HIV-1 Cleavage Prediction

Machine Learning for Physicists
Technische Universität Dortmund

Simone Chiarella

July 31, 2023

Contents

1 Introduction **2**

2 Data set **2**

3 Methods **3**
3.1 Data Preprocessing 4
3.2 Multi-Layer Perceptron Classifier 4
3.3 *k*-NN and Logistic Regression Classifiers 6

4 Results **7**

5 Summary **10**

References **10**

1 Introduction

The human immunodeficiency viruses (HIV) are two species of retrovirus that infect humans and cause acquired immunodeficiency syndrome (AIDS), a condition in which the failure of the immune system allows life-threatening opportunistic infections and cancers to thrive. Worldwide the predominant and most dangerous one is HIV-1, due to its high infectivity with respect to HIV-2 [1].

Generally speaking, when a virus enters a cell, it replicates its genome, which involves the synthesis of viral mRNA and proteins [2]. One of these proteins is HIV-1 protease, an enzyme involved with peptide bond hydrolysis, that is essential for the life-cycle of HIV [3]. In fact, HIV-1 protease cleaves (i.e. cuts the bonds between amino acids) newly created polyproteins to produce the mature protein components of HIV virion, which is the infectious form of the HIV-1 virus outside of the host cell. Because HIV-1 protease binds to these polyproteins in octapeptide length before cleavage, vulnerable locations in a protein substrate are referred to as octapeptide regions, which are made up of eight amino acids in sequence [4].

Predicting the cleavage site in a substrate is important for the development of HIV-1 protease inhibitors, which are molecules that bind tightly to the HIV-1 protease making it impossible to cleave peptide substrates, thus preventing the virus to replicate. To maximize the effectiveness of the inhibitors, a deep understanding of the amino acid composition and bond types is required. Unfortunately, laboratory-based methods for the study of substrates are time-consuming and labour-intensive. For this reason, predicting the protease substrate specificity - i.e. which octapeptides are subject to cleavage by the protease - can help the research to focus on the sequences on which there is less confidence about their cleavage status.

We based our work on the study of Onah and colleagues [4], in which several machine learning models were implemented to address the problem of the HIV-1 cleavage site prediction. We implemented ourselves some of those methods, trying to replicate the results obtained in [4].

The code that we developed for this work is available at [5].

2 Data set

The data set is composed of 5848 octapeptides, of which 1001 are cleaved and 4847 are non-cleaved, as shown in Figure 1. Each octapeptide (also called octamer) is composed of 8 amino acids in sequence (e.g. AECFRIFD). In such sequences, the cleavage site is always between the fourth and the fifth amino acids. There are 20 different possible amino acids that can be used to form an octapeptide, so there is a total of $20^8 = 2.56 \times 10^{10}$ possible combinations, but not every combination is present in the polyproteins produced by the HIV genome.

In the data set, each octamer is represented through a 178-dimension feature vector. The different features are explained below in details.

Amino acid binary profile (AABP) Each octapeptide sequence is represented using orthogonal encoding, so that each amino acid is represented by a 20-bit vector with 19 bits set to zero and one set to one. For example, alanine (A) is 10,000,000,000,000,000,000. Since each octapeptide consists of 8 amino acids, each octamer is mapped to a 160-dimensional vector.

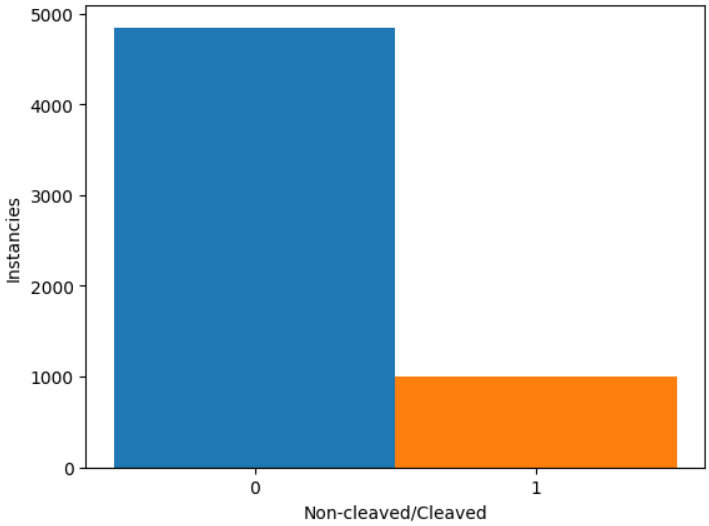

Figure 1: Data set distribution. It can be noticed that the data set is heavily imbalanced, with much more non-cleaved instances than cleaved instances.

