Published in *Proceedings of 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, Guayaquil, Ecuador, 11-15 november 2019, which should be cited to refer to this work.
DOI: 10.1109/LA-CCI47412.2019.9037032

Applying one-class learning algorithms to predict phage-bacteria interactions

Juan Fernando López
Department of Electrical and Automation
Universidad Autónoma de Occidente
Cali, Colombia
juan.felo9@gmail.com

Jesús Alfonso López Sotelo
Department of Electrical and Automation
Universidad Autónoma de Occidente
Cali, Colombia
jalopez@uao.edu.co

Diogo Leite
Computational Intelligence for Computational Biology
School of Bussiness and Engineering Vaud (HEIG-VD)
Yverdon-les-Bains, Switzerland
diogo.leite@heig-vd.ch

Carlos Peña-Reyes
Computational Intelligence for Computational Biology
School of Bussiness and Engineering Vaud (HEIG-VD)
Yverdon-les-Bains, Switzerland
carlos.pena@heig-vd.ch

***Abstract* — The need to predict phage-bacteria interactions is a nowadays concern to overcome bacterial resistance issue; public genome databases contain highly imbalanced datasets which have hindered this task. Throughout this paper we will investigate, implement and evaluate One-Class Learning algorithms in order to predict phage-bacteria interactions using only positive samples. We will use the programming language Python aided by Scikit-Learn, Tensorflow and keras to develop the machine learning models and test them with real phage-bacteria interactions datasets. We trained the models using cross validation technique generating a gridsearch with all the datasets to find several combinations of hyperparameters available. Furthermore, we optimized those hyperparameters by using Pareto fronts based on seven different performance metrics, improving the efficiency of each algorithm for a given dataset. To refine each algorithm's performance separately we used the ensemble learning technique with an odd number of algorithms by simple voting. Finally, we managed to achieve an overall performance of 80% in predicting phage-bacteria interactions trained only with positive classes, this percentage in practice means that when a patient has an infection resistant to antibiotics, we have 80% of saving the life rather than maybe a 0% while finding the correct phage for the pathogenic host.**

Keywords— phage-bacteria interaction, One-Class Learning, optimization, imbalanced datasets.

I. Introduction

The misuse of antibiotic drugs contributes to the emergence and rapid dissemination of antibiotic resistance worldwide, threatening medical progress [1]. A re-emerging therapy, dubbed phage-therapy, might represent a novel alternative. Phage-therapy is based on viruses (bacteriophages) that specifically infect and kill bacteria during their life cycle. The success of phage therapy mainly relies on the exact matching between the pathogenic bacteria and the therapeutic phage. However, this is a time-consuming process achieved in laboratories and time is a precious and critical resource in a clinical context. Hence, the fast identification of potential phage candidates capable of dealing with a given bacteria is essential for using phage-therapy in routine. Machine learning algorithms trained on public genome databases constitute a promising approach to achieve this goal. Unfortunately, public databases contain highly imbalanced interaction data (i.e., mostly positive phage-bacterium interactions); making it harder to use classic machine learning algorithms that needs relatively balanced classes to work. To address this problem, we are exploring the use of One-Class learning methods, which are robust tools to deal with imbalanced datasets.

Throughout this paper, Machine Learning tools will be used to process, analyze, and generate data on phage-bacteria pairs in order to improve the efficiency of phage therapy. This procedure consists in introducing selected phages into an infected organism and it will only attack the bacteria [2]. However, this treatment has several limitations and one of them is to find the specific bacteriophage for a bacterium, since there are estimated to be trillions of bacteriophages but each one attacks very specific bacteria. Currently, the process of finding phage-bacteria pairs is carried out in laboratories through infection tests, which can take between eight hours and two days of testing [3].

Therefore, when performing phage therapy, one of the most important factors is to find the correct phage for the treatment; if the infection is very high there would not be enough time to perform tests in the laboratory, demanding to have detailed information about the different bacteria-bacteriophage pairs and thus being able to respond quickly to any situation.

This is where Machine Learning begins to play a very important role in this problem, giving tools to classify pre-existing data of phage-bacteria pairs and predict missing pairs, being able to generate a detailed database so that it can be accessible to people who need this type of treatment.

