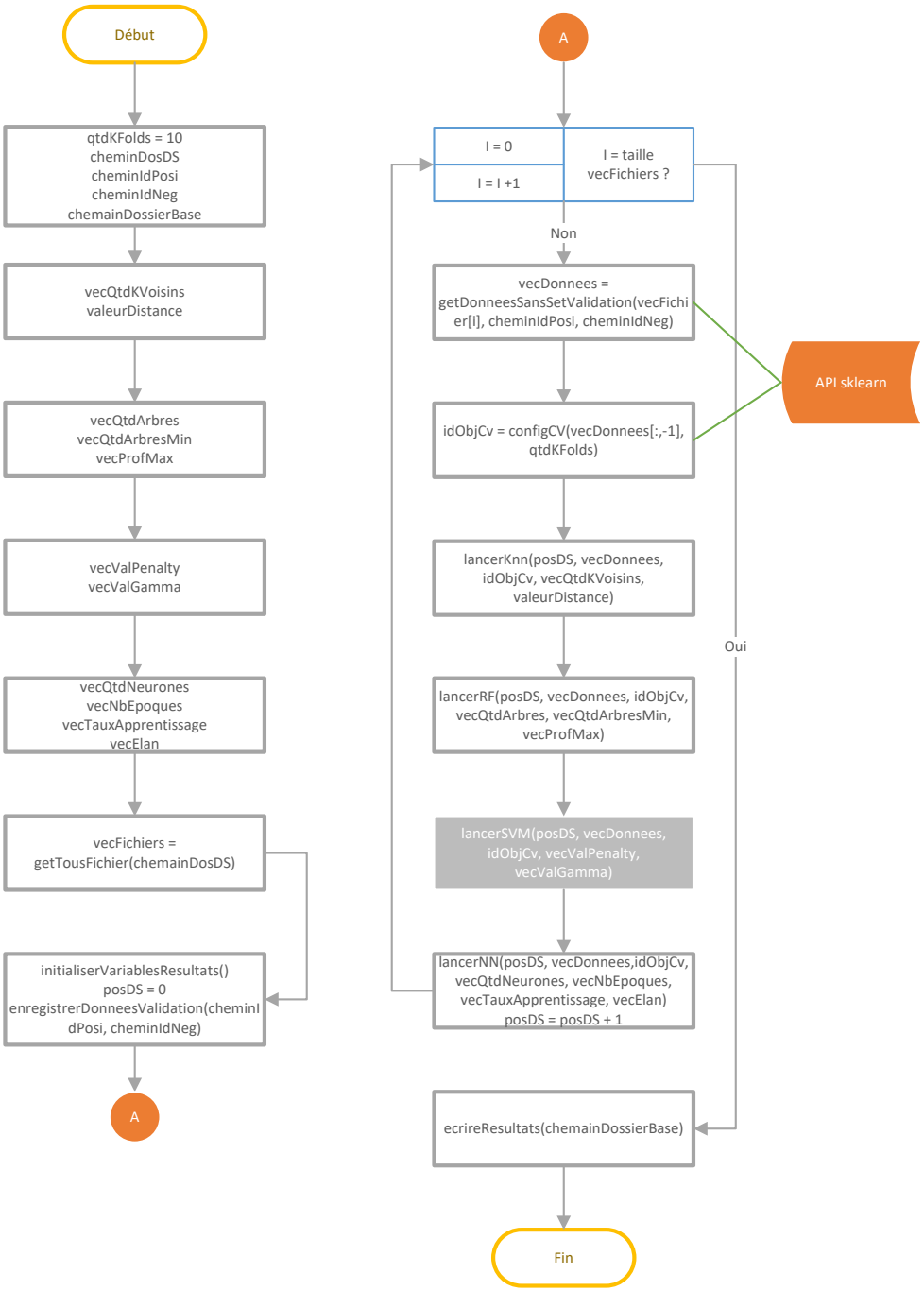


Figure 47 Diagramme de flux du script d'apprentissage automatique – vision générale

Après avoir initialisé les variables et lancé le script, la première fonction appelée « getTousFichier(chemainDosDS) » va chercher tous les sets de données contenu dans le dossier « chemainDosDS » et les ordonner par ordre alphabétique. Ordonner les fichiers est une tâche nécessaire afin de savoir quels résultats correspondent à quels sets de données étant donné que ceux-ci sont enregistrés au fur et à mesure que les sets de données sont utilisés. La

fonction « initialiserVariablesResultats() » va créer et initialiser les matrices contenant les résultats de performances, décrits sous la section 4.5 (page 55), de chaque modèles créés pour chaque set de données. La variable « posDS » est utilisée pour calculer les positions où doivent être enregistré les scores dans les différentes matrices. La fonction « enregistrerDonneesValidation(cheminIdPosi, cheminIdNeg) » garde les registre de test dans une matrice afin que ceux-ci puissent être testés sur chacun des modèles créés. Pour chaque set de données le script va :

1. Appeler la fonction « getDonneesSansSetValidation(vecFichier[i], cheminIdPosi, cheminIdNeg) » qui reçoit comme paramètres le chemin du set de données, du CSV contenant les indices des registres positifs et négatifs et retourne une matrice contenant les données normalisées sans les registres utilisés dans le set de test.
2. Appeler la fonction « configCV(vecDonnees, qtdKFolds) » reçoit les labels du set de données et le nombre de *folds* à créer. Avant de générer les différents *folds* le registres sont mélangés aléatoirement et il est garanti que tous les registres passent par un *fold* de validation. La fonction retourne un objet contenant les index des registres qui seront utilisés pour le set de validation et d'entrainement pour chaque *fold* ;
3. Appeler les fonctions qui vont exécuter les méthodologies d'apprentissage automatique en passant les vecteurs contenant les configurations à tester, le numéro du set de données et la matrice de données en paramètres. Pour chaque configuration testée, le script, enregistre les différentes mesures de performance dans les matrices précédemment crées par la méthode « initialiserVariablesResultats() ». Sont lancées les méthodes suivantes :
    a. « lancerKnn(posDS, vecDonnees, idObjCv, vecQtdKVoisins, valeurDistance) » ;
    b. « lancerRF(posDS, vecDonnees, idObjCv, vecQtdArbres, vecQtdArbresMin, vecProfMax) » ;
    c. « lancerSVM(posDS, vecDonnees, idObjCv, vecValPenalty, vecValGamma) » ;
    d. « lancerNN(posDS, vecDonnees,idObjCv, vecQtdNeurones, vecNbEpoques, vecTauxApprentissage, vecElan)» ;
    e. Additionner une unité à la variable « posDS » ;

Après avoir lancé toutes les expériences, le script enregistre les matrices qui contiennent les résultats de performances dans différents fichiers CSV, en appelant la méthode « ecrireResultats(chemainDossierBase) ». Ces fichiers sont utilisés pour générer les graphiques et analyser les résultats obtenus. Les fichiers enregistrés contiennent les données de performance enregistrées dans les matrices précédemment crées, voir section 5.3.1 (page 82).

5.3.2 Script d'apprentissage automatique – appel des algorithmes

Pour chacun des sets de données, le script lance une méthode qui va parcourir les vecteurs, initialisés au début du script, contenant toutes les configurations avec lesquelles vont être créés et validés les différents modèles. Le procédé est le même pour tous les algorithmes, et il n'a été décrit ici que pour MVS.

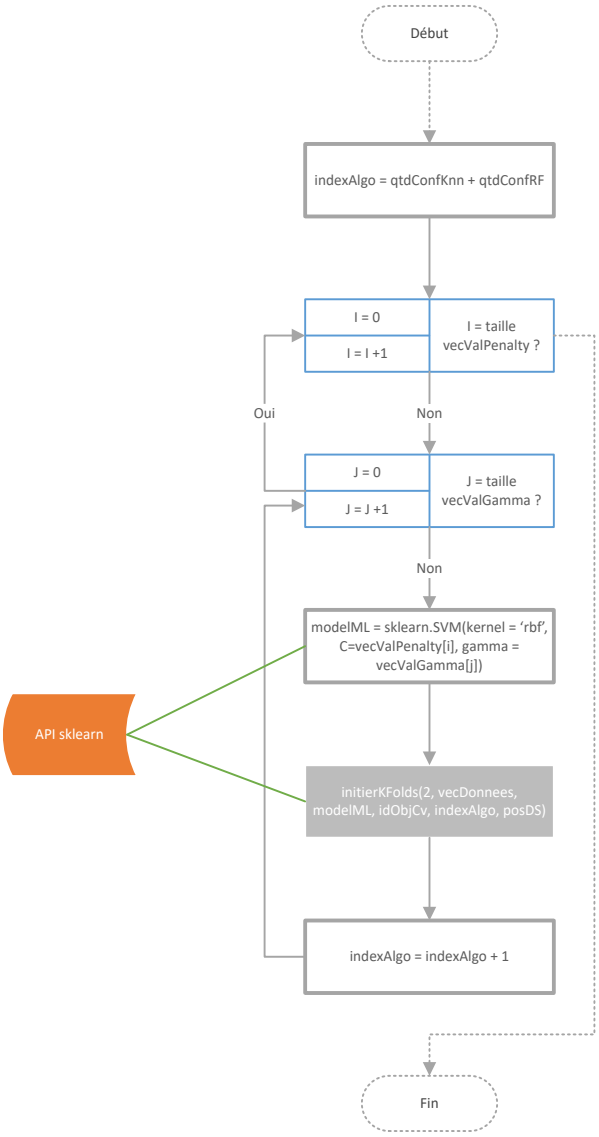


Figure 48 Diagramme de flux du script d'apprentissage automatique– exemple appel SVM

Pour chaque technique d'apprentissage automatique, l'algorithme va actualiser la variable « indexAlgo » en accord avec le nombre de configuration des précédentes techniques. Cette dernière indique où enregistrer les résultats obtenus dans les différentes matrices de performance. Pour chaque configuration l'algorithme va :

1. Créer un modèle d'apprentissage automatique avec les respectives configurations ;
2. Appeler la méthode « initierKFolds(2, vecDonnees, modelML, idObjCv, indexAlgo, posDS) » en passant en paramètres : le numéro de l'algorithme [25], les données précédemment traitées « vecDonnees », le model « modelML », l'objet CV « idObjCv »,

[25] 0 – Knn, 1 – RF, 2 – SVM, 3 – NN. L'API de NN étant différentes de toutes les autres, ce paramètre est utilisé pour que le script puisse savoir s'il s'agit de la méthodologie NN ou pas. (Voir section 5.3.3, page 86).

la position où garder les résultats de performance « indexAlgo » et le numéro du set de données « posDS ». Cette méthode est décrite dans la section 5.3.3 (page 87).

3. Additionner une unité à la variable « indexAlgo »

5.3.3 Script d'apprentissage automatique – entrainement, validation et test

Cette sous-section décrit l'implémentation de la partie du script qui s'occupe d'entraîner, valider et tester le modèle pour chacun des *folds*. Tout le procédé servant à sauvegarder les résultats de performance y est également décrit.

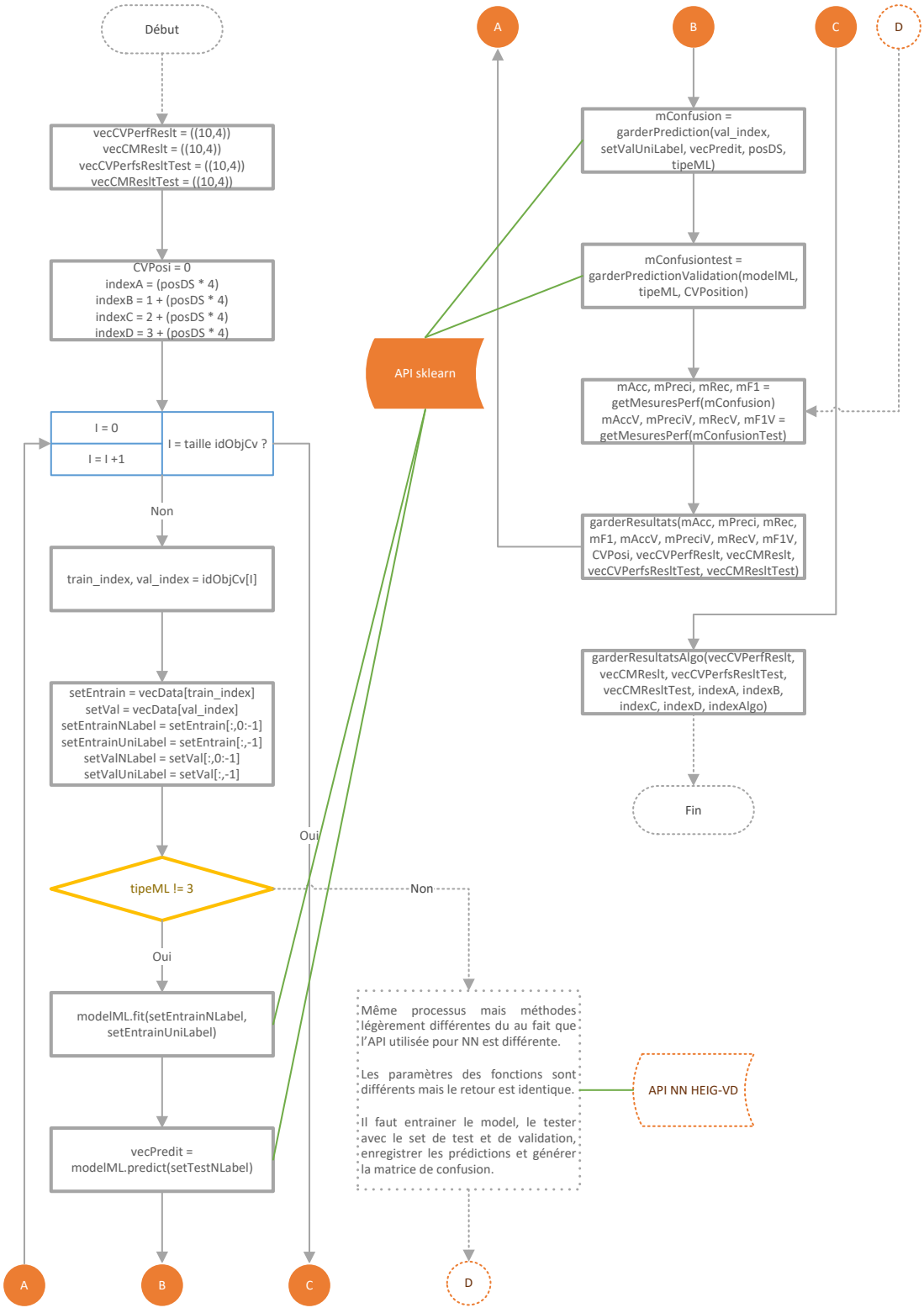


Figure 49 Diagramme de flux du script d'apprentissage automatique – exemple exécution