Physicochemical properties (*PCP*) They consist of 14 scalar quantities, representing the average values of octapeptide physicochemical properties like polarity, acidity, aromaticity, etc.

Atomic bond compositions (*BTC*) They are 4 scalar values quantifying the prevalence of the four considered bonds (single, double, hydrogen and aromatic) inside the amino acids that form an octapeptide.

Physicochemical properties and atomic bond compositions were obtained using the peptide features extraction algorithms implemented in Pfeature Python package [6, 4].

In order to eliminate redundant features, Ohna and colleagues applied the variance feature selection algorithm of scikit-learn version 1.0.2 [7] on the 14 physicochemical properties and 4 bond compositions feature vectors, finding out that there are no features with the same value in the physicochemical properties and bond compositions [4].

Finally, the target vector is a binary variable, assuming the value of 1 for cleaved sequences and 0 for non-cleaved sequences. Our classification task is to tell whether a given octapeptide will be cleaved or not between the fourth and the fifth position.

The article [4] and the data set [8] we based our work on are freely accessible (Creative Commons Attribution 4.0 International License [9]).

3 Methods

We chose as main method to address this problem a multi-layer perceptron classification. As alternative methods, we implemented a k-NN classifier and a logistic regression classifier.

3.1 Data Preprocessing

First of all, we normalized the design matrix using `MinMaxScaler()`. Since our features have very different ranges and distributions, scaling helps to standardize the range of features and ensures that each feature contributes equally to the analysis. In this way we prevent the supervised learning models from getting biased toward a specific range of values [10]. We chose to normalize the data with `MinMaxScaler()` because it is the preferred method when the features are not distributed according to a Gaussian [10, 11], like the AABP. Furthermore, normalization is a good scaling technique when the features are already bounded, as the AABP and the PCP's are.

Then we dealt with the imbalance of the original data set. In order to avoid having a biased test set, not representative of the actual data set - which would lead to an optimistic estimate of model performance - we performed the train-test splitting before balancing the whole data set [12]. Balancing the training set is recommended because it helps prevent the model from becoming biased towards the most populated class. For balancing, we performed a random undersampling only on the training set, leaving the same number of cleaved and non-cleaved instances. Since the validation and test sets were not undersampled, in principle we assigned to them a smaller fraction than the usual 10% for validation and 20% for test. Indeed, in train-test splitting we assigned the 85% to the training set, the 5% to validation set and the 10% to the test set, stratifying on the target variable in order to keep the same proportions as the original data set. The number of instances given to each class are shown in Table 1.

It must be pointed out that, differently from our work, in [4] neither feature scaling nor training data balancing were applied.

	Training	Validation	Test
N. of non-cleaved instances	851	240	488
N. of cleaved instances	851	50	100
N. of total instances	1702 (66%)	290 (11%)	588 (23%)

Table 1: Number of instances and proportions for each class.

3.2 Multi-Layer Perceptron Classifier

At the very beginning, we tried to build a naive multi-layer perceptron (MLP), manually setting the hyperparameters to explore which combinations could work in principle. As we expected, because of our rather small data set, we found out that even shallow and small networks returned good results in terms of loss.

After this first rough experiment, we used the Gridsearch method to find the best combination of hyperparameters. We used `binary-crossentropy` as loss function, `adam` as optimizer and `accuracy` as metrics. We used `relu` as activation function for all the layers but the output one, for which we used `sigmoid`. The number of epochs was set to 30 and the batch size to 64, so as to speed up the training process.