Unfortunately, there is also another obstacle for this approach. The pre-existing data of phage-bacteria pairs can be found in public databases such as Genbank and phagesdb.org but these datasets contain highly imbalanced interaction data. This means that they contain only positive phage-bacteria interactions and in order to apply classic Machine Learning algorithms the data must be relatively balanced (i.e.

approximately the same amount of positive and negative phage-bacteria interactions).

To address this problem, One-Class learning methods, which are robust tools to deal with imbalanced datasets, are going to be explored, studied, implemented and validated through this project.

To clarify, this paper is a follow up of the work made by Leite *et al* [4] which consists in the omics analysis of some bacteria and their respective phages (DNA structure, transcription, proteomics) in order to process this data through some of the most used machine learning algorithms (multilayer perceptron, decision tree, N-nearest neighbors and support vector machines). Good results and quite accurate predictions were obtained, however, there were many outcomes such as the imbalanced dataset they used, which led to the proposal of trying new methods to approach the issue.

II. BACKGROUND

A. *One-class learning*

Often, real data in nature and in daily problems gives us only a representation of one type of class' targets or gives us a very imbalanced dataset. This means that classes are not represented equally and that there are many more samples of one class than the other, which makes one of the classes insignificant [5]. This is a big problem in classic supervised learning as it mainly works with binary classification and need two similarly distributed classes to perform correctly; one of the most common issues is the accuracy paradox which tells that a model has an excellent accuracy, but it is only reflecting the predominant class' accuracy.

Specifically speaking, the public databases of phage-bacteria interactions from Genbank and phagesdb.org contain only positive interactions. This is because the laboratories that contributed to the databases were not supposed to register negative interactions of phages and bacteria but only register the positive ones. This is a perfect example of one-class classification where only examples from one of the classes are available but there's also the need for predicting the other unknown class.

Next, we are briefly describing the five one-class learning algorithms used in this paper; it is important to recognize that most of them have similarities to classic supervised learning techniques such as K-Nearest Neighbors [6] and Multilayer Perceptron [7], amongst others.

*1) Replicator Neural Network (RNN):*The RNN [8] is a type of neural network that, as its name indicates, replicates the input data in order to learn a representation of it. The structure is very similar to an MLP but as it only uses samples from one class, the target of the net is the same as the input of it. It sounds redundant, but this structure allows the neural net to learn how to reconstruct the input very well, meaning that it will learn to reconstruct data of one single class; when the network tries to reconstruct data from another class, it won't know how to and will produce a higher reconstruction error, sign of an out-of-class instance.

2) Local Outlier Factor (LOF): The idea of the LOF [9] is very similar to *K-Nearest Neighbors*, which classifies new datapoints based on a similiraty measure, in this case based on how their neighbors are classified [6],but its difference is that it considers the local density deviation of each sample as compared to its neighbors. This means that an outlier is detected when it has much lower density deviation than its K neighbors while inliers have much more density as they are thought to have almost similar characteristics.

3) Isolation Forest (iTree): The iTree [10] is a random forest algorithm but the difference is that they are only trained with one of the classes trying to isolate, as its names says, the outlier samples. The isolation process occurs when the length path or the decision is made very near the root meaning that the random partitioning of the trees produces shorter paths for anomalies. This early partitioning is due to the high difference between the features of inliers and outliers, and normally, decision trees converge shorter when the feature-value is distinguishable from normal values.

4) One Class Support Vector Machine (OC-SVM): The OC-SVM [11] is very similar to the regular SVM as it applies kernel-based feature space transformation and then tries to maximize a hyperplane in order to separate or classify the data in an optimal way, but as it only uses data points from one class, the process of training is different. In the feature space, it generates a boundary from the origin of the data points and maximizes the distance between the boundary (hyperplane) and the origin creating a separation between inliers and outliers.

5) Elliptic Envelope (EllEnv): The elliptic envelope [12] is based on the minimum covariance determinant (MCD) method which is a robust estimator of multivariate data. It uses the Mahanalobis distance as a discriminant factor for high-dimensional data as well as the minimum volume ellipsoid (MVE). The foundation of this process is fitting the known data into a Gaussian density and then being able to calculate the Mahanalobis distance of each data point in order to optimize the minimum covariance determinant.