Chaque fois que la fonction « initierKFolds » est appelée celle-ci va créer deux matrices temporaires qui vont enregistrer l'*accuracy*, la précision, le *recall* et la mesure F1 de chaque *fold* pour le set de validation « vecCVPerfReslt » et de test « vecCVPerfsReslTest ». Sont également

créées deux autres matrices afin de garder les VP, VN, FN et FV de chaque *fold* pour les performances obtenues sur les sets de validation « vecCMReslt » et de test « vecCMResltTest ». La variable « CVPosi » contiendra le numéro du *fold* en cours de traitement. Les variables « indexA », « indexB », « indexC » et « indexD » contiennent les indices qui serviront à calculer les positions où vont être enregistrés les résultats finaux des *folds* dans les matrices de résultats. Pour chaque *fold* le script va :

1. Obtenir les indices du set de validation « val_index » et d'entraînement « train_index » ;
2. Séparer les variables des *labels* pour les deux sets de données ;
3. Vérifier s'il s'agit de la technique de réseaux de neurones ou d'une autre. Indépendamment du type de technique d'apprentissage automatique, les mêmes procédés sont appliqués. L'API *sklearn* utilisée actuellement ne dispose pas technique de réseaux de neurones, l'« API NN HEIG-VD » développée par les équipes de l'HEIG-VD a été utilisée pour la technique des réseaux de neurones;
4. Entraîner le modèle en lui passant comme paramètres le set d'entrainement, « setEntrainNLabel », sans les labels et un vecteur, « setEntrainUniLabel », contenant uniquement les *labels* ;
5. Évaluer les performances du modèle en lui passant comme paramètre le set de validation « setTestNLabel » sans les *labels*. Ce dernier retourne un vecteur, « vecPredit » où dans chaque position est inséré le résultat prédit ;
6. Appeler la fonction « garderPrediction(val_index, setValUniLabel, vecPredit, posDS, typeML) » qui reçoit les index du set de validation « val_index », le vecteur contenant les résultats espérés « setValUniLabel », le vecteur contenant les résultats prédits par le modèle pour le set de validation « vecPredit », le numéro du set de données « posDS » et le type de modèle entrainé « typeML ». Celle-ci va enregistrer les résultats prédits dans les matrices de performances précédemment créées, calculer et retourner la matrice de confusion ;
7. Appliquer le modèle au set de test qui contient les 5% de registres avec des interactions positives et 5% d'interaction négatives en appelant la méthode « garderPredictionValidation(modelML, typeML, CVPosition) ». Cette dernière reçoit comme paramètres le model qui a été entrainé « modelML », le type de modèle « typeML » et le numéro du *fold* « CVPosition », les deux derniers paramètres sont uniquement utilisés pour indiquer les positions où vont être enregistrées les prédictions obtenues. Cette méthode enregistre les prédictions obtenues dans les matrices de résultats créées au début du script et retourne la matrice de confusion ;
8. Appeler la méthode « getMesuresPerf(mConfusion) » en lui envoyant la matrice de confusion en paramètre afin de calculer et retourner les différentes mesures de performances tels que l'*accuracy*, la précision, le *recall* et la mesure F1. La méthode est utilisée pour calculer les résultats obtenus avec les sets de validation et le set de test ;
9. Enregistrer les mesures de performances obtenues avec les sets de données de validation et de test en appelant la méthode « garderResultats(mAcc, …, mRecV, mF1V, CVPosi, vecCVPerfReslt, vecCMReslt, vecCVPerfsResltTest, vecCMResltTest) » qui reçoit comme paramètres les résultats de performance retourner par la fonction précédente

« getMesuresPerf(mConfusion) », le numéro du *fold* « CVPosi », et les quatre matrices temporaires créées au début afin de garder les performances de chaque *fold* « vecCVPerfReslt, vecCMReslt, vecCVPerfsResltVal, vecCMResltVal ». Cette méthode enregistre les mesures de performance et valeurs prédites pour tous les sets de validation dans les matrices créées au début du script, les prédictions du set de test étant enregistrées à l'étape 7.

Après s'être occupé de tous les *folds*, le script calcule les moyennes et garde les résultats de performances dans les matrices globales en appelant la méthode « garderResultatsAlgo(vecCVPerfReslt, vecCMReslt, vecCVPerfsResltVal, vecCMResltVal, indexA, indexB, indexC, indexD, indexAlgo) » celle-ci reçoit comme paramètres les quatre matrices temporaires qui contiennent les résultats de tous les *folds* « vecCVPerfReslt, vecCMReslt, vecCVPerfsResltVal, vecCMResltVal », les positions où, comme cité ci-dessus, vont être enregistrés les résultats obtenus « indexA, indexB, indexC, indexD » ainsi que le numéro de la technique d'apprentissage automatique testée « indexAlgo ». Les cinq derniers paramètres servent à calculer l'emplacement où vont être enregistrés les résultats dans les matrices globales. Comme décrit dans la section 4.5 (page 55), la méthode calcule la moyenne des quatre mesures de performances obtenues dans les dix *folds* puis enregistre les résultats dans les matrices globales.

6 Évaluation et test

Ce chapitre présente les résultats obtenus avec les sets de données et les diverses configurations d'algorithme d'apprentissage automatique testés. Cette première étape a permis d'identifier 3 sets de données qui ont ensuite été combinés pour entraîner et valider le modèle final d'apprentissage automatique où chaque algorithme a été exécuté avec sa configuration optimale et les prédictions combinées par vote. Les résultats obtenus ont été comparés avec ceux des auteurs cités dans la section 4.1 (page 36), même si les objectifs finaux sont différents.

Dans la section 6.1 (page 91) sont présentées les configurations et séries de tests élaborés qui ont permis d'aboutir à une configuration optimale pour chacune des quatre techniques d'apprentissage automatique utilisées ainsi que de choisir les meilleurs sets de données utilisés pour l'entraînement des modèles. La section 6.2 (page 104) décrit le modèle final qui a été sélectionné avec apprentissage ensembliste basé sur un système de vote. Pour terminer, la section 6.3 (page 106) compare les résultats obtenus avec ceux décrits dans la section 4.1 (page 36).

6.1 Configuration et tests

Afin de sélectionner les meilleurs sets de données et configurations pour chacun des algorithmes utilisés, deux séries de tests ont été lancés. La première série, décrite dans la section 6.1.1 (page 92) , avait pour but de déterminer les sets de données qui permettent l'obtention des meilleurs résultats en fonction de plusieurs configurations. La deuxième série, décrite dans la section 6.1.2 (page 100), a permis d'affiner le choix de la configuration optimale à utiliser pour chaque algorithme d'apprentissage automatique.

6.1.1 Phase de test A

Avant de pouvoir créer les divers modèles, il a été nécessaire de générer les sets de données avec lesquels ceux-ci allaient être entraînés. Le numéro « Nº » a été attribué à chaque set de données et ce numéro est utilisé dans les *heatmaps*: les 10 premiers sont basés sur le nombre de *bins* (NB), allant de 5 à 50 pour les scores brutes (n°1-5) et les scores normalisés (n°6-10). Les 8 suivants se basent sur une taille de *bins* qui est fixe (TB), allant de 1 à 20 pour les scores bruts (n°11-15) et de 1E-6 à 5E-6 pour les scores normalisés (n°16-18). Finalement, le set n°19 correspond au set issu de l'extraction de la composition chimique des protéines.

Tableau 11 Sets de données générés

Nº	NB	Nº	NB – N	Nº	TB	Nº	TB - N
1	5	**6**	5	**11**	1	**16**	$1*10^{-6}$
2	10	**7**	10	**12**	5	**17**	$2,5*10^{-6}$
3	15	**8**	15	**13**	10	**18**	$5*10^{-6}$
4	30	**9**	30	**14**	15		
5	50	**10**	50	**15**	20	***19****	*%AA*

Il n'a pas été créé de set de données avec plus de 50 *bins,* puisque que le score IPP le plus élevé est de 50. Par ailleurs, les scores bruts étant des nombres entiers, choisir une taille de *bin* plus petite que 1 n'aurait pas été bénéfique.

Le Tableau 12 illustre les configurations utilisées lors de la première série d'expériences. Ce premier lancement a permis de déterminer les configurations et les sets de données avec les meilleures performances.

Tableau 12 Configurations 1 – apprentissage automatique

Algorithme	Paramètre	Configurations
k-plus proches voisins	Nombre de voisins	2 – 3 – 4 – 5 – 6 – 7 – 8 – 9
	Distance	Distance Euclidienne (*p=2*)
Forêts aléatoires	Nombre d'arbres	100 – 1'000 – 5'000
	Nombre de registre minimum par feuille	2 – 3 – 4
	Profondeur maximum	0 – pas utilisé
Machine a vecteur de support	Valeur de penalty	10'000 – 1' 000 – 100 – 10 – 1 – 0.1 – 0.01
	Valeur de l'élan	0.0001 – 0.001 – 0.01 – 0.1 – 10 – 100 – 1'000 – 10'000
Réseaux de neurones	Nombre de neurones	2 – 3 – 4 – 5 – 6
	Époques	10 – 25 – 50 – 75 – 100
	Taux d'apprentissage	0.01
	Élan	0.1 – 0.4 – 0.7

Le temps nécessaire au script pour entraîner et évaluer les performances des modèles basé sur les forêts aléatoires étant minime, un peu moins de 50 minutes, il n'a pas été nécessaire de limiter la profondeur maximum des arbres et l'on s'est donc concentré plutôt sur leur nombre ainsi que sur le nombre de registres par feuille. Dans l'algorithme des réseaux de neurones, le taux d'apprentissage n'a pas été testé avec d'autres valeurs que « 0.01 » dû au fait que celui-ci a apporté de bons résultats. Par ailleurs, il est à noter que MVS prend beaucoup de temps à entraîner les modèles, puisque sur les 50 heures de calculs nécessaires, un peu plus de 20h lui ont été consacrées.

Accuracy, F1, sensitivité et spécificité ont été les mesures utilisées pour générer les *heatMap* illustrant les résultats obtenus. Sur l'axe des ordonnées sont représentés tous les sets de données précédemment générés et, sur l'axe des abscisses, le numéro des configurations avec lesquelles ont été créés les divers modèles. La barre à droite de tous les *heatmap* représente l'échelle de couleur des scores de performance, ceux-ci étant compris entre [0.75-0.99] et devant être multipliés par 100 pour obtenir un pourcentage.

La Figure 50 illustre les résultats obtenus avec l'algorithme des K-plus proches voisins. Sur l'axe des abscisses sont indiqués le nombre de voisins avec lesquels les modèles ont été entraînés. Ces résultats indiquent que k=2 voisins est la meilleure configuration avec ~91% d'*accuracy* et ~92% de F1 pour les meilleurs sets de données. En général, les modèles entraînés avec k-plus proches voisins ont une meilleure sensibilité face à la spécificité. Cela signifie qu'ils arrivent à mieux prédire les interactions positives. Les *heatmap* concernant *l'Accuracy* et le score F1 montrent clairement que les sets de données ayant un plus grand nombre de *bins* permettent l'obtention de meilleurs résultats et parmi ceux-ci les sets de données NB10 et TB 1 ont permis l'obtention des meilleurs scores. Le set de données basé sur la composition chimique des séquences protéiques obtient de moins bons résultats avec cet algorithme que les autres sets de données.

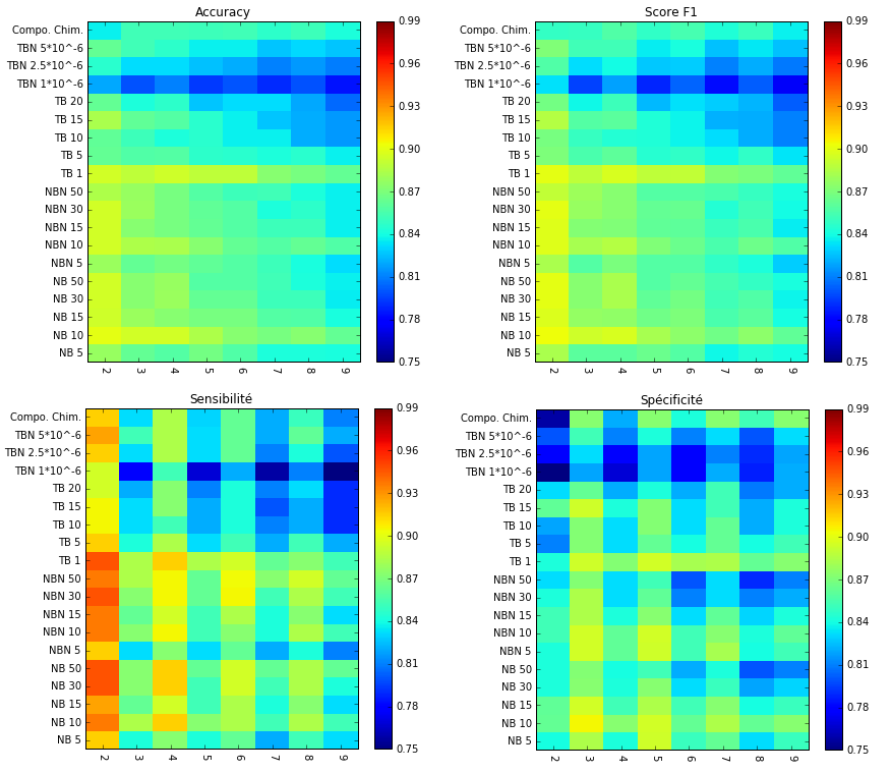


Figure 50 *Heatmap* – Résultats apprentissage automatique pour k-plus proche voisins

La Figure 51 illustre les résultats obtenus avec l'algorithme de forêts aléatoires. Dans le Tableau 13 sont décrits les numéros des configurations représentées sur l'axe des abscisses dans les graphiques des résultats obtenus avec RF. Contrairement à l'algorithme précédent, pour celui-ci il n'y a aucune configuration qui soit meilleure que l'autre, la différence des résultats obtenus dépend essentiellement du type de set de données utilisé. Indépendamment de la mesure de performance utilisée, il est possible de voir une différence de résultats obtenus avec les sets de données où l'on a indiqué le nombre de *bins* (NB et NBN) et ceux où la taille de *bins* fut indiquée (TB et TBN) à l'exception du set de données contenant des *bins* de taille 1 (TB1). Les scores de performance de ce dernier set de données et celui contenant 50 *bins* sont identiques étant donné que le score IPP maximum est de 50. Parmi les sets de données où le nombre de *bins* a été fixé, indépendamment d'avoir les données normalisées, on remarque que plus il y a de *bins,* plus les performances sont élevées (NB et NBN). Tout comme avec l'algorithme K-plus proches voisins, le set de données basé sur la composition chimique des séquences protéines ne donne pas de bons résultats. Tous les modèles entraînés ont montré de meilleures performances de sensibilité face à la spécificité indiquant qu'ils arrivent à mieux prédire les interactions positives que les négatives.