Then we ran the Gridsearch over the combinations of hyperparameters shown in Table 2. As a result of the Gridsearch we obtained the dependencies shown in Figure 2. All the trends are flat, so there are no strong dependencies on any hyperparameter.

Dense nodes	[8] [16] [32] [64] [128] [8,4] [16,8] [32,16] [64,32]
Learning rate	[0.001] [0.0001]
Dropout	[0] [0.2] [0.4]
L2 lambda	[0] [0.001]

Table 2: Combinations of hyperparameters used in the Gridsearch. The number of total combinations is 162.

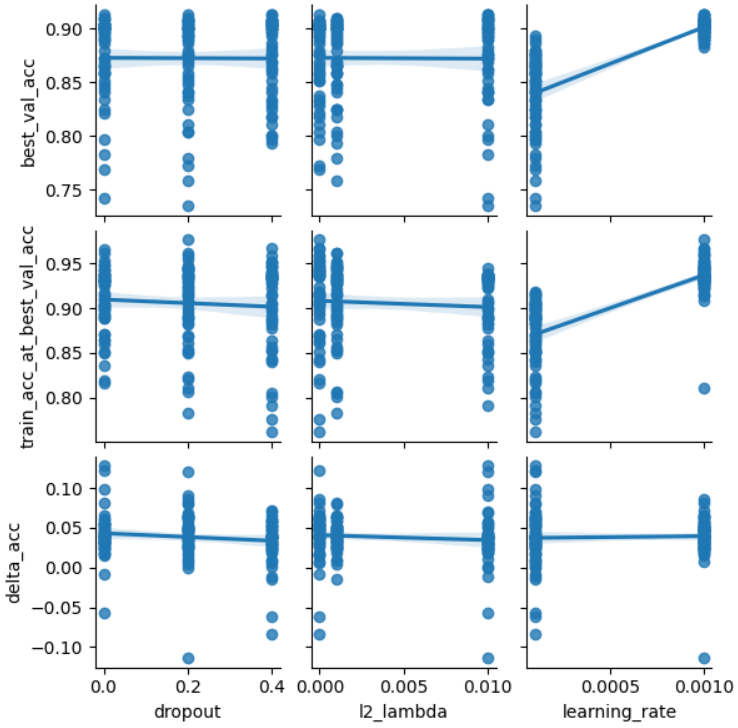

Figure 2: Dependencies on the hyperparameters of the best validation accuracy, of the training accuracy reached in the epoch with the best validation accuracy, and of their difference.

In Figure 3 we plotted loss and accuracy of the models from Gridsearch that showed the three best validation accuracies. We don't observe overtraining, but it can be noticed that the blue loss, relative to a MLP with one hidden layer of 64 nodes and no dropout oscillates more than the other two. This is probably due to the too big size of the network with respect to the data set, and also to the dropout rate set to zero.

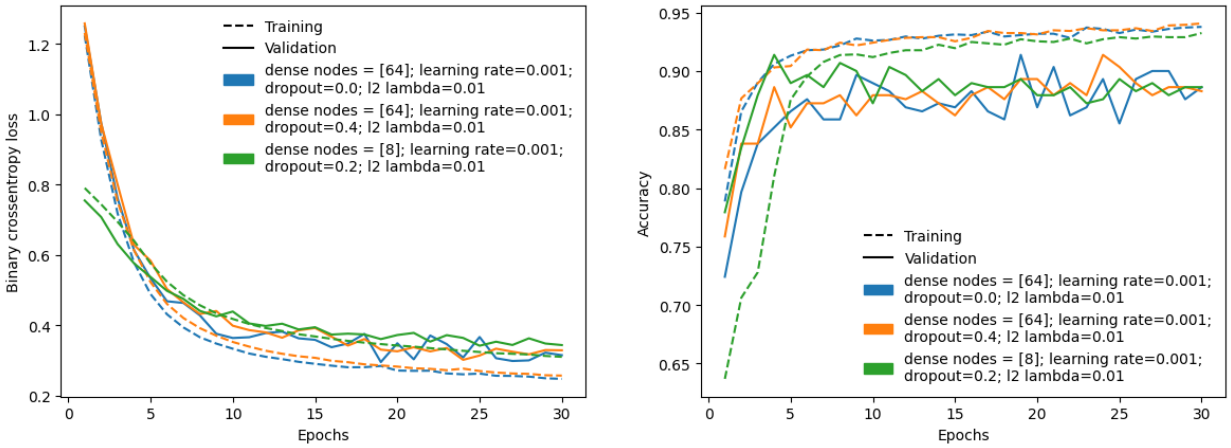

Figure 3: Loss and accuracy of the models getting the highest validation accuracies in Gridsearch.