B. *Phage-Bacteria Interaction:*

Phage primary goal is to infect the bacteria; it means that after moving onto the bacterium it will produce new phages and then release them into other bacteria. Regularly this cycle is called lysis which consists in destructing the outer layer of the bacteria and releasing new phage particles that were formed inside the infected cell, and then these phage virions will start infecting new hosts [13].
The receptor-binding proteins (RBP) from the phages are the ones that allow them to bind to the receptors of a bacterium.

There are different types of RBP and some of them are very specifically, it means that every phage can only bind to a reduced number of bacteria. Their infectivity can vary drastically even across bacterial strains of the same species. Therefore, as said before, it is very important for phage therapy, to find the exact match between the therapeutic phage and the host bacteria.

Here is where protein-protein interactions (PPI) takes place because the relation between a phage and its host is mainly due to the interaction of their encoded proteins. Bacteria can encode an average of 3500 proteins while phages can express in average 75 proteins, resulting in approximately 262,500 candidate PPIs for each phage-bacteria interaction (this number is different for each pair, so some post-processing is necessary to equalize the interaction's features). Moreover, in order to analyze the PPIs of each phage-bacteria pair it is necessary to analyze a functional subunit of the proteins: a domain. This, since a PPI occur when one or more bindings between pairs of their domains are done. This other type of interaction is called domain-domain interactions (DDI) [4].

PPI and DDI take an important role through this paper as they are going to let us generate the features to train the algorithms later.

III. METHODOLOGY

Several datasets of phage-bacteria interactions are going to be used in which PPI and DDI are contained, being divided into different numbers of features (e.g. 5, 10,15 or 20 features), also another dataset is going to be used but presenting the chemical composition of each RBP in the interaction [4].

The collection of interactions was made from two databases (PhageDB and GenBank) at strain level. The negative interactions for the validation and test process were assumed by the fact that phages are very specific to each strain of bacteria; so, if a phage interacts with a specie of bacteria means that no negative interaction is made, meaning that phages only attack one bacterial specie. Equilibrium is made by generating the same amount of negative interactions as positive ones.

A database call DOMINE was used to extract each DDI which is described as a score of each interaction. PPI score is calculated as a result of the sum of all its DDI scores. Different datasets were made from these interactions by dividing the frequencies of scores into different number of bins or different size of bins and considering or not PPI's with score zero.

Besides of these datasets, an additional dataset based on the chemical composition of both phage and bacterium proteins is used to train the machine learning algorithms. 21 features of this dataset consist in the relative abundance of amino acids in the protein sequence while other five features consist in the abundance of some important elements such as oxygen, carbon, hydrogen, nitrogen, etc. One more feature was extracted from the molecular weight of the protein and as each interaction consists of two proteins, there were 54 features in total. But also, as each phage-bacteria interaction has approximately 250 thousand PPI's, the dimensionality of this dataset was higher than expected, ergo a dimensionality reduction technique was made (the mean average and the average of the 54 features of each organism across all their proteins) resulting in only 108 features for the last dataset [4].

Each of the resulting datasets contains 4594 samples, which are split in training, test, and validation sets as illustrated in Figure 1. Note that one-class learning is performed only on positive interactions, but for test and validation purposes, where 10-fold cross validation is applied, it is necessary to include negative interactions. As mentioned before, the databases do not contain negative interaction data, so for this phase we used surrogate interactions generated as described in [4]. Metrics such as sensitivity (for both training and test set), specificity, accuracy, positive predictive value (PPV), negative predictive value (NPV) and f1-score are going to be extracted out of this process.