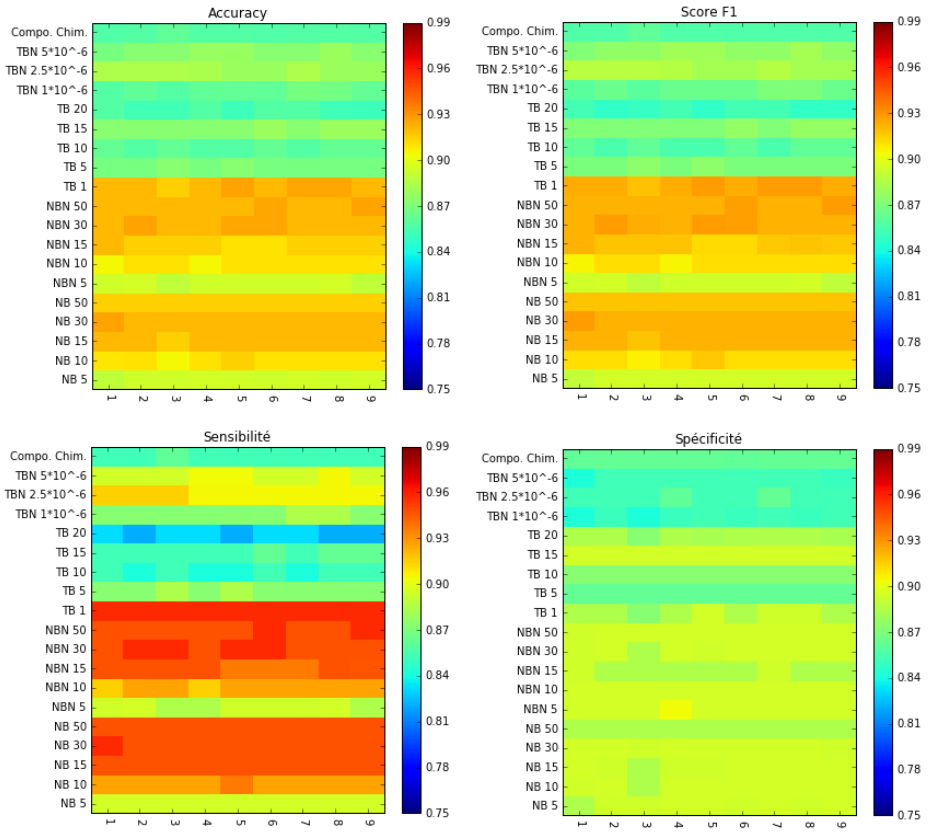


Figure 51 *Heatmap* – Résultats apprentissage automatique pour forêts aléatoires

Tableau 13 Représentation des configurations RF

Params.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
N. arb	100	100	100	1’000	1’000	1’000	10’000	10’000	10’000
Min. arb	2	3	4	2	3	4	2	3	4

Dans la Figure 52, il est possible de voir les résultats obtenus avec l’algorithme MVS. Chaque ligne orange indique que la valeur de la variable penalty a changé, en accord avec les valeurs présentées dans le Tableau 12. Cette figure montre que plus la valeur de la variable *penalty* est petite, plus les scores de performance ont tendance à être mauvais. Tout comme avec RF, les meilleurs résultats ont été obtenus avec les sets de données où le nombre de *bins* est fixé (NB et NBN). La même exception apparaît avec cet algorithme pour le set de données ayant 50 *bins* de taille 1 (TB1), pour la même raison que dans le cas RF. Il est observé que les meilleurs scores de performance sont obtenus avec une valeur d’élan autour de 1. Comme pour RF, plus les sets de données possèdent de *bins*, plus les résultats sont bons (NB et NBN). Le set de données basé sur la composition chimique des séquences protéiques n’obtient toujours pas de meilleurs résultats que les sets de données basés sur les domaines des protéines. Tout comme les deux derniers algorithmes, celui-ci a de meilleurs scores de sensibilité que de spécificité indiquant

une meilleure prédiction pour les interactions positives. Il est nécessaire de prendre en compte qu'une sensitivité ou spécificité à 100% signifie que le modèle a sur-appris et s'est adapté aux données, mais ne pourra pas prédire correctement de nouvelles données. C'est pourquoi ces configurations ne seront pas considérées dans les étapes suivantes.

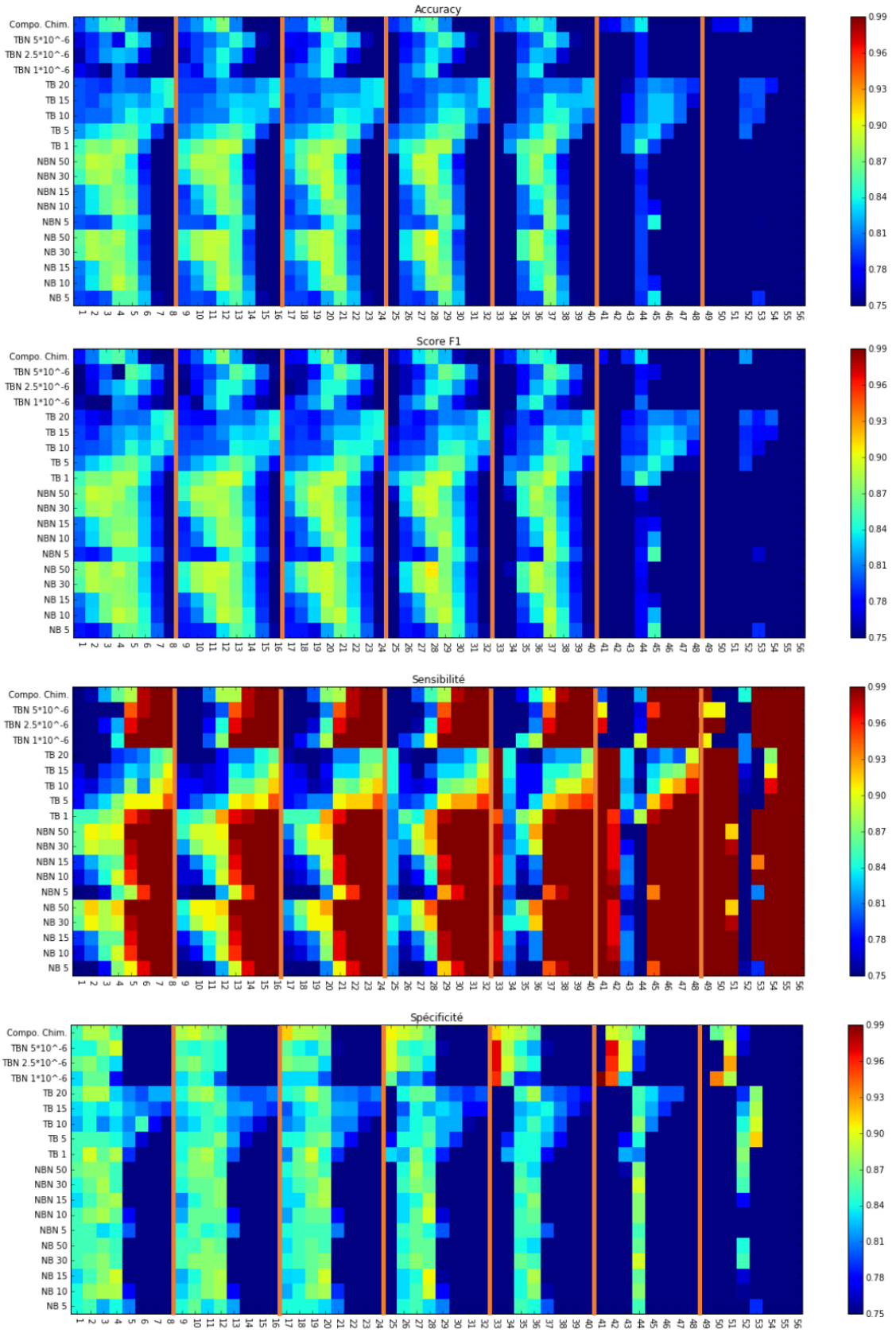


Figure 52 *Heatmap* – Résultats apprentissage automatique pour MVS

Dans la Figure 53 sont illustrés les résultats obtenus à travers les modèles entraînés avec l'algorithme NN. Les barres noires indiquent un changement du nombre de neurones, en accord avec les paramètre définis dans le Tableau 12. Pour cet algorithme, il est possible de voir que plus le nombre de neurones augmente, plus les performances sont bonnes. Tout comme pour RF et MSV, les meilleurs résultats ont été obtenus avec les sets de données basé sur les nombre de *bins* (NB et NBN) et celui contenant des *bins* de taille 1. Parmi ces sets de données, plus il y a de *bins,* plus les résultats sont bons. Contrairement aux autres algorithmes et de manière générale, NN à une spécificité plus haute ce qui indique que celui-ci arrive à mieux prédire les interactions négatives que positives.

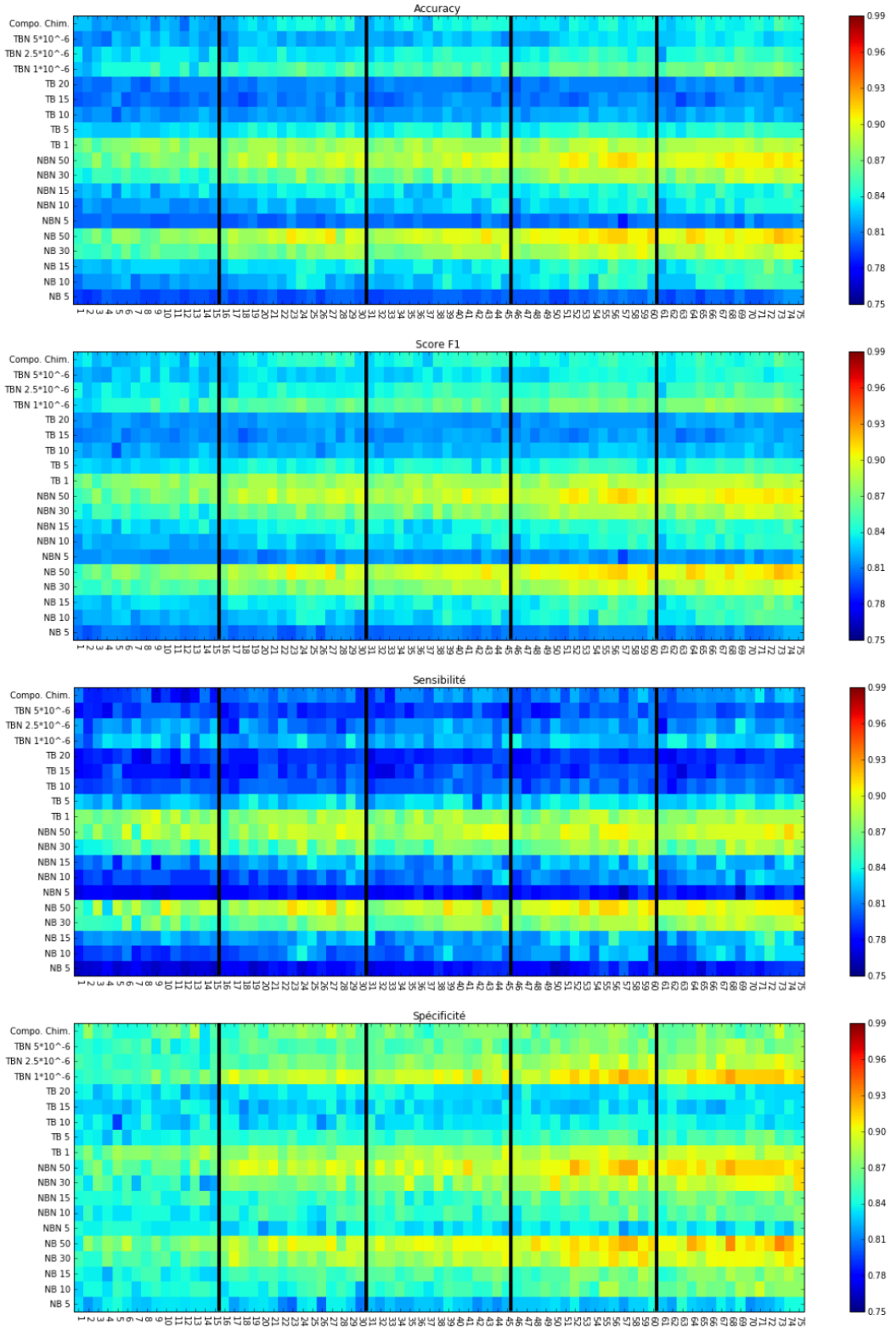


Figure 53 *Heatmap* – Résultats apprentissage automatique pour NN

6.1.2 Phase de test B

Le Tableau 14 illustre les configurations testées lors de la deuxième série de *runs*. Pour cette deuxième série, nous avons également utilisé un set de données ne contenant qu'un seul *bin*, correspondant au nombre de IPPs par paire bactérie-bactériophage. Tout comme avec la première série, les mesures de performances utilisées sont l'*Accuracy*, F1, sensitivité et spécificité. Après avoir analysé les résultats obtenus sur la première phase de test, les sets de données contenant 50 bins de taille 1 (TB 1), 50 bins de taille sélectionnée par l'API (NB 50), 30 bins de taille sélectionnée par l'algorithme (NB 30), 1 bins de taille sélectionnée par l'algorithme (NB 1) et le set de données basé sur la composition chimique des séquences protéiques ont été choisis pour cette seconde phase. Indépendamment des résultats obtenus avec le set de données basé sur la composition chimique des séquences protéiques, celui-ci a quand même été gardé afin de maintenir une diversification des variables. Le set de données contenant un seul bin a quant à lui été généré et utilisé dû au fait que certains étudiants de la HEIG-VD, auxquels avait été fourni le script de génération de scripts, ont obtenus de bons résultats de performance sur de simples algorithmes d'apprentissage automatique sans agrégation de modèles.

Tableau 14 Configurations 2 – apprentissage automatique

Algorithme	**Paramètre**	**Configurations**
k-plus proches voisins	Nombre de voisins	1 – 2 – 3
	Distance	2
Forêts aléatoires	Nombre d'arbres	100 – 1'000 – 5'000
	Nombre de registres minimum par feuille	2 – 3 – 4
	Profondeur maximum	0
Machine à vecteur de support	Valeur de penalty	100 – 10
	Valeur de l'élan	1
Réseaux de neurones	Nombre de neurones	7 – 8 – 9 – 10
	Époques	10 – 25 – 50 – 75 – 100
	Taux d'apprentissage	0.01
	Élan	0.1 – 0.4 – 0.7

La Figure 54 illustre les résultats obtenus avec l'algorithme K-plus proches voisins. Sur la partie de gauche sont illustrées les performances obtenues avec le set de données ne contenant qu'un seul *bin* (NB1) avec, sur l'axe des abscisses le nombre de voisins et en ordonnées, les mesures de performances. La partie droite de la figure illustre les scores de performance obtenus avec un seul voisin pour les sets de données qui ont été maintenus pour cette deuxième série (en ordonnées). Il est possible de voir que le nouveau set de données (NB 1) n'arrive pas à surpasser les résultats obtenus avec les autres et qu'un ou deux voisins donne les meilleurs résultats (l'implémentation de kNN donne les mêmes résultats avec 1 ou 2 voisins).