We manually implemented some of the best models that came out from the Gridsearch analysis. The tested hyperparameter combinations are shown in Table 3. All of them showed good performances, which indeed are very similar for all the models, but the best one - according to the performance evaluation described in section 4 - is the model 3, whose architecture is shown in Figure 4. Model 3 loss and accuracy are shown in Figure 5.

	Dense nodes	Learning rate	Dropout	L2 lambda	Batch size	N. of epochs
Model 1	[16, 8]	0.01	0.4	0.01	32	50
Model 2	[64]	0.001	0.4	0.01	64	40
Model 3	[64]	0.001	0	0.01	64	40
Model 4	[64, 32]	0.001	0.4	0.01	64	40
Model 5	[8]	0.001	0.2	0.01	32	60

Table 3: Hyperparameter sets of the tested multi-layer perceptrons.

The Gridsearch method succeeds at identifying the best hyperparameter combinations, even if they still have to be tested one by one. Indeed, Gridsearch can help a lot with the minimization problem, but it is based on finding the best validation accuracy, which in our case is less important than other performance metrics (see section 4).

3.3 *k*-NN and Logistic Regression Classifiers

For the two alternatives methods, we used the same balanced training set as for the MLP, but we merged the validation and test to form a larger imbalanced test set.

In the *k*-NN classifier, we set the the number *k* of nearest neighbours to 4, as they do in [4].

For the logistic regression classifier we also used the same parameters as in [4]: therefore, we set `solver='lbfgs'`, `multiclass='ovr'`, `penalty='l2'` and `n_jobs=-1`.

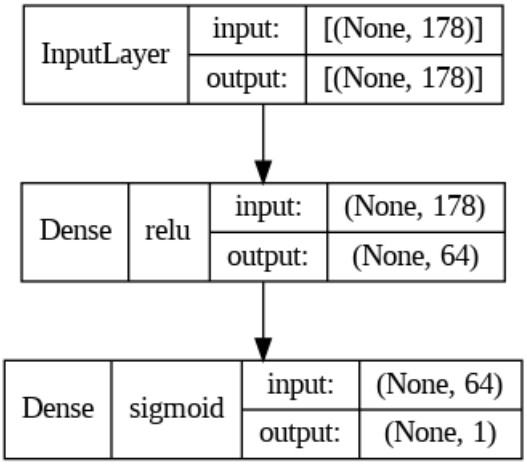

Figure 4: Best performing network architecture, with `binary-crossentropy` as loss function, `adam` as optimizer, learning rate of 0.001, dropout rate of 0, batch size of 64, L2 lambda of 0.01, trained for 40 epochs.