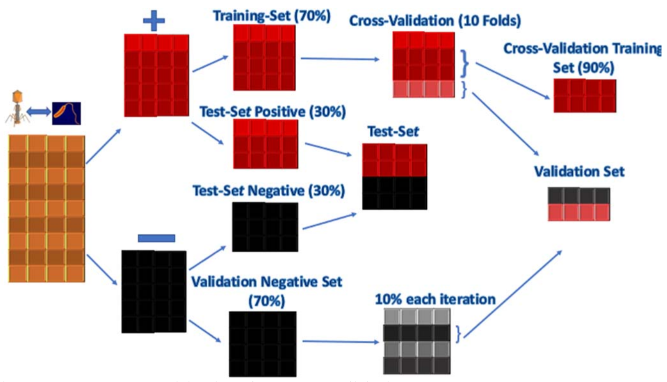

Figure 1. Dataset partitioning for cross-validation

Each of the algorithms described before are going to be implemented in Python using machine-learning libraries such as scikit-learn [14], Keras [15] and Tensorflow [16]. A grid search is implemented for each algorithm in order to select parameters for the best performance considering each of the metrics. Finally, the search is refined using Pareto optimality as explained later and the performance of each algorithm is improved by using an ensemble learning technique [17]. Heatmaps are also used for a better understanding and visualization of the grid search process.

IV. RESULTS AND ANALYSIS

We are going to show the training results of RNN model and its gridsearch performance in 15 different datasets. For simplicity we are only showing the F1-Score and accuracy performance presented through heatmaps; these were made with the results of the cross-validation process where each cell represents each combination of hyperparameters in each corresponding dataset (y-axis are the different datasets used

and x-axis the index of each combination of hyperparameters, i.e. iterations).
Black represents a performance of 1 (being darker colors higher performance) and white represents a performance of 0 (being lighter colors a worst performance); The fuchsia color represents a performance of 0.5 approximately. The heatmaps are a way for better understanding the behavior of the gridsearch in a visual way but the exact selection of the hyperparameters combinations is made furthermore.

TABLE 1. HYPERPARAMETERS CONFIGURATION

One-Class SVM (800 iterations)	Gamma Degree Nu Kernel	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1] [1,2,3,4] [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1] ['poly','rbf']
Isolation Forest (798 iterations)	N_estimators Max_samples Max features	[1,5,10,20,50,70,100,120,150,180,200,220,230,250,280,300, 350,400,500] [1,5,10,50,70,100,150,180,200,230,250,280,300,1500] [0.5,0.8,1]
RNN (80 iterations)	Layer_div Thresh Epoch	[2,3,4,5] [0.1,0.2,0.3,0.4,0.5] [50,100,150,200]
Elliptic Envelope (30 iterations)	Support_fraction Assumed_centerd	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,1.1,1.2,1.3,1.4,1.5] [True, False]
LOF (48 iterations)	N_neighbors Algorithm	[1,5,10,20,40,50,80,100,200,350,400,500,750,800,900,1000] ['ball tree', 'kd tree', 'brute']

The different configurations of parameters are shown in table 1, each one representing one iteration in the gridsearch process. Notice that the incremental steps are different for each, this because each one has different range. Each iteration for each algorithm were saved in an Excel spreadsheet letting us search easily the given iteration number corresponding to each combination of hyperparameters.

In figure 2 we can remark that the RNN f1-score in the dataset DS_ZB108 was almost the same and high (0.75 approximately) for every parameter's combination while for the DS_CH, very low f1-score were obtained. We were expecting to get a similar behavior on the accuracy heatmap but in figure 3 we observe that every combination in all the dataset were approximately the same, obtaining the best accuracy with the DS_CH dataset in the first combinations and in the last ones.

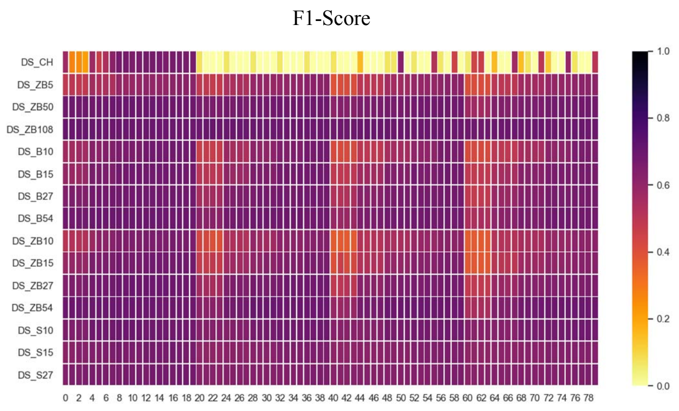

Figure 2. Heatmap for RNN F1-Score.