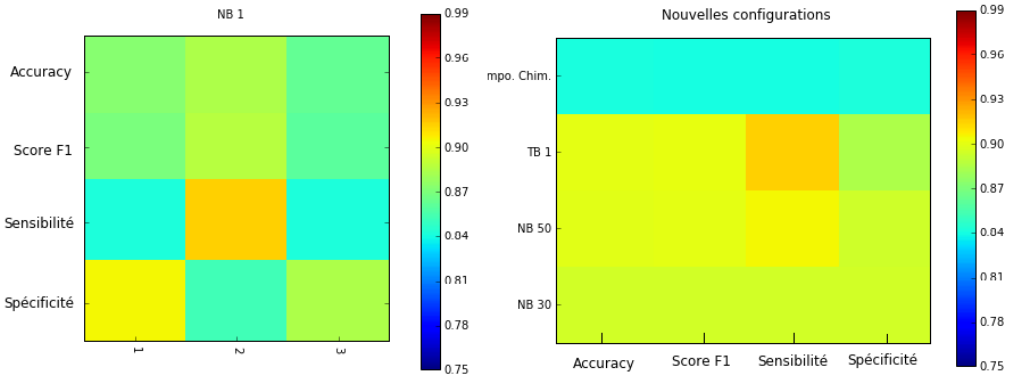


Figure 54 *Heatmap* – Résultats configurations 2 pour K-plus proches voisins

Dans la Figure 55 sont représentés les résultats obtenus pour le set de données composé d'un seul bin avec l'algorithme forêts aléatoires. L'axe des ordonnées représente les mesures de performance et celui des abscisses les configurations. Face aux résultats de performance illustrés sur la Figure 51, la sensibilité est moins élevée ce qui signifie que le modèle n'arrive pas à bien prédire les interactions positives. L'*accuracy* et F1 arrivent à des scores d'approximativement 88% alors que les meilleurs résultats pour ces deux mesures atteignent plus de 96% lors de la première phase de test.

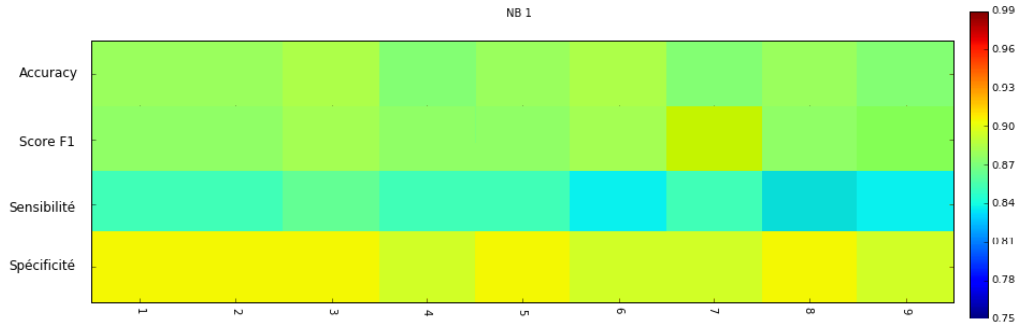


Figure 55 *Heatmap* – Résultats configurations 2 pour forêts aléatoires

La Figure 56 illustre les résultats de performances obtenus avec le set de données à un *bin* pour les configurations représentées dans le Tableau 14 pour l'algorithme MVS. Les meilleurs résultats sont obtenus lorsque la valeur de l'élan est haute. L'*accuracy* et F1 atteignent, approximativement, 87% pour les meilleures configurations alors que celles-ci atteignent plus de 90% avec les configurations et set de données gardés comme illustré sur la Figure 52.

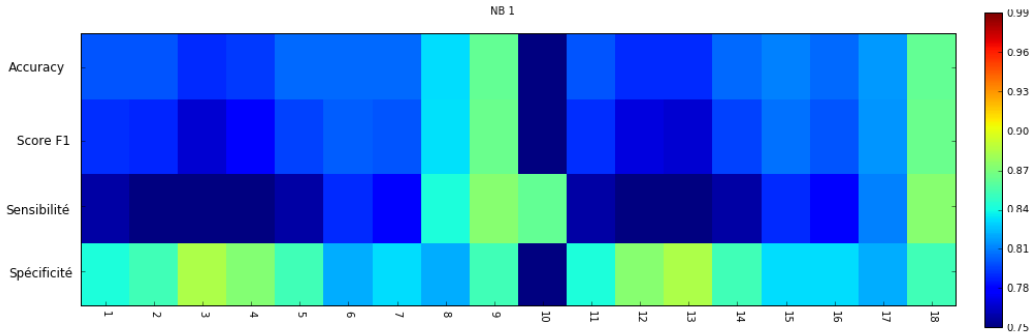


Figure 56 *Heatmap* – MVS avec 1 bins

Dans la Figure 57 sont représentés les résultats obtenus avec les configurations illustrées dans le Tableau 14 pour les sets de données retenus avec l'algorithme MVS. Face aux résultats obtenus avec les premières configurations, il est possible d'affirmer que ces derniers sont équivalents. Sensibilité toujours meilleure face à la spécificité mais qui, cependant, possède un pourcentage de 100% pour le set de données à 50 *bins* indiquant un sur-apprentissage et éliminant ainsi ces deux configurations.

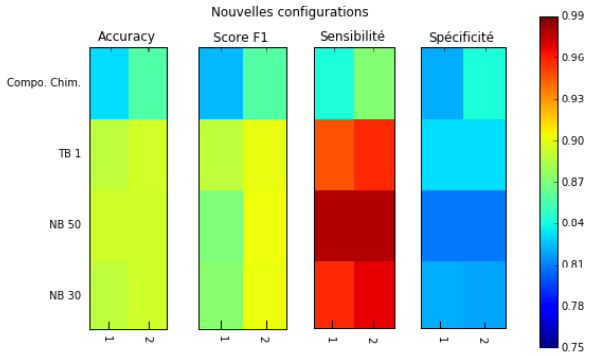


Figure 57 *Heatmap* – Configurations MVS avec set de données retenus

Dans la Figure 58 sont illustrés les résultats obtenus avec le set de données qui contient 1 *bin* et ceux retenus pour cette deuxième phase de test avec les configurations illustrées dans le Tableau 14 pour l'algorithme des réseaux de neurones. Sur la Figure 58, on constate que plus le nombre d'époques augmente, plus les performances sont élevées. Les nouveaux tests ont permis de confirmer ceci et montrent que les scores stagnent avec 9 et 10 neurones dans la couche intermédiaire. La spécificité continue d'être meilleure que la sensibilité permettant, tout comme avec les premières configurations, d'affirmer que les modèles entraînés ici avec NN ont plus de facilité à prédire les interactions négatives que positives.

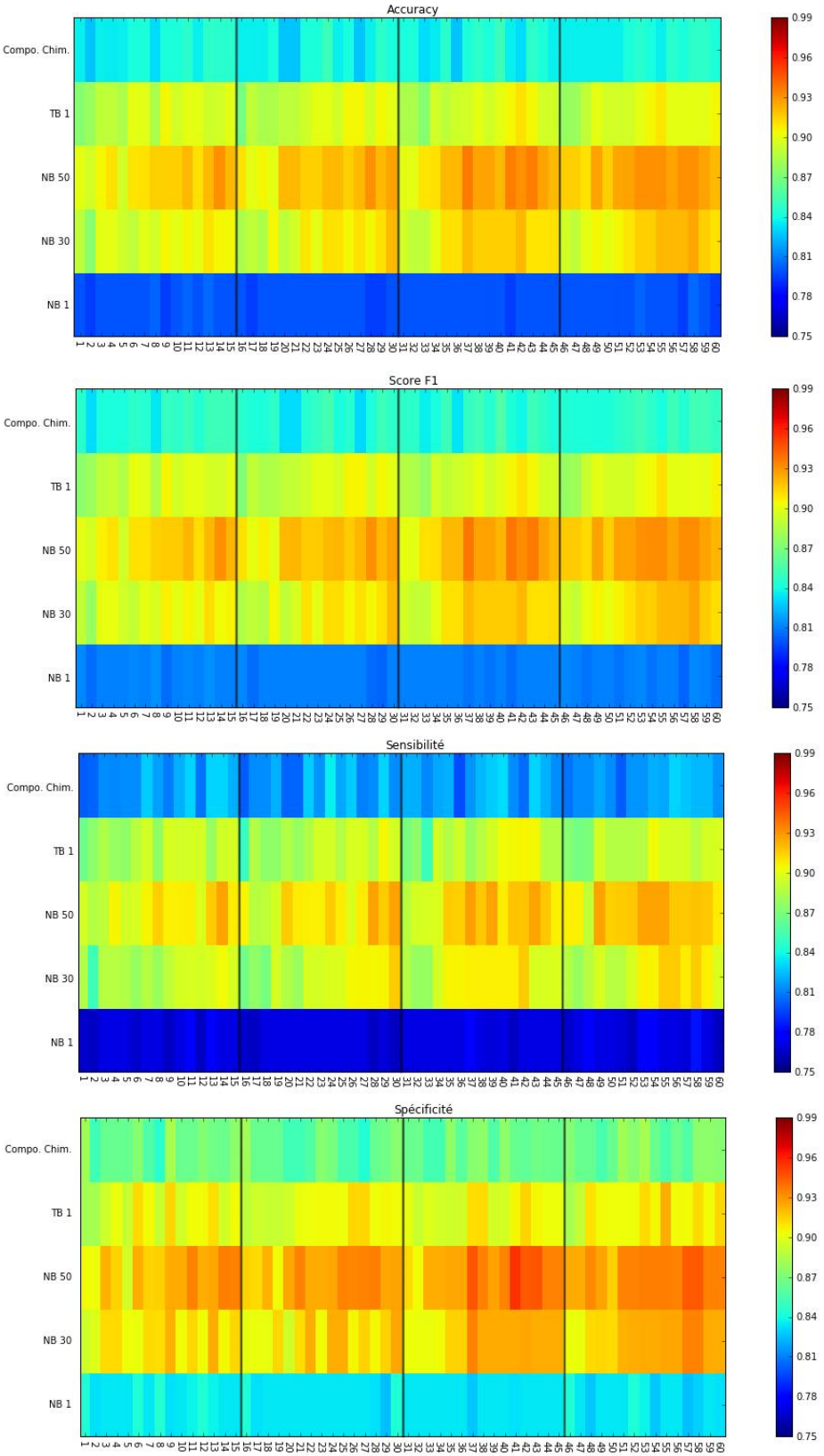


Figure 58 *Heatmap* – NN nouvelles configurations avec NB1 et set de données gardés

6.2 Résultats du modèle final

Pour l'élaboration du modèle final, les configurations et sets de données qui ont obtenus les meilleurs résultats de performance ont été sélectionnés afin de pouvoir élaboré l'algorithme d'apprentissage automatique basé sur l'approche ensembliste avec un système de vote. En ayant pour base l'analyse de résultats des deux chapitres précédents, les sets de données maintenus sont :

- Set de données contenant 50 *bins* de taille définie par l'API ;
- Set de données contenant 50 *bins* de taille 1 ;
- Set de données basé sur les séquences protéiques. Celui-ci n'ayant obtenus d'excellents résultats a quand même été sélectionné, dû au fait qu'il soit basé sur les séquences protéiques.

Avec les sets de données retenus, un nouveau set a été créé en combinant ceux-ci. Le set de données utilisé pour l'entraînement, validation et test, illustré dans la Figure 59, est donc composé de 208 caractéristiques. Ceci est possible dû au fait que les id des registres retirés pour le set de test ont été les mêmes sur tous les sets de données générés.

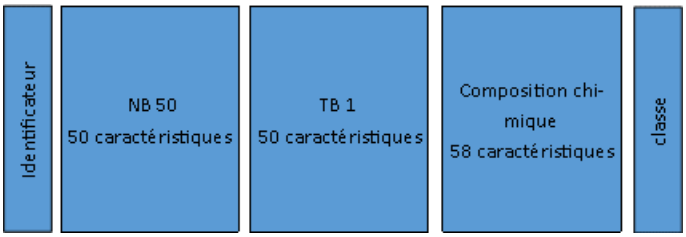


Figure 59 Set de données de test

Les configurations retenues pour chacun des algorithmes d'apprentissage automatique sont pour :

- Knn : 2ème configuration de la première série de tests - 2 voisins ;
- RF : 2ème configuration de la première série de tests - 100 arbres et 3 arbres minimum dans chaque feuille ;
- SVM : 28ème configuration de la première série de tests – 10 de valeur de penalty et 0.1 de Gamma ;
- NN : 37ème configuration de la deuxième série de tests – 9 neurones, 50 époques, taux d'apprentissage de 0 .01 et 0.1 d'élan.

L'approche ensembliste élaborée somme le nombre de fois qu'une interaction a été classée comme positive et négative parmi les quatre modèles entraînés. La classe attribuée étant celle ayant reçu le plus haut score, en cas d'égalité la classe positive est assumée. Le choix de maintenir la classe positive et dû au fait que toutes les prédictions positives vont être testées en laboratoire contrairement aux négatives qui ne le seront pas.

Pour les résultats finaux, il s'est avéré plus intéressant de voir les performances obtenues sur chacune des bactéries. Pour chaque bactérie ont été sélectionnés les résultats prédits dans

chacune des paires où celles-ci apparaissent afin de pouvoir élaborer sa matrice de confusion et calculer la sensibilité et spécificité à partir de celle-ci.

Les résultats des paires bactérie-bactériophage utilisées pour la validation et le test ont été calculés séparément. Sur la Figure 60 sont illustrés les scores de performance moyens obtenus avec les sets de validations et sur la Figure 61 les résultats pour le set de test. Les points représentés sur les figures 59 et 60 sont composés d'un numéro représentant l'identifiant de la bactérie et leur taille est proportionnelle au nombre d'occurrences de celle-ci dans la base de données.

Sur la Figure 60, il est possible de voir que la bactérie n° 1 a une sensitivité de plus de 90% et une spécificité de 87%. Cette même bactérie est présente dans une grande fraction des interactions positives où sur les 1065 elle apparaît 915 fois. Contrairement à la bactérie N° 1, les autres sont plus présentes dans les interactions négatives ce qui fait augmenter la spécificité sachant que cette mesure évalue le nombre d'interactions négatives correctement prédites. Cependant, certaines bactéries bien représentées dans la base des interactions négatives n'ont pourtant pas obtenu de bons résultats de spécificité (ID N° 3, 26, 12, 0, 27 et 25).