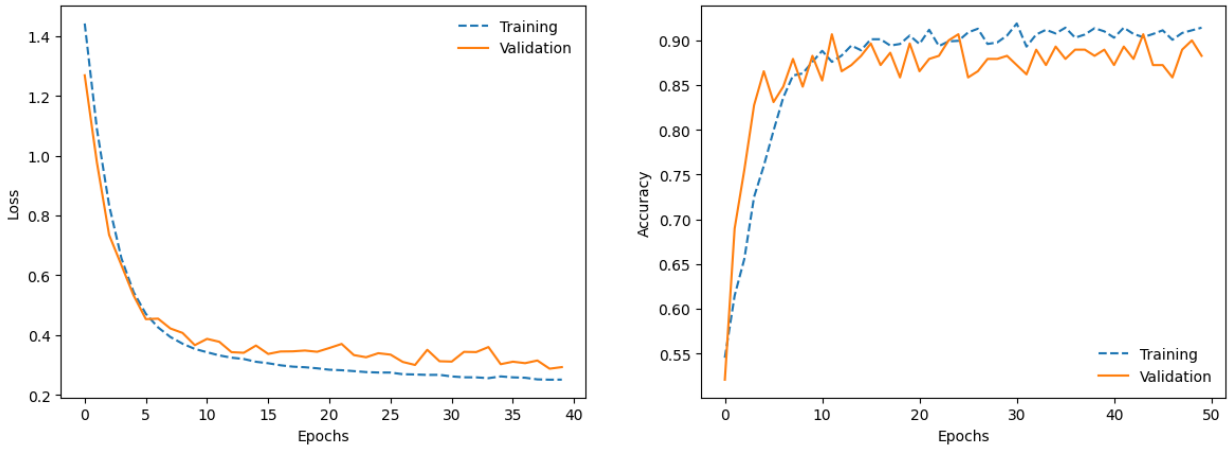

Figure 5: Loss and accuracy of the network described in Figure 4.

4 Results

The following five performance metrics have been identified by the literature to be significant for binary classification tasks [13, 14, 4]. We define the correct positive predictions as *TP*, the correct negative predictions as *TN*, the wrong positive predictions as *FP* and the wrong negative predictions as *FN*.

Accuracy It is a standard metrics for quantifying the overall performance of a classifier. It is defined as $Acc. = \frac{TP+TN}{TP+FP+FN+TN}$.

Sensitivity It is defined as $Sen. = \frac{TP}{TP+FN}$, so it is the true positive rate.

Specificity It is defined as $Spe. = \frac{TN}{TN+FP}$, so it is the true negative rate.

AUC-ROC The ROC curve is the plot of the true positive rate against the false positive rate. The AUC-ROC is the area under the ROC. It is a good performance metrics if the goal of the model is to perform equally well on both classes [15].

F-score It takes into account both sensitivity and precision, since it is defined as $F-score = 2 \times \frac{Prec. \times Sen.}{Prec. + Sen.}$.

Sensitivity and specificity are equally important in our case, so we cannot give more importance to one with respect to the other. Indeed, a good sensitivity is important to detect sequences that may be cleaved, which is our main goal. On the other hand, a high specificity is also important, otherwise the model return as cleaved a lot of sequences that are actually non-cleaved. Since the actual cleavage status must be verified with time-consuming laboratory-based methods, and considering that non-cleaved sequences are more common than the cleaved ones, it is important to be confident that the classifier doesn't return too many false positives, otherwise the classifier would be useless.

In Table 4 are shown the values of the mentioned performance metrics achieved by the selected classifiers. We used as overall performance metrics a simple average between all the other metrics. According to this average, the MLP classifier is the one that performs better, even if its performance is basically the same achieved by the LR classifier. As found also in [4], the *k*-NN classifier is the one that performs worse. Differently from us, in [4] the LR classifier achieved a better performance than the MLP classifier. Anyway, despite of the different MLP architecture and the different preprocessing of the data, our results are very close to the ones obtained in [4].

The confusion matrices for the classifiers are shown in Figure 6, while their ROC curves are plotted in Figure 7.

	Accuracy	Sensitivity	Specificity	AUC-ROC	F-score	Performace avg.
MLP	0.93	0.90	0.93	0.96	0.93	**0.93**
k-NN	0.83	0.85	0.83	0.90	0.85	**0.85**
LR	0.91	0.92	0.90	0.96	0.91	**0.92**

	Accuracy	Sensitivity	Specificity	AUC-ROC	F-score
MLP	0.92	0.89	0.95	0.97	0.90
k-NN	0.69	0.47	0.91	0.78	0.69
LR	0.91	0.94	0.97	0.97	0.91

Table 4: Performances of the multi-layer perceptron, *k*-NN and logistic regression classifiers implemented by us (top) and implemented in [4] (bottom).