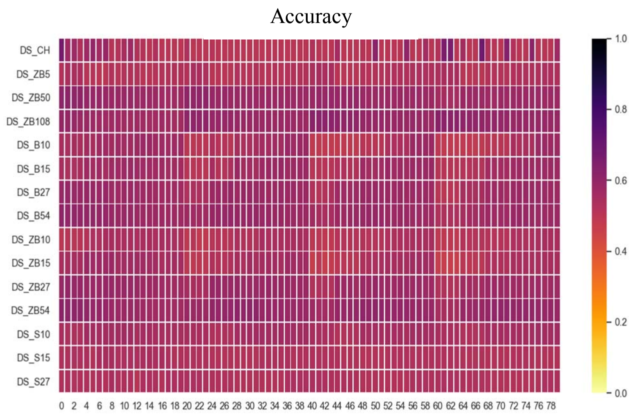

Figure 3. Heatmap for RNN Accuracy

As we said earlier, it is not enough with the heatmap process to choose the suitable hyperparameters for each algorithm as we have an imbalanced dataset and some of the previous metrics are not very representative. Hence, we are implementing the Pareto fronts next to optimize the hyperparameters for the seven metrics used in the project and find a balance between the optimal performance and choosing this combination of hyperparameters for the final training/test phase.

Following, the Pareto fronts of Accuracy and PPV relation (the line in red) is shown for the One-Class SVM trained on the chemical composition dataset (CH), even though we are only showing one algorithm's fronts for two of the performance metrics, the process has been made to each one of the algorithms to determine which combination of parameters suits best for each dataset with all of the metrics mentioned before.

For visual effects the following fronts are going to be shown in two dimensions, meaning that each metric is going to be related with each other to calculate the fronts for each relation, but in practice, the fronts are calculated in seven dimensions (as there are seven metrics) and the best combination of parameters (if there is more than one in each front) is chosen by the higher accuracy. The blue dots represent each combination of hyperparameters.

We can observe that in figure 4, in the PPV vs. Accuracy subplot, we got a Pareto front of approximately nine points (out of 800 combinations of hyperparameters) in which each of these points are combinations of hyperparameters that are Pareto efficient for the mentioned metrics because they have the higher PPV without being worst in Accuracy and vice versa. In this case, the need of a "choosing criteria" must be done and it could be choosing the point that has higher Accuracy or higher PPV (as we can only choose one combination of hyperparameters that belongs to the Pareto front to train each algorithm). For this, we select the one that has higher accuracy. This decision was because the dataset used for the validation phase is equally distributed (equal positive and negative samples even though negative ones were assumed), making accuracy a more suitable metric.

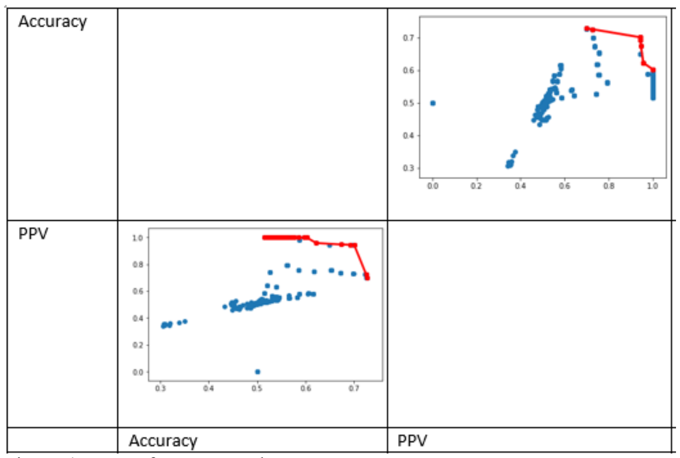

Figure 4. Pareto fronts example

For example, with the Elliptic Envelope algorithm and a given dataset, the best combination of hyperparameters according to the gridsearch process were "TRUE" for the "assumed centered" one and 0.6 for the "support fraction". But after applying Pareto fronts to the algorithm, we find out that the most optimal combination considering not only f1-score and accuracy (as we did in the heatmaps), but all the seven metrics were "FALSE" and 0.6 respectively.