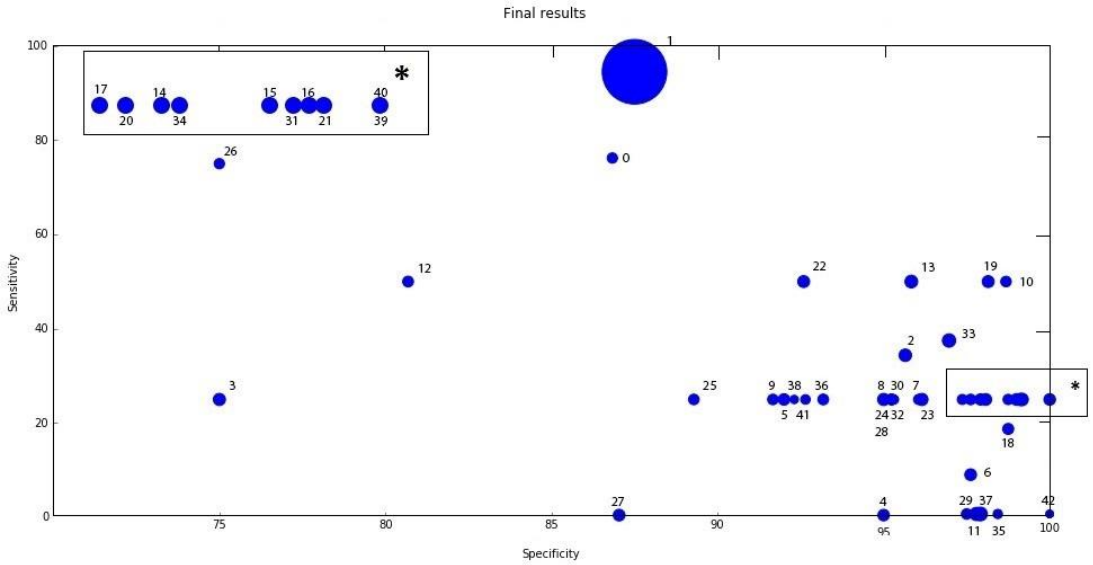


Figure 60 Résultats par bactérie – Entrainement

Dans la Figure 61, représentant les résultats sur le set de test, il est possible de constater que plusieurs bactéries ont obtenu une spécificité de 100% et une sensitivité de 0%. Ceci est dû au fait que, dans le set de test, celles-ci sont utilisées uniquement dans des interactions négatives et sont présentes dans peu d'interactions négatives. Le contraire se passe avec la bactérie N° 1 où, dans le set de test, elle n'est utilisée que dans des interactions positives. Les bactéries N° 7, 8, 17, 24 et 30 possèdent une sensibilité de 100% et cette valeur s'explique par le fait que le set de test comporte moins de 4 interactions négatives et que toutes ont été correctement prédites.

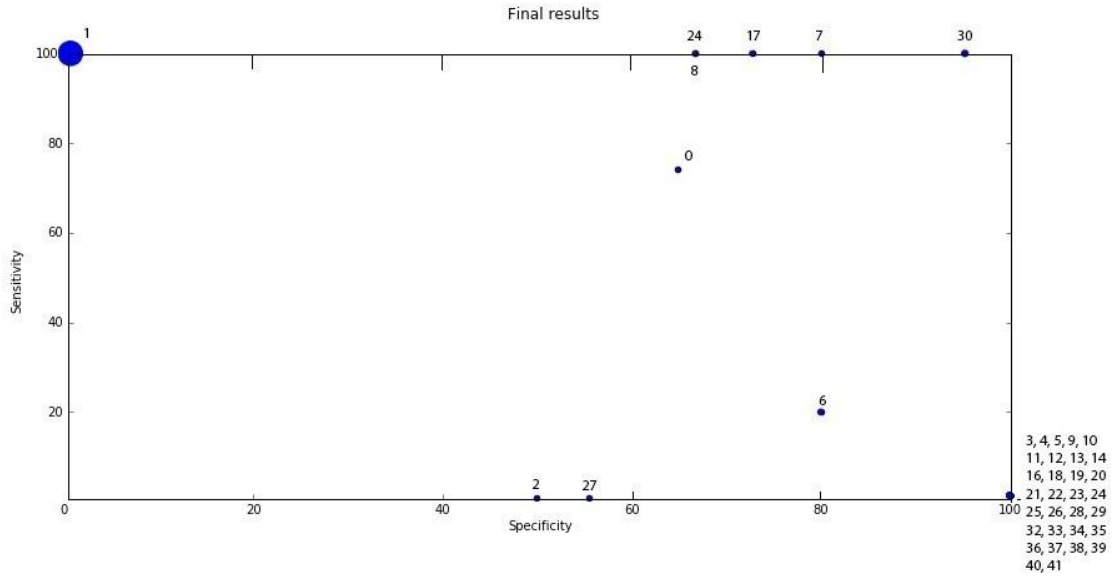


Figure 61 Résultats par bactéries – Test

En analysant les résultats obtenus avec le set d'entraînement et de test, il est possible de dire que le modèle élaboré est performant. Les résultats obtenus avec le set de test pour chaque bactérie, en tenant compte de celles qui n'apparaissent que dans les paires positives ou négatives, viennent supporter l'hypothèse qu'il est possible d'utiliser les techniques d'apprentissage automatique pour résoudre le problème des paires d'interaction bactéries-bactériophage afin de contribuer à la résolution du problème de la résistance aux antibiotiques, voir chapitre 7 (page 111) pour les étapes futures du projet.

6.3 Comparaison des résultats

Dans la section précédente, les résultats de performances pour chacune des bactéries ont été analysés. Après avoir effectué une recherche, et comme décrit dans la section 4.1 (page 36), il n'a été trouvé aucune approche identique à ce qui a été développé indiquant que ce projet, pionnier en la matière, peut réellement devenir intéressant en termes médicaux, de recherche mais également commerciaux. À titre indicatif, les résultats obtenus sur l'ensemble du modèle avec ceux des autres auteurs ont été comparés, décrit sous la section 4.1 (page 36). Ces derniers ont été calculés sur la moyenne des matrices de confusion obtenues sur chaque *fold* avec l'agrégation de modèles. Le Tableau 15 contient les résultats de performance obtenus avec les sets de validations et de test.

Tableau 15 Résultats finaux

	Accuray	F1	Précision	Sensibilité	Spécificité	Echantill-onnage	Apprenti-ssage	Set de données
Validation	91.1%	90.9%	94.3%	87.9%	94.7%	Validation croisée (10)	Agrégation de modèles (Knn, RF, MVS, NN)	EN 1’917 TE 213
Test	87.7%	87.3%	85.3%	89.5%	86.0%			

Sachant que les modèles élaborés par les auteurs décrits dans la section 4.1 (page 36) n'ont pas les mêmes objectifs finaux que ceux du modèle développé tout au long du projet, il a quand même été décidé d'effectuer une comparaison de résultats. Face aux résultats obtenus par (Cui et al. 2012) lors de ses deux approches, tous ceux obtenus avec le modèle élaboré durant le projet sont supérieurs. Le modèle élaboré au cours du projet, surpasse les scores des auteurs de presque plus de 10% pour la sensibilité mais est presque identique pour ce qui est de la spécificité. Les résultats obtenus par (Dyer et al. 2007) indiquent une haute précision et basse sensibilité signifiant que celui-ci arrive à bien prédire la classe des vrais positifs mais se trompe lorsqu'il doit prédire les vrais positifs et les classifient comme faux négatifs dans plus de 60% de cas. La précision du modèle élaboré surpasse de plus de 15% et un peu moins du double pour la sensibilité. Pour terminer, les résultats que (Coelho et al. 2014) a obtenus sont globalement les mêmes pour l'*accuracy*, F1 et précision, ce qui n'est pas le cas de la sensibilité qui n'atteint que 28% dans son approche et plus de 89% avec le modèle développé.

Comme cité plus haut, dans la section 4.1 (page 36), ces comparaisons ne sont faites qu'à titre indicatif car les objectifs finaux ne sont pas les mêmes. Les auteurs cités visent à créer des modèles capables de prédire des interactions entre des protéines alors que le modèle développé cherche à prédire des interactions entre une bactérie et un bactériophage. Cependant, les résultats obtenus avec le modèle développé étant supérieurs aux approches comparées, ceci peut être considéré comme une forme de soutenir l'hypothèse du projet, décrite dans le chapitre 1 (page 1), qui consiste en l'utilisation de techniques d'apprentissage automatique pour élaborer un modèle capable de prédire les interactions entre les bactéries et bactériophages.

6.4 Tests d’hypothèses statistiques

Tous résultats obtenus dans le cadre d’un projet d’investigation nécessitent d’être validés et comparés aux résultats obtenus par d’autres auteurs ou simplement en comparant les résultats obtenus avec diverses configurations d’un système ou, dans le cadre du projet, diverses configurations d’algorithmes d’apprentissages automatiques (Riou & Landais 1998). Les tests d’hypothèses statistiques sont une façon de mesurer la performance entre deux ou plusieurs

algorithmes et garantir que si l'un des algorithmes est meilleur qu'un autre, ce n'est pas dû au hasard (Demšar 2006).

Afin de comparer les résultats de divers algorithmes, des tests paramétriques peuvent être utilisés lorsque les conditions nécessaires à leur utilisation sont effectivement vérifiées. Dans le cas contraire, des tests non-paramétriques peuvent être utilisés. Les tests paramétriques ont l'avantage d'être statistiquement plus robustes dans leurs résultats (Triola & Triola 2012). Le choix du test dépend également du nombre d'algorithmes que l'on veut comparer/évaluer, deux ou plus. L'*accuracy*, la mesure F1 et l'aire sous la courbe ROC sont les principales mesures utilisées pour mesurer la performance des algorithmes. Ce sont ces mêmes mesures qui pourront être soumises aux tests statistiques d'hypothèses. Dans cette section, l'ensemble des résultats obtenus et utilisés pour les tests sont appelés des échantillons qui, pour un test, doivent tous être basé sur le même type de résultat de performance.

Pour la comparaison de deux algorithmes, le teste paramétrique test-t ou non-paramétrique Wilcoxon peut être utilisé, suivant que les scores de performance suivent une distribution normale[26] ou non (Demšar 2006). Ces derniers ont pour base deux hypothèses : (H0) les deux algorithmes sont équivalents ou (H1) les algorithmes sont différents. H0 est acceptée ou réfutée en accord avec un pourcentage d'erreur, généralement de 5%. Si H0 est acceptée alors, il est possible d'affirmer que les algorithmes n'ont pas de différences significatives pour le seuil déterminé (généralement 5%) (Demšar 2006).

Il est également intéressant de comparer plusieurs algorithmes. Dans ce cas, c'est le test paramétrique ANOVA qui peut être utilisé, à condition que les conditions suivantes soient validées : les échantillons suivent la distribution normale, les échantillons de chaque algorithme sont indépendants[27] et leurs variances sont similaires[28]. Dans le cas contraire, le test non paramétrique de Friedman peut être utilisé (Demšar 2006). Tout comme le test-t et le test de Wilcoxon, ces tests sont basés sur deux hypothèses : (H0) les algorithmes sont équivalents ou (H1) Il existe au moins deux algorithmes différents (Rakotomalala 2013). Si H0 est réfutée, il est possible d'utiliser un test post-hoc qui, comparant les échantillons deux à deux, va indiquer lesquels sont différents. Le test post-hoc du Tukey est utilisé si ANOVA a été utilisé, et celui de Nemenyi si c'est le test de Friedman qui a été utilisé (Demšar 2006). La Figure 62 schématise les tests qui peuvent être utilisés selon en accord avec les échantillons disponibles.

[26] Peut être vérifié à l'aide du test de Shapio-Wallis – H0 : suit une distribution normale
[27] Peut être vérifié à l'aide du test de Chi-square – H0 : variables indépendantes
[28] Peut être vérifié à l'aide du test de Levene – H0 : variances équivalentes

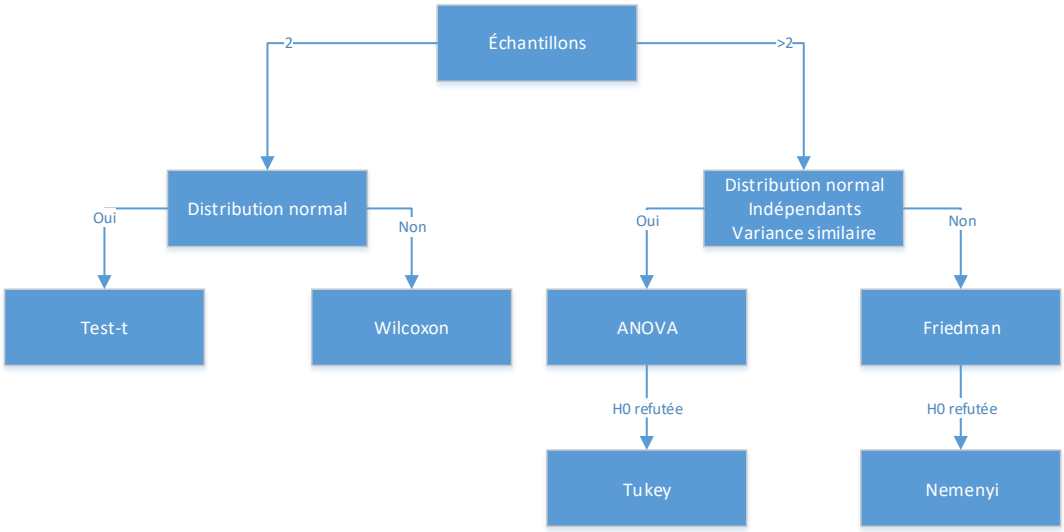


Figure 62 Schéma tests statistiques – Adapté de (Japkowicz 2011)

Dans le cadre du projet, il n'a été créé qu'un seul modèle ne permettant pas d'effectuer de comparaison. Sachant que, comme décrit dans le chapitre 7 (page 111), le projet fut financé, l'étude faite concernant les tests statistiques d'hypothèses sera utile dans le cadre d'une thèse de doctorat. Au stade actuel du développement du projet, il n'est pas encore possible d'affirmer avec certitude quels seront les algorithmes qui vont être élaborés ou même quelles nouvelles variables vont être utilisées, voir chapitre 7 (page 111).

7 Conclusion

La résistance aux antibiotiques est un réel problème qui nécessite une solution le plus rapidement possible. Actuellement, plusieurs études sur la phagothérapie sont en cours dans diverses universités et hôpitaux dont le Centre Hospitalier Universitaire Vaudois en suisse (Coulon 2015) et l'Université d'Aveiro au Portugal (Oje 2016). La phagothérapie est vue comme une alternative potentielle aux antibiotiques, dont le concept consiste à trouver un virus bactériophage qui attaque la bactérie causant la maladie chez le patient. Le bactériophage étant très spécifique, celui-ci ne possède que peu ou pas d'effets secondaires (Coulon 2015). Le principal inconvénient de cette thérapie réside dans la difficulté de trouver les bactériophages qui arrivent à attaquer les bactéries ciblées, ce qui, actuellement, est fait de façon empirique. Les abordages existants, étudiés sous la section 4.1 (page 36), ne permettent que l'identification des interactions au niveau des protéines et non pas entre bactéries et bactériophages.