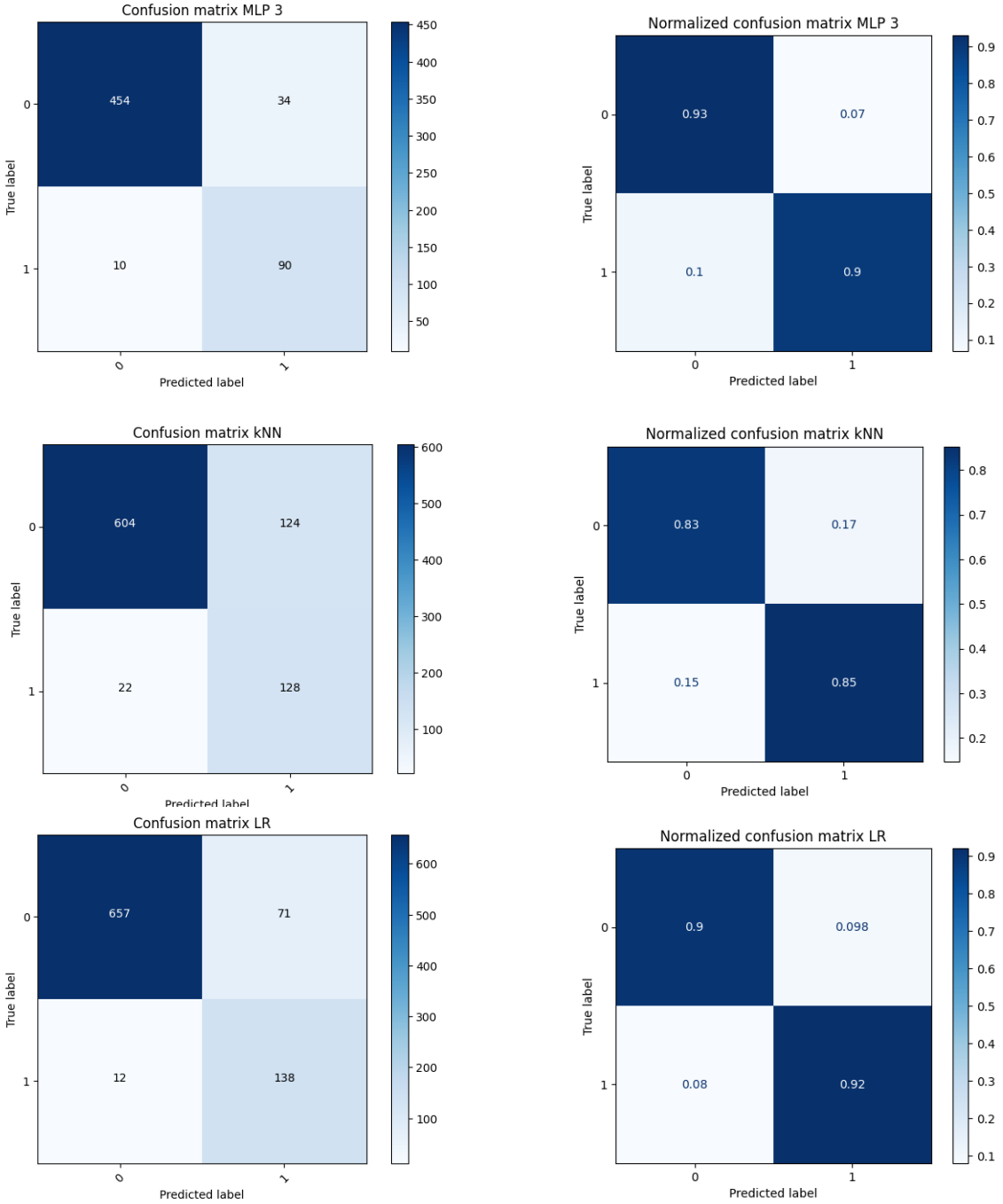

Figure 6: Confusion matrices for the MLP classifier (first row), for the *k*-NN classifier (second row) and for the LR classifier (third row).