Having selected the optimal hyperparameters for each algorithm, the results of the algorithms are going to be shown in the next tables (2, 3, 4 and 5) for the Chemical composition dataset which was the dataset that presented the best average results.

The higher results for each metric are highlighted in bold-font and the bolded-font algorithms are the three best ones for this dataset. Following, the ensemble learning results with all the five algorithms and then the ensemble learning using only the three best algorithms. Scores in bold font in table 4 and 5 are the scores which are improved towards the algorithms used.

TABLE 2. FINAL HYPERPARAMETERS SELECTED BY PARETO FRONTS FOR CHEMICAL COMPOSITION DATASET

Algorithm	*Parameters*			
One-Class SVM	*Kernel* = "poly"	*Degree* = 3	*Nu* = 0.1	*Gamma* = 1
Isolation Forest	*N_estimators* = 1	*Max_samples* = 1500	*Max_ features* = 0.3	
Elliptic Envelope	*Assume_ centered* = True	*Support_fraction* = 1.5		
LOF	*N_neighbors* = 100	*Algorithm* = "brute"		
RNN	*Layer_division* = 3	*Threshold* = 0.1	*Epochs* = 100	

TABLE 3. RESULTS FOR CHEMICAL COMPOSITION DATASET

Algorithm	Accuracy	F1-Score	Sensitivity	Specificity	PPV	NPV
ELLEnv	**79.7%**	**80.7%**	84.9%	**74.4%**	**76.9%**	**83.1%**
iForest	68.0%	73.3%	87.4%	48.5%	63.0%	79.3%
RNN	75.6%	78.1%	86.8%	64.2%	70.9%	69.3%
LOF	53.9%	67.0%	**93.8%**	14.1%	52.2%	69.3%
OC-SVM	72.2%	73.7%	77.6%	66.9%	70.2%	74.8%

TABLE 4. RESULTS FOR ENSEMBLE LEARNING WITH ALL ALGORITHMS

Accuracy	F1-Score	Sensitivity	Specificity	PPV	NPV
72.7%	76.7%	89.6%	55.8%	67.1%	**84.2%**

TABLE 5. RESULTS FOR ENSEMBLE LEARNING WITH THE THREE-BEST ALGORITHMS (ELLENV, RNN AND OC-SVM)

Accuracy	F1-Score	Sensitivity	Specificity	PPV	NPV
82.3%	**83.2%**	**87.5%**	**77.1%**	**79.3%**	**86.1%**

Through the above results we can observe that using all algorithms for the ensemble learning, only one metric was improved but using the three-best algorithms for it, all metrics were improved showing best results for predictive purposes.

Also, we can observe that in overall, the algorithms show a better sensitivity than specificity, this being expected because of the way the One Class Learning algorithms are trained: only with the positive class; meaning that they are more likely to detect positive interactions than negative ones.

Even though, they are only trained with positive class, the NPV values are often higher than the PPV, meaning that the algorithms are most likely to correctly identify negative interactions than positive interactions, this is expectable too as the sensitivity is higher meaning the algorithm is less likely to predict a negative interaction as a positive one, so they are going to be predicted as negative ones correctly.

In other words, for better understanding of the results, we can clarify that almost all sensitivity scores are very high, being 90% or higher and as we said before, due to the training process of the algorithms they are very likely to correctly identify positive interactions as such. On the other hand, it's seen that these algorithms tend to overfit the data, expressing specificity results less than 50% meaning that negative interactions are misclassified as positive ones. Negative interactions classified as such tend to be correctly classified though, this based on the NPV scores which are relatively high (70% above), meaning that the ones that are classified as negative ones are more likely to be truly negative interactions.

V. CONCLUSIONS

The main contribution of this work was achieved by successfully investigating, implementing and applying one-class learning algorithms to phage-bacteria interactions using their omics features, which is a methodology that hasn't been implemented before with the same tools and techniques used here.