Le domaine de l'apprentissage automatique est de plus en plus utilisé dans les plus divers domaines pour prédire ou classifier divers types d'informations. L'utilisation de ces techniques est très présente dans le *marketing* afin de détecter le type de clientèle des supermarchés, déterminer la quantité que chacun d'eux est prêt à dépenser ou tout simplement essayer de savoir ce que chacun des clients pourrait acheter et ce depuis un moment déjà (Shaw et al. 2001). Dans le domaine de la santé, ces techniques sont utilisées dans diverses formes comme l'analyse d'image, l'aide au diagnostic de maladies ou dans le *fitness* (Venturerader 2016; Oliveira et al. 2016).

La thèse réalisée a permis de répondre à la question « Est-ce que l'apprentissage automatique pourrait être utilisé pour détecter les paires d'interactions bactéries-bactériophages ? ». Avec des performances de plus de 90% d'*accuracy*, les résultats obtenus et présentés dans le chapitre 6 (page 91) ont permis d'affirmer que cet manière d'aborder le problème semble être une bonne voie à suivre. Le chapitre 2 (page 9), montrant l'état de l'art a permis de comprendre tout l'aspect biologique et informatique essentiel au bon déroulement du projet, notamment pour l'extraction des variables utilisées.

Dans le chapitre 2 (page 9), l'aspect biologique a permis de prendre conscience de la complexité et l'importance du projet. Acquérir les compétences biologiques a été une étape importante sans laquelle le projet n'aurait pu aboutir. L'acquisition de vocabulaire scientifique pour que toute l'équipe du projet puisse se comprendre a été et est, encore fondamentale. L'aspect informatique référant aux divers types d'apprentissages automatiques, l'objectif de chacun d'eux et découvrir quelques-uns des algorithmes qu'ils utilisent fut également important. Même si tout au long du projet, tous n'ont pas été utilisés, l'étude faite sera utile pour la continuation du projet au cour d'un doctorat.

Le projet étant accepté, le chapitre 3 (page 29) a permis de présenter un modèle d'affaire, basé sur une proposition de valeur et valeur du marché actuel pour une postérieur commercialisation du projet. Celui-ci peut servir de base à un modèle ci-après, plus élaboré et détaillé. Avant quelconque implémentation, il est nécessaire de procéder à une analyse détaillée des données fournies et de ce qui va en être fait. Ce dernier procédé, décrit dans le chapitre 4 (page 35), a permis de prendre conscience de l'importance d'effectuer une analyse et ne pas partir directement dans l'implémentation. Après avoir analysé et détaillé tout ce qui doit être implémenté, il est nécessaire de produire une bonne documentation contenant les spécificités des composants utilisés et détails des scripts implémentés pour que de postérieurs ingénieurs puissent utiliser et comprendre le fonctionnement du projet, ces informations sont décrites dans le chapitre 5 (page 59).

Le dernier point et pas des moins importants, consiste en l'élaboration des tests et analyses de résultats obtenus. Un projet peut avoir d'excellents résultats mais si ceux-ci ne sont pas bien commentés ou expliqués, ils ne sont d'aucune utilité pour personne. Dans le chapitre 6 (page 91) sont décrits tous les tests faits, discriminer chaque choix de configuration retenue pour l'élaboration du modèle d'apprentissage automatique final et sont commentés les résultats obtenus notamment les résultats de spécificité et de sensibilité.

7.1 Objectifs atteints

Initialement, il a été nécessaire d'acquérir les concepts biologiques afin d'entrer dans le sujet de thèse proposé. La première partie du projet a donc été consacrée à l'acquisition des connaissances biologiques et à la compréhension du fonctionnement de la phagothérapie décrite dans l'état de l'art. Par la suite, une analyse des deux bases de données fournies a été faite (Phage_bact et DOMINE), afin de définir le type d'informations qu'elles contiennent, comment les utiliser et élaborer les scripts pour la génération des divers sets de données. L'étape suivante a consisté en une dissection qui permet de déterminer les configurations des divers sets de données utilisés ainsi que les paramètres utilisés pour l'élaboration d'un algorithme d'apprentissage automatique basé sur l'agrégation de modèles.

Une machine virtuelle *Ubuntu* 14 contenant *Anaconda*, l'IDE utilisé pour le développement des scripts, a été fournie. Il a été nécessaire de procéder à :

- L'installation du système de gestion de bases de données mySQL et y importer Phage_Bact et DOMINE ;
- Création de tableaux nécessaires à l'élaboration des variables dont :
    - Pour les variables basées sur les domaines d'interactions un tableau pour
        - Les domaines qui composent les protéines de chaque bactérie et bactériophage ;
        - Toutes les interactions de domaines pour chaque paire de protéine d'une bactérie et bactériophage ;
        - Les scores IPPs.
    - Pour les variables basées sur la composition chimique des séquences protéiques un seul tableau contenant les pourcentages de chaque acide aminé, composant chimique et poids de chaque protéine.

Tout au long du projet ont été réalisées plusieurs réunions afin de discuter des résultats obtenus, de l'avancement du projet et des étapes suivantes. Parmi celles-ci les plus pertinentes furent :

- La décision des sets de données à générer ;
- L'analyse des résultats obtenus lors de la première, seconde phase de tests et les discussions concernant l'approche d'agrégation de modèles ;
- L'analyse de résultats finaux.

En plus du développement du projet, un poster décrivant la méthodologie a été présenté à la conférence des SIB-Day 2016 à Bienne (Suisse). Ce dernier, disponible dans l'annexe nº 2 SIB et SIB-Days 2016, présente l'approche utilisée pour l'extraction des variables basées sur les domaines d'interactions et la composition chimique des séquences protéiques.

Il est possible de dire que les objectifs initialement proposés ont été respectés avec succès. Les résultats démontrés ont permis de consolider l'hypothèse de départ qui consistait à déterminer si l'approche par apprentissage automatique serait favorable à la résolution du problème des paires d'interactions entre bactéries et bactériophages. Ce projet montre à quel point le monde de l'informatique se veut de plus en plus interdisciplinaire liant ici la biologie et la médecine au domaine de l'informatique et donc de la bio-informatique.

7.2 Limitations et Travaux futurs

La principale limitation du projet réside en la base de données contenant les paires d'interactions entre les bactéries et bactériophages qui a été fournie au début du projet. Celle-ci fut l'unique source d'information dont le principal problème fut que sur les 1'065 interactions positives, plus de 930 étaient basées sur une unique bactérie, réduisant la diversité de la base de données. Une deuxième limitation advient du fait que la dernière version de la base de données DOMINE date de 2007 et ne contient probablement pas toutes interactions entre domaines.

L'algorithme d'apprentissage automatique actuellement développé possède un système de vote qui se base uniquement sur la somme des voix qu'obtiennent chacune des classes et en cas d'égalité attribue la classe positive. Il serait intéressant d'actualiser l'algorithme de sorte à ce qu'il tienne compte des performances de chacune des quatre techniques d'apprentissage automatique qui le composent et, en cas d'égalité, attribue la classe qui possède les meilleures performances. Il pourrait s'avérer intéressant de créer une approche d'agrégation de modèles sur laquelle une ou plusieurs des techniques utilisées soit responsable de la prédiction pour certains types (espèces) de bactéries. Il s'avérerai également intéressant de donner plus d'attention au prétraitement et nettoyage des données. Vérifié l'inexistence de corrélation et gain d'information de chacun des *bins*, se concentrer sur l'étape 3 de la méthodologie de (Fayyad et al. 1996) serai un point à tenir en compte dans le futur.

Tous les scripts ont été développés de sorte à ce qu'il soit possible de lire des données dans divers formats en ne modifiant que la méthode de lecture. Une possible amélioration consisterai à adapter les scripts au calcul multiprocesseur permettant de réduire le temps nécessaire de prédiction qui, pour l'algorithme actuel, est de plus de 2 jours avec les données utilisées tout au long du projet.

L'un des objectifs à long terme consisterait en l'élaboration d'un graphe où les nœuds représenteraient les bactéries et bactériophage et les segments représenteraient quant à eux, les interactions qui seraient constamment mises à jour. Il serait également intéressant de développer un outil en ligne qui pourrait être consulté par les médecins.

Le projet étant financé, les résultats obtenus vont servir à continuer le développement de ce dernier dans le cadre d'une thèse de Doctorat qui me permettra donc de poursuivre ce travail.

Références

Alfredson, T. & Hopkins, J., 2007. *Théorie et pratique de la négociation - Approche de la littérature*, Rome.

Ambapour, S., 2009. *Théorie des ensembles flous : application à la mesure de la pauvreté au Congo*,

Audebert, P., 2005. *Bien négocier* Éditions d.,

Barnes, J., 2015. Azure Machine Learning Microsoft Azure Essentials. In p. 13- 15.

Bassi, S., 2009. *Python for Bioinformatics*, Chapman and Hall/CRC.

Beckett, S.J. & Williams, H.T.P., 2013. Coevolutionary diversification creates nested-modular structure in phage-bacteria interaction networks. *Interface focus*, 3(6), p.20130033.

Benson, D.A., 2004. GenBank. *Nucleic Acids Research*, 33(Database issue), p.D34- D38.

Blake, J.A. et al., 2013. Gene ontology annotations and resources. *Nucleic Acids Research*, 41(D1), p.530- 535.

Bohineust, A., 2011. La recherche sur les antibiotiques peine à décoller. *Figaro*, p.2.

Brei, V.A. & Rossi, C.A.V., 2005. Confiança, valor percebido e lealdade em trocas relacionais de serviço: um estudo com usuários de Internet Banking no Brasil. *Revista de Administração Contemporânea*, 9(2), p.145- 168.

Campo, D., 2004. *A análise do valor percebido pelo cliente como ferramenta para a formulação de estratégias competitivas - uma aplicação de Conjoint Analysis*. Universidade Municipal De São Caetano Do Sul.

Chawla, N. V, 2005. Data Mining for Imbalanced Datasets: An Overview. *Data Mining and Knowledge Discovery Handbook*, p.853- 867.

Claveri, J.-M., 2007. *Bioinformatique* T.I. Faculté de Médicine de l'Université de la Méditerranée, éd., Marseille.

Coelho, E.D. et al., 2014. Computational prediction of the human-microbial oral interactome. *BMC systems biology*, 8, p.24.

Collman, J.P., 2003. NATURALLY DANGEROUS: Surprising Facts About Food, Health, and the Environment. Disponible ici: https://web.stanford.edu/~jpc/Chapter4.htm [Consulté le février 7, 2016].

Conti, J., 2013. Le retours des virus guérisseurs.

Coulon, A., 2015. Le CHUV se lance dans la phagothérapie. *Le Temps*, p.7.

Cui, G., Fang, C. & Han, K., 2012. Prediction of protein-protein interactions between viruses and human by an SVM model. *BMC Bioinformatics*, 13(Suppl 7), p.S5.

Dauzat, L., 2016. Business modèle – Proposition de valeur. , p.1- 6. Disponible ici: http://www.liliandauzat.com/business-model/business-modele-la-proposition-de-valeur/.

Demšar, J., 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7, p.1- 30.

Denamur, E. & Matic, I., 2006. Evolution of mutation rates in bacteria. *Molecular Microbiology*, 60(4), p.820- 827.

Denis, F., 2002. *Les bactéries, champignons et parasites transmissibles de la mère à l'enfant*, John Libbey Eurotext.

Denis, F. & Miclet, L., 2006. *Intelligence Artificielle : Les Systèmes Experts*. Université de Rennes 1.

Dias, M.G.C., 2004. *Revista do Curso de Administração da Faculdade da Serra Gaúcha*, Rio Grande do Sul.

Dominguez, S., 2000. O valor percebido como elemento estratégico para obter a lealdade dos clientes. *Caderno de pesquisas em administração, São …*, 7(4), p.12.

Donohue, M., 2012. Human genetics, dna replication, protein synthesis, mutations. , p.47. Disponible ici: http://pt.slideshare.net/mdonohue/human-genetics-dna-replication-protein-synthesis-mutations-12517784 [Consulté le février 7, 2016].

Dublanchet, A., 2014. Qu'est-ce que la phagothérapie ? *HEGEL - HEpato-GastroEntérologie Libérale*, 4(4), p.354- 370.

Dublanchet, A. & Fruciano, E., 2008. Brève histoire de la phagothérapie. *Médecine et Maladies Infectieuses*, 38(8), p.415- 420.

Ducoeurjoly, P., 2010. Phagothérapie : ll n'y a pas que les bonnes bactéries qui peuvent nous aider à conserver la santé. Les virus aussi. *Nexus*, 71, p.44- 45.

Dupont, C. & Audebert, P., 1994. *La négociation: applications et exercices*, Dalloz.

Dyer, M.D., Murali, T.M. & Sobral, B.W., 2007. Computational prediction of host-pathogen protein protein interactions. *Bioinformatics (Oxford, England)*, 23(13), p.i159- i166.

Eddy, S.R. & Wheeler, T.J., 2015. HMMER User ' s Guide. , (February), p.0- 77.

Edgar, R. et al., 2012. Reversing Bacterial Resistance to Antibiotics by Phage-Mediated Delivery of Dominant Sensitive Genes. *Applied and Environmental Microbiology*, 78(3), p.744- 751.

Fallis, A.., 2013. Introduction to Data Mining. *Journal of Chemical Information and Modeling*, 53(9), p.1689- 1699.

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. From data mining to knowledge discovery in databases. *AI magazine*, p.37- 54.

Fikes, B.J., 2014. New natural RNA function found. Disponible ici: http://www.sandiegouniontribune.com/news/2014/dec/15/double-stranded-rna-humans-gronemeyer/ [Consulté le février 7, 2016].

Fischetti, V.A., 2008. Bacteriophage lysins as effective antibacterials. *Current Opinion in Microbiology*, 11(5), p.393- 400.

Flores, C.O. et al., 2011. Statistical structure of host – phage interactions. *Proceedings of the National Academy of Sciences*, 108, p.E288.

Fraley, C. & Raftery, a E., 1998. *How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis*,

France Université Numérique, 2005. Introduction à la bioinformatique. , p.59.

Gabriel, M., 2010. Marketing na Era Digital. In R. Prates, éd. *Marketing Na Era Digital - Conceitos, Plataformas e Estratégias*. p. 31- 32.

Gallien, A., 2005. Code génétique. Disponible ici: http://svt.ac-dijon.fr/schemassvt/article.php3?id_article=410 [Consulté le février 7, 2016].

Gava, R., 2001. Desenvolvendo uma proposta de valor para uma empresa incubada: O caso da Millefolium. , p.688- 703.

Gay, C., 2015. Le Business Model Canvas | Graines de Startup. Disponible ici: http://www.grainesdestartup.fr/le-business-model-canvas.html [Consulté le février 6, 2016].

Gayet, S., 2016. L'augmentation de la résistance aux antibiotiques représente « un immense danger pour la santé mondiale ». *Altantico*, p.1- 4.

GenBank, 2015. GenBank Release Notes. Disponible ici: http://www.ncbi.nlm.nih.gov/genbank/statistics [Consulté le février 1, 2016].