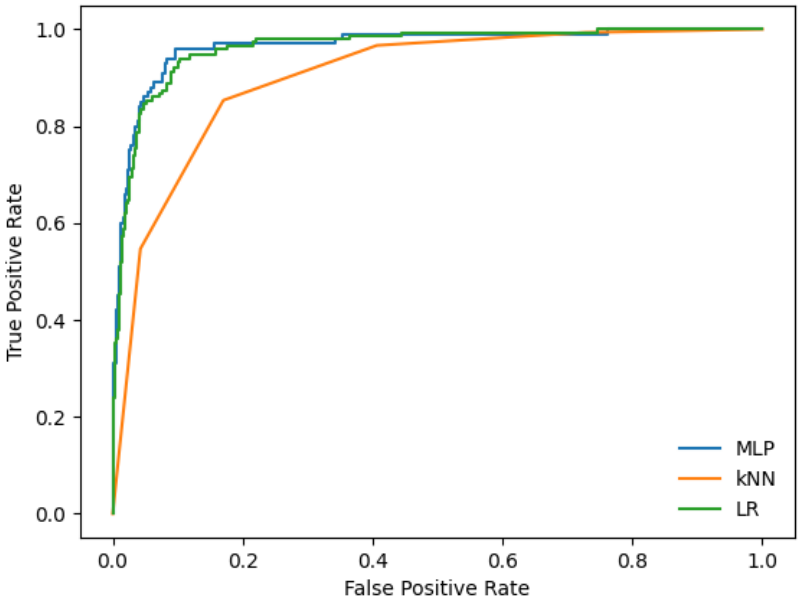

Figure 7: Comparison of the ROC curves for the three classifiers.

5 Summary

Our results confirm the feasibility of predicting the HIV-1 cleavage sites by using supervised learning. We succeeded in the purpose of implementing supervised algorithms that achieve performances comparable to what was previously done in [4]. We found two models that behave equally well in dealing with the HIV-1 cleavage prediction problem: the multi-layer perceptron and the logistic regression classifiers.

The performance could probably be further improved by increasing the data set, also with data augmentation techniques.

References

[1] *HIV*, (2023) https://en.wikipedia.org/wiki/HIV.

[2] *Virus*, (2023) https://en.wikipedia.org/wiki/Virus.

[3] *HIV-1 protease*, (2023) https://en.wikipedia.org/wiki/HIV-1_protease.

[4] E. Onah, P. F. Uzor, I. C. Ugwoke, et al., "Prediction of HIV-1 protease cleavage site from octapeptide sequence information using selected classifiers and hybrid descriptors", BMC Bioinformatics **23**, 10.1186/s12859-022-05017-x (2022).

[5] *HIV-cleavage-prediction repository on GitHub*, https://github.com/sim1-99/HIV-cleavage-prediction.

[6] A. Pande, S. Patiyal, A. Lathwal, et al., "Computing wide range of protein/peptide features from their sequence and structure", bioRxiv, 10.1101/599126 (2019).

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research **12**, 10.48550/arXiv.1201.0490 (2011).

[8] *Data set: 178D octapeptide sequence descriptors calculated from Pfeature software*, (2022) https://static-content.springer.com/esm/art%3A10.1186%2Fs12859-022-05017-x/MediaObjects/12859_2022_5017_MOESM2_ESM.xlsx.

[9] *License: Attribution 4.0 International (CC BY 4.0)*, http://creativecommons.org/licenses/by/4.0/.

[10] *MinMaxScaler vs StandardScaler – Python Examples*, https://vitalflux.com/minmaxscaler-standardscaler-python-examples/#Why_is_Feature_Scaling_needed.

[11] *The Ultimate and Practical Guide on Feature Scaling*, (2021) https://jeande.medium.com/the-ultimate-and-practical-guide-on-feature-scaling-d03fbe2cb25e.

[12] *Having an Imbalanced Dataset? Here Is How You Can Fix It.* (2019) https://towards-datascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb.

[13] M. Sokolova, N. Japkowicz, and S. Stan, "Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation", Advances in Artificial Intelligence **4304**, 1015–1021 (2006).

[14] L. S. D. Mosley, "A balanced approach to the multi-class imbalance problem", PhD thesis (Iowa State University, 2013).

[15] *Precision - Recall Curve, a Different View of Imbalanced Classifiers*, (2020) https://sinyi-chou.github.io/classification-pr-curve/.