We want to remark the true meaning of the performances of the algorithms, whether separated or in ensemble learning, which gives us approximately 80% of confidence when finding a correct phage for a given bacteria being trained with only positive classes; this percentage in practice means that when a patient has an infection resistant to antibiotics, we have 80% of saving his life rather than maybe a very much less percentage of success while finding the correct phage for the bacteria in the laboratory.

One aspect of the project to also highlight because of its novelty, is the process of refining the methodology using several techniques such as gridsearch to find the combinations of hyperparameters, Pareto efficiency to find the optimal combination of hyperparameters for each algorithm and each dataset and also using ensemble learning of two types (all algorithms and three-best algorithms) to try to improve the individual results of each algorithm, which allowed to efficient and trustworthy predictions of phage-bacteria interactions.

We showed that One Class Learning algorithms are a very successful approach in dealing with imbalanced datasets, specifically speaking in our phage-bacteria interaction datasets, showing very promising results (i.e. getting high performance metrics in overall using ensemble learning approach) and leading the way to more types of applications.

We want to highlight the performance of the Replicator Neural Network which was the best algorithm of the five we tried, showing very promising results in each test as well as always being part of the three-best algorithm in every three-best ensemble learning approach.

VI. FUTURE WORK

Foremost, the next step will be trying to use real negative interactions datasets tested in labs in order to try the previous process and validate the results. The process can be used as a filter, to confirm the real negative interactions, or also it can be used as a predictor in order to predict negative and positive interactions.

Also, trying other One Class Learning algorithms with the same datasets to observe differences in performance and simplicity. Another important task is trying to observe or analyze the structure of the datasets and their behavior (distribution, prevalence, density) to know what types of algorithms are best suitable and to refine them better.

At finally, trying to implement a more efficient and reliable way to select the appropriate hyperparameters for the algorithms, because of the complexity of Pareto fronts and the final selection based on accuracy might affect the whole performance.

REFERENCES

[1] World Health Organization. "Antibiotic Resistance, descriptive note". WHO press release, October 2017.

[2] R. Orrego. "Fagoterapia: alternativa para el control de enfermedades bacterianas". Salmonexpert, Santiago, Chile, June 18th 2015.

[3] E. Mullin. "Virus e inteligencia artificial se unen contra las bacterias resistentes". MIT Technology Review. Boston, USA, February 1st 2018.

[4] D. Leite *et al*. "Computational prediction of host-pathogenic interactions through omics data analysis and machine learning". Springer-Verlag Berlin Heidelberg, 2011.

[5] D. Johannes. "One-Class Classification: Concept-Learing in the absence of counterexamples". Technological University of Delft, ISBN: 90-75691-05-x. 2001.

[6] D. Subramanian. "A simple introduction to K-Nearest Neighbors Algorithm". Towards Data Science. June 8th 2019.

[7] S. Hayking. "Neural Networks and Learning Machines". 3rd edition. Ontario, Canada, 2009.

[8] S. Hawkins, H. HE, G. Williams, and R. Baxter. "Outlier detection using replicator neural networks". CSIRO Mathematical and Information Sciences.

[9] M. Breunig, HP. Kriege, R. NG, and J. Sander. "LOF: Identifying Density-Bases Local Outliers". International Conference on Management of Data, Dallas, USA, 2000.

[10] F. Liu, K. Ting, and Z. Zhou. "Isolation Forest". Eight IEEE International Conference of Data Mining. 2008.

[11] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylort, and J. Platt. "Support Vector Method for Novelty Detection ". Conference in advances in neural information processing systems, pp.582-588, 2000.

[12] P. Rousseeuw and K. Van Driessen. "A Fast Algorithm for the Minimum Covariance Determinant". Tehcnometrics Vol. 41, No. 3, pp. 212. Belgium, 1999.

[13] S. Abedon. "Introduction to Bacteriophages". Phage-therapy.org. August 28th 2016.

[14] F. Pedregosa *et al*. "Scikit-learn: Machine Learning in Python". JMLR 12, pp. 2825-2830, 2011.

[15] F. Chollet *et al*. "Keras". GitHub, 2015.

[16] M. Abadi *et al.* "Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

Z. Zhou. "Ensemble Learning". National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. 2012.