Gerristsen, V.B., Barbié, V. & Durinx, Christine, Beckmann, J.S., 2016. Bioinformatique et médecine personnalisée : la Suisse pionnière.

Gil-Mohapel, J.M. & Rego, A.C., 2011. Doença de huntington: Uma revisão dos aspectos fisiopatológicos. *Revista Neurociencias*, 19(4), p.724- 734.

Gomes, C. & Gomes, L., 2004. Modelagem de aspetos qualitativos do processo de negociação. *Revista de Aministração Mackenzi*, 1, p.83- 103.

Griess, S., 2013. *Les problèmes posés par Bacillus cereus dans l'industrie agroalimentaire*,

Guérif, S., 2006. *Réduction de dimension en Apprentissage Numérique Non Supervisé*. université Paris 13.

Guthmiller, J.M. & Novak, K.F., 2002. Periodontal Diseases. In ASM Press.

Han, J., Kamber, M. & Pei, J., 2012. *Data mining: concepts and techniques*,

Institut national du cancer, 2012. Médecine personnalisée du cancer. *Institut National de la santé et de la Recherche Médicale*.

Japkowicz, N., 2011. Performance Evaluation for Learning Algorithms. Disponible ici: http://www.icmla-conference.org/icmla11/PE_Tutorial.pdf [Consulté le février 18, 2016].

Johnson, G.B. et al., 2011. Biologie - Version luxe. In De Boeck Supérieur, p. 361.

Kotsiantis, S., Zaharakis, I. & Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. , 31, p.249- 268.

Labrie, S.J., Samson, J.E. & Moineau, S., 2010. Bacteriophage resistance mechanisms. *Nature reviews. Microbiology*, 8(5), p.317- 327.

Larouss, 2016. Encyclopédie Larousse en ligne - biologie. Disponible ici: http://www.larousse.fr/encyclopedie/divers/biologie/27091 [Consulté le février 18, 2016].

Larousse, 2016. Bio-informatique.

Larranaga, P., 2006. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1), p.86- 112.

Leite, D. et al., 2016. *In silico prediction of phage-bacteria infection networks as a tool to implement personalized phage therapy*,

Lu, T.K. & Koeris, M.S., 2011. The next generation of bacteriophage therapy. *Current Opinion in Microbiology*, 14(5), p.524- 531.

Lubanovic, B., 2015. *Introducing python, modern computing in simple packages* First edit. O'Reilly, éd.,

Maletic, J.I. & Marcus, A., 2005. *Data Cleansing Data Mining and Knowledge Discovery Handbook*,

Marx, V., 2013. Biology: The big challenges of big data. *Nature*, 498(7453), p.255- 260.

Matsuzaki, S. et al., 2005. Bacteriophage therapy: A revitalized therapy against bacterial infectious diseases. *Journal of Infection and Chemotherapy*, 11(5), p.211- 219.

McNair, K., Bailey, B. a. & Edwards, R. a., 2012. PHACTS, a computational approach to classifying the lifestyle of phages. *Bioinformatics*, 28(5), p.614- 618.

Mitchell, A. et al., 2015. The InterPro protein families database: The classification resource after 15 years. *Nucleic Acids Research*, 43(D1), p.D213- D221.

Morin, Y. et al., 2006. Archive Larousse : Larousse Médical - papille linguale - papillomavirus. *Larousse*, p.732.

Mougenot, I., 2012. *FMIN366 Information Biologique : Chapitre d'introduction*, Montpellier.

Nathan, C. & Cars, O., 2014. Antibiotic Resistance — Problems, Progress, and Prospects. *New England Journal of Medicine*, 371(19), p.1761- 1763.

Nicola, S., Ferreira, E.P. & Ferreira, J.J.P., 2014. *A Quantitative Model for Decomposing & Assessing the Value for the Customer*. Faculdade de Engenharia da Universidade do Porto.

Nourani, E., Khunjush, F. & Durmuş, S., 2015. Computational approaches for prediction of pathogen-host protein-protein interactions. *Frontiers in microbiology*, 6(February), p.94.

O'Neill, J., 2014. Antimicrobial Resistance : Tackling a crisis for the health and wealth of nations. *Review on Antimicrobial Resistance*, (December), p.1- 16.

Oje, 2016. Bactérias patogénicas. Vírus podem vir a substituir antibióticos. *OJE - O Jornal Económico*.

Oliveira, T., Leite, D. & Marreiros, G., 2016. PersonalFit – Fitness App with intelligent plan generator. *C3S2E16*.

Oprean, C., 2010. *État de l'art sur les aspects méthodologiques et processus en Knowledge Discovery in Databases*,

Osterwalder, A. & Pigneur, Y., 2011. *Business Model Generation* Patrik van. T. Clack, éd., Rio de Janeiro.

Parham, P., 2003. Structure des anticorps et origines de la diversité des cellules B. In De Boeck, éd. *Le système immunitaire*. p. 31- 35.

Petsko, G.A. et al., 2008. Structure et fonction des protéines. In De Boeck Supérieur, p. 2- 6.

Phan, J.H., Quo, C.-F. & Wang, M.D., 2006. Functional genomics and proteomics in the clinical neurosciences: data mining and bioinformatics. *Progress in brain research*, 158, p.83- 108.

Pigneur, Y. & Fritscher, B., 2014. Business Model Design An Evaluation of Paper-based and Computer-Aided Canvases. In *Proceedings of the Fourth International Symposium on Business Modeling and Software Design*. SCITEPRESS - Science and and Technology Publications, p. 236- 244.

Piramuthu, S., 2004. Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research*, 156(2), p.483- 494.

Polikar, R., 2014. Ensemble learning. , 4(2009), p.1- 18.

Promega, 2010. *Amino Acids*, Madison.

Raghavachari, B. et al., 2008. DOMINE: A database of protein domain interactions. *Nucleic Acids Research*, 36(SUPPL. 1), p.656- 661.

Raghupathi, W. & Raghupathi, V., 2014. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2, p.3.

Rakhuba, D. V. et al., 2010. Bacteriophage receptors, mechanisms of phage adsorption and penetration into host cell. *Polish Journal of Microbiology*, 59(3), p.145- 155.

Rakotomalala, R., 2013. *Comparaison de populations - Tests paramétriques*, Lyon.

Ravat, F., Jault, P. & Gabard, J., 2015. Bactériophages et phagothérapie: utilisation de virus naturels pour traiter les infections bactériennes. *Annals of Burns and Fire Disasters*, XXVIII(March).

Reece, J.B. et al., 2013. *Campbell Biology 10th Edition*,

Reis, J. et al., 2015. Crescimento e renovação celular. In *Preparação para o Exame Final Nacional 2016 - Biologia e Geologia - 11.º Ano*. p. 142- 150.

Riou, B. & Landais, P., 1998. Principes des tests d'hypothèse en statistique: α, β et P. *Annales Françaises d'Anesthésie et de Réanimation*, 17(9), p.1168- 1180.

Rodrigues, N., 2014. Apresentação meta-aprendizado. Disponible ici: http://pt.slideshare.net/niltonrpereira/apresentao-metaaprendizado [Consulté le février 7, 2016].

Rusconi, F., 2011. *Manuel de spectrométrie de masse à l'usage des biochimistes* Editions T., Paris.

Samson, J.E. et al., 2013. Revenge of the phages: defeating bacterial defences. *Nature reviews. Microbiology*, 11(10), p.675- 87.

Sanjuán, R. et al., 2010. Viral Mutation Rates Viral Mutation Rates . , 84(July), p.9733- 9748.

Satizabal, H.-F. & Perez-Uribe, A., 2016. Machine Learning - Artificial Neural Networks. Disponible ici : http://www.alunaweb.net/andres/MLG/Labo2-ANN/ [Consulté le juillet 7, 2016].

Scholz, M., 2006. *Approaches to analyse and interpret biological profile data*. Universität Potsdam.

Scikit Learn, 2016. API Reference. Disponible ici: http://scikit-learn.org/dev/modules/classes.html#module-sklearn.neural_network [Consulté le juillet 7, 2016].

Seed, K.D., 2015. Battling Phages: How Bacteria Defend against Viral Attack. *PLOS Pathogens*, 11(6), p.e1004847.

Shaw, M.J. et al., 2001. Knowledge management and data mining for marketing. *Decision Support Systems*, 31(1), p.127-137.

Sheldon, R. & Moes, G., 2005. *Beginning MySQL*, Wiley Pub.

Somogyi, A., 2012. Maladies infectieuses - Tome 1: Infections parasitaires, virales et mycosiques, Volume 1. In Elsevier Masson, p. 1-2.

Sonnhammer, E. et al., 1998. Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Oxford University Press*, 26(1), p.320–322.

Stimec, A., 2011. *La négociation - 2e édition*, Dunod.

Tan, P.-N., Steinbach, M. & Kumar, V., 2006. Introduction to Data Mining. *Aging*, 1.

Terrapon, N., 2010. *Recherche de domaines protéiques divergents à l'aide de modéles de Markov cachées : application à Plasmodium falciparum*. Université Montpellier 2.

Tollari, S., 2006. *Image indexing and retrieval by combining textual and visual informations*. Université du Sud Toulon-Var.

Triola, M. & Triola, M.F., 2012. Statistiques non paramétriques. In *Biostatistique pour les sciences de la vie et de la santé: édition revue et corrigée*. Pearson Education France, p. 270-271.

Tsai, C.F. et al., 2009. Intrusion detection by machine learning: A review. *Expert Systems with*

Applications, 36(10), p.11994-12000.

Tuffery, S., 2012. *Data Mining et statistique décisionnelle: L'intelligence des données* Technip.,

Underwood, B.R. et al., 2006. Huntington disease patients and transgenic mice have similar pro-catabolic serum metabolite profiles. *Brain : a journal of neurology*, 129(Pt 4), p.877-86.

Ventureradar, 2016. Top 5 Companies Revolutionizing Healthcare with Machine Learning.

Vermeesch, J. & Matthijs, G., 2014. Mes gènes, mon identité ?: Comprendre la génétique et ses enjeux. In Mardaga, p. 27-30.

Ville, B. de, 2001. *Microsoft Data Mining: Integrated Business Intelligence for e-Commerce and Knowledge Management*, Digital Press.

Vinay, M. et al., 2015. Phage-Based Fluorescent Biosensor Prototypes to Specifically Detect Enteric Bacteria Such as E. coli and Salmonella enterica Typhimurium. *Plos One*, 10(7), p.e0131466.

Watson, J.D. & Berry, A., 2003. ADN, le secret de la vie. In Odile Jacob, p. 181.

Wills, S., 2015. DataSource (biojava 4.1.0 API). Disponible ici: http://www.biojava.org/docs/api/org/biojava/nbio/core/sequence/DataSource.html [Consulté le février 18, 2016].

Witten, I.H., Frank, E. & Hall, M. a., 2001. Data Mining Pratical Machine Learning Tools and Techniques. *Angewandte Chemie International Edition*, 40(6), p.9823.

World Heat Organization, 2014. WHO's first global report on antibiotic resistance reveals serious, worldwide threat to public health. Disponible ici: http://www.who.int/mediacentre/news/releases/2014/amr-report/en/ [Consulté le février 20, 2016].

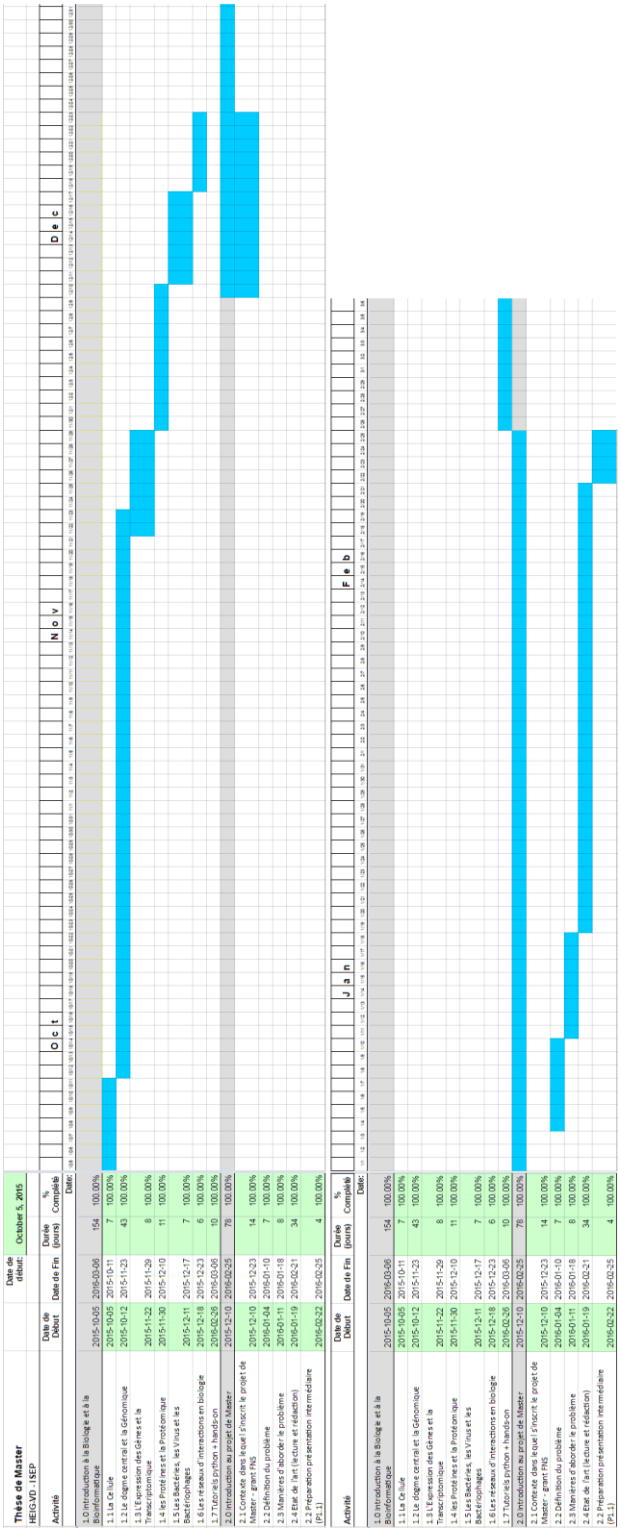
Zahoorullah, S.M., 2015. Biotechnology Within Your Teach. In *A Textbook of Biotechnology*. New Delhi: SmgeBooks, p. 2-31.

Zeng, Y. et al., 2009. Using the augmented Chou's pseudo amino acid composition for predicting protein submitochondria locations based on auto covariance approach. *Journal of theoretical biology*, 259(2), p.366-72.

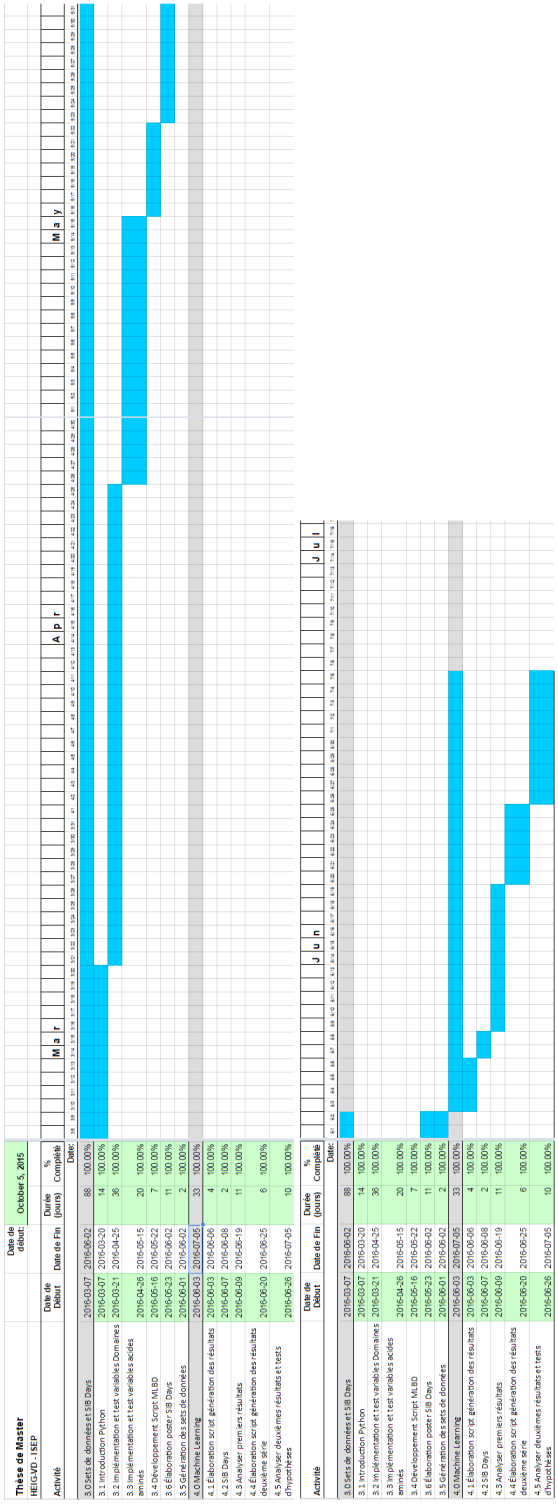
Zhang, S., Zhang, C. & Yang, Q., 2003. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6), p.375–381.

Annexe Nº 1 - Diagramme de Gant

Activités 1 & 2



Activités 3 et 4



Annexe Nº 2 – SIB et SIB-Days 2016

1—Abstract

The emergence and rapid dissemination of antibiotic resistance worldwide threatens medical progress and calls for innovative approaches for the management of multidrug resistant infections. Phage therapy might represent such an alternative. This re-emerging therapy uses viruses that specifically infect and kill bacteria during their life cycle to reduce/eliminate bacterial load and cure infections.

The success of phage therapy however relies on the exact matching between both the target pathogenic bacteria and the therapeutic phage. Therefore, having access to a fully-characterized phage library is necessary to start with phage therapy. An essential second step to conceive personalized phage therapy treatments is the capacity to predict the interactions between the target pathogen and its potential phage. To address this, we aim at developing predictive *in silico* models of phage-bacteria infection networks, using genomic features from sequenced phages and bacteria, and taking advantage of bioinformatics and machine learning techniques.

Using the publicly available information from Genbank and phagesdb.org, we were able to construct a dataset containing +1000 known phage-bacteria interactions with corresponding sequenced genomes. An equal amount of potential negative interactions were added to the dataset by considering the specificity of phage-bacteria interactions. We are currently extracting features from the genomes to build quantitative datasets to train machine learning models. These features include distribution of predicted protein-protein interaction scores, as well as proteins' amino acids frequency and chemical composition. Future work will focus on the development of ensemble machine learning models to optimize the predictive power of our methodology.

2—Context

Causes of antibiotics resistance

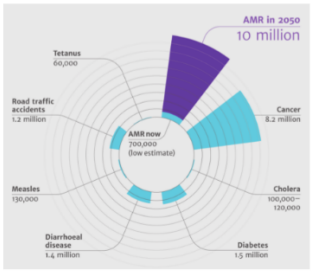
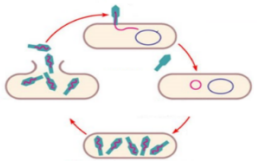


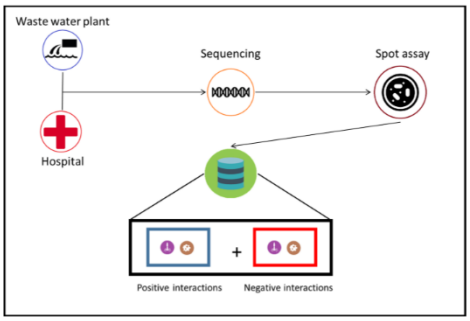
Image from [1]

Cycle of infection of a Lytic phage (results in bacterial death)

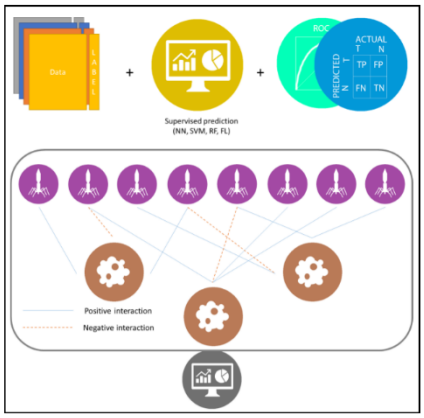


3—Overview of the project

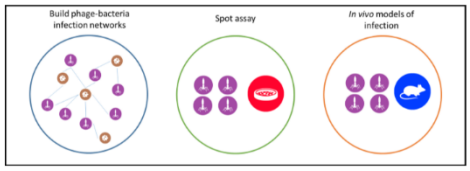
A—Bacteria and phages collection, sequencing and labeling.



B—*In silico* predictions of phage-bacteria interactions. (neural network, SVM, fuzzy logic, random forest)



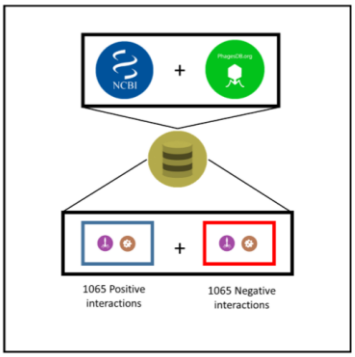
C—Prospective and validation study.



Computational objectives

1—**features extraction** -> 2—features selection -> 3—predictive model -> 4— test the model

4—Data used to start working on the computational models



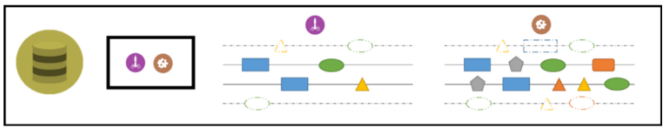
- Collect all publicy available phage genoms from NCBI [2] and Phagesdb.org [3];
- If the host bacteria is sequenced (NCBI[2]), retrieve genome as well;
- Predict genes using GeneMarkS [4] if necessary;
- constitute positive dataset of 1065 entries. A negative dataset of equal size was built artificially by matching phages with bacteria from a different species than their know host.

Create two type of data sets:

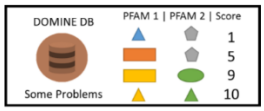
- Based on domains interactions (5[a])
- Based on proteins sequence (5[b])

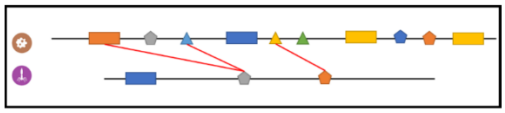
5[a]—Features extraction with protein domains

Predict domains using HMMER-PFAM [5] for all proteins of all phages and bacteria in the database

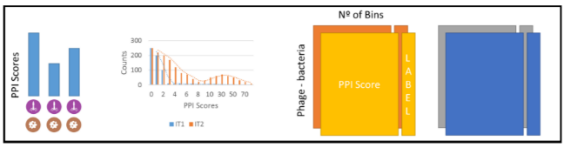


Identify interacting domains (DOMINE [6])

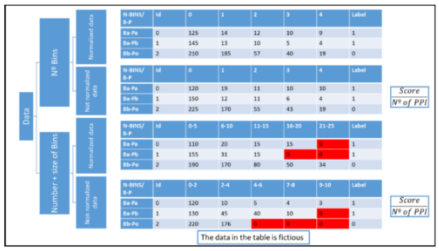




Build data set from PPI scores: use histogram of scores to define a binning and get the same number of features for each pair of phage-bacteria



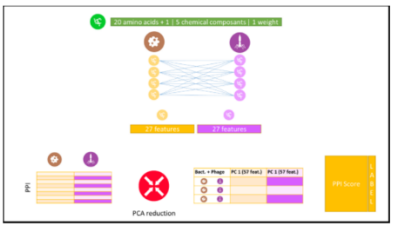
Example of a fictious dataset obtained whit this procedure



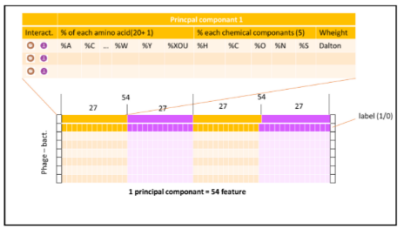
5[b]—Data Set by Protein sequence

Combination of all proteins between each bacteria and phage

- Extract % of amino acid, % chemical components (C,H,O,N,S) and weight -> 27 features
- Concatenate both vectors to get a PPI -> 27 + 27 = 54 features
- For a given pair phage-bacteria we obtain a matrix of size -> Nº PPI × 54
- We use PCA reduction to reduce the number of features per phage-bacteria to ≈ 100 features



Structure of the data set



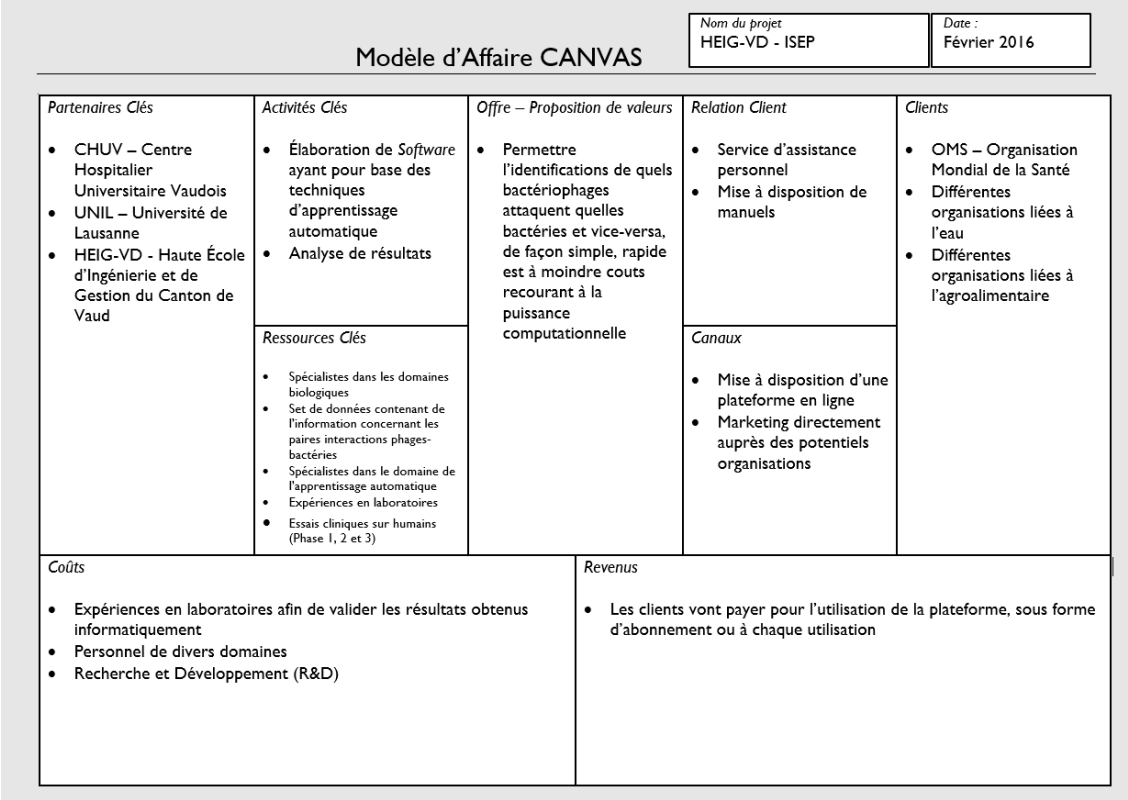
6—Future work and References

Future work:

- Select the data set to use in machine learning;
- In each data set, remove the redundant/correlated variables and those that provide little information gain;
- Select the tests to assess model performance;
- Select the machine-learning models to build a predictive model;
- Assess model performance.

References:
[1] Antimicrobial Resistance : Tackling a crisis for the health and wealth of nations; [2] http://www.ncbi.nlm.nih.gov/; [3] http://phagesdb.org/; [4]http://www.ncbi.nlm.nih.gov/genomes/MICROBES/genemark.cgi; [5] http://www.ebi.ac.uk/Tools/hmmer/search/hmmscan ; [6] http://domine.utdallas.edu/cgi-bin/Domine; [Features extraction based on article by E. Coelho] Computational prediction of the human-microbial oral interactome

Annexe Nº 3 – Modèle d'affaire CANVAS



Modèle d'Affaire CANVAS

Nom du projet	*Date :*
HEIG-VD - ISEP	Février 2016

Partenaires Clés	*Activités Clés*	*Offre – Proposition de valeurs*	*Relation Client*	*Clients*
• CHUV – Centre Hospitalier Universitaire Vaudois • UNIL – Université de Lausanne • HEIG-VD - Haute École d'Ingénierie et de Gestion du Canton de Vaud	• Élaboration de *Software* ayant pour base des techniques d'apprentissage automatique • Analyse de résultats *Ressources Clés* • Spécialistes dans les domaines biologiques • Set de données contenant de l'information concernant les paires interactions phages-bactéries • Spécialistes dans le domaine de l'apprentissage automatique • Expériences en laboratoires • Essais cliniques sur humains (Phase 1, 2 et 3)	• Permettre l'identifications de quels bactériophages attaquent quelles bactéries et vice-versa, de façon simple, rapide est à moindre couts recourant à la puissance computationnelle	• Service d'assistance personnel • Mise à disposition de manuels *Canaux* • Mise à disposition d'une plateforme en ligne • Marketing directement auprès des potentiels organisations	• OMS – Organisation Mondial de la Santé • Différentes organisations liées à l'eau • Différentes organisations liées à l'agroalimentaire

Coûts	*Revenus*
• Expériences en laboratoires afin de valider les résultats obtenus informatiquement • Personnel de divers domaines • Recherche et Développement (R&D)	• Les clients vont payer pour l'utilisation de la plateforme, sous forme d'abonnement ou à chaque utilisation