HIV-1 Cleavage Prediction

Machine Learning for Physicists
Technische Universität Dortmund

Martina Turchini

July 31, 2023

Contents

1 Introduction

The human immunodeficiency viruses (HIV) are a type of retrovirus that infect the immune system of the human body. They belong to the Lentiviruses, due to their long incubation time, which can be years long after they entered the body. Two types of HIV have been characterized: HIV-1 and HIV-2. The former is deadlier and most diffuse worldwide [1]. Overtime, HIV causes acquired immunodeficiency syndrome (AIDS), a condition in which progressive failure of the immune system allows life-threatening opportunistic infections and cancers to thrive[2]. AIDS is a global problem, resulting in 25 million death all over the world, in particular in developing countries because of the lack of opportunities to access to health care and information about the disease[1].

The infective particle of HIV is composed of two copies of positive-sense single-stranded RNA.

Generally speaking, when a virus enters a cell, it replicates its genome, which involves the synthesis of viral mRNA and proteins [3]. One of these proteins is HIV-1 protease, an enzyme involved with peptide bond hydrolysis, that is essential for the life-cycle of HIV [4]. 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 [5].

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 [5], in which several machine learning models have been implemented to address the problem of the HIV-1 cleavage site prediction. We implemented some of those methods, trying to replicate the results obtained in [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. In every octapeptide, the cleavage site is always located between the fourth and the fifth amino acids. Each octapeptide (in the following also called octamer) is composed of 8 amino acids in sequence (e.g. AECFRIFD). 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.

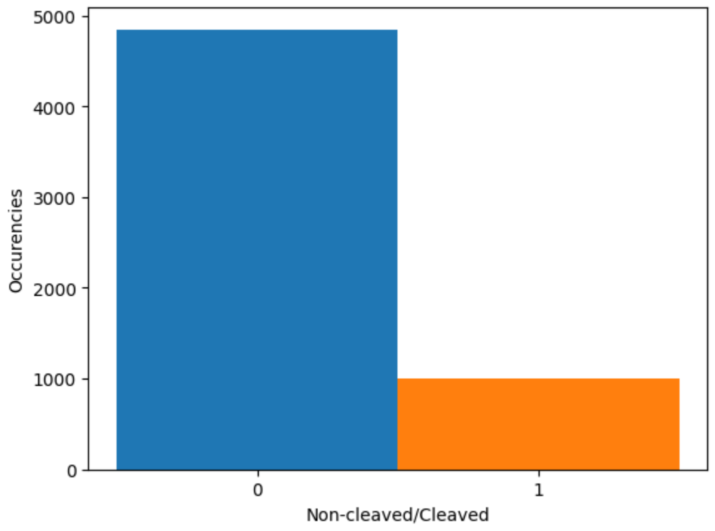


Figure 1: *Data set distribution.* It can be noticed that the data set is heavily imbalanced, with a clear predominance in the number of non-cleaved entries (0) than cleaved entries (1).

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.

Physicochemical properties (*PCP*) They consist of 14 scalar values, representing the average value, along the octapeptide, of 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 a sequence.

Physicochemical properties and atomic bond compositions were obtained using the peptide features extraction algorithms implemented in Pfeature Python package [6, 5].

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 [5].

Finally, the target vector is a binary variable, assuming the value of 1 for cleaved sequences and 0 for non-cleaved sequences.

The article [5] and the data set [8] we based our work on are freely accessible (Creative Commons Attribution 4.0 International License [9]).

The code we implemented is also freely accessible on GitHub [10].

3 Methods

The goal of the work was a classification task, namely to tell whether a given octamer will be cleaved or not between the fourth and the fifth position.

For this goal to be achieved, three different machine learning methods were implemented, in order to compare the results between each other. Multi-Layer Perceptron (MLP) classifier was used as the main method, while k-Nearest Neighbour (k-NN) and Logistic Regression (LR) classifiers were implemented as alternative methods. We chose the listed method because they are known to perform well on binary classification problems.

3.1 Data Pre-processing

First of all, we had to normalize the model matrix using `MinMaxScaler()` from `Sklearn`. Since our features are measured at very different scales, they do not contribute equally to the model fitting and they might end up creating a bias towards a specific range. We scaled the features with `MinMaxScaler()` because it works best when the data distribution is not gaussian [11], like AABP.

As clearly visible in Figure 1, the data set is unbalanced: the non-cleaved octamers far outweigh the cleaved ones. Balancing the dataset is important to prevent the model from being biased towards one class. Therefore, we performed an random under-sampling method on the training set which allowed us to remove some observations from the majority class, in order to match the number with the minority class [12]. The data set was previously divided in 3 subsets: validation, testing and training to prevent the model from over-fitting and to evaluate it effectively. The number of instances in each class are listed in Table 1.

	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 (MLP)

The main method we used for the binary classification problem was the MLP classifier. The hyperparameters define the model architecture and there are many different options to chose between.

The hyperparameter tuning is defined as the process of searching for the best values in the possible hyperparameter space to built the optimal model [13].

GridSearchCV method, from the Scikit-learn's model_selection package, was used in this work to perform the hyperparameter tuning and determine the optimal values for the model. After choosing predefined values for the hyperparametes, GridSearchCV function tries all the possible combinations and evaluates the model for each one of them using the Cross-Validation method.

The model we implemented had `binary-crossentropy` as loss function, `adam` as optimizer and `accuracy` as metric. The number of epochs was 30, while the batch size was set to 64. All layers had `relu` as activation function, except for the output one, for which we choose `sigmoid`.

The hyperparameter values we considered for the tuning are the following: dropout, lambda (λ) for L2 regularizer, number of dense nodes and learning rate. The values we considered for each type are summarized in Table 2.

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 λ	[0] [0.001]

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

From the GridSearchCV we obtained the dependency of the best validation accuracy, of the corresponding training accuracy reached in the same epoch and their difference compared to the different hyperparameter values. Those dependencies are visible in Figure 2.

In Figure 3 the accuracy and loss of the three combinations with the highest validation accuracy obtained with GridsearchCV are plotted. We don't observe overtraining, but some oscillations are visible in the accuracy plot in Figure 3b for the validation of all the three combinations.

From the best results, we manually build a few models. The architecture of one of the best networks is shown in Figure 4, with `binary-crossentropy` as loss function, `adam` as optimizer, learning rate of 0.001, number of epochs equal to 40, batch size of 64, L2 lambda of 0.01. Accuracy and loss for this model are visible in Figure 5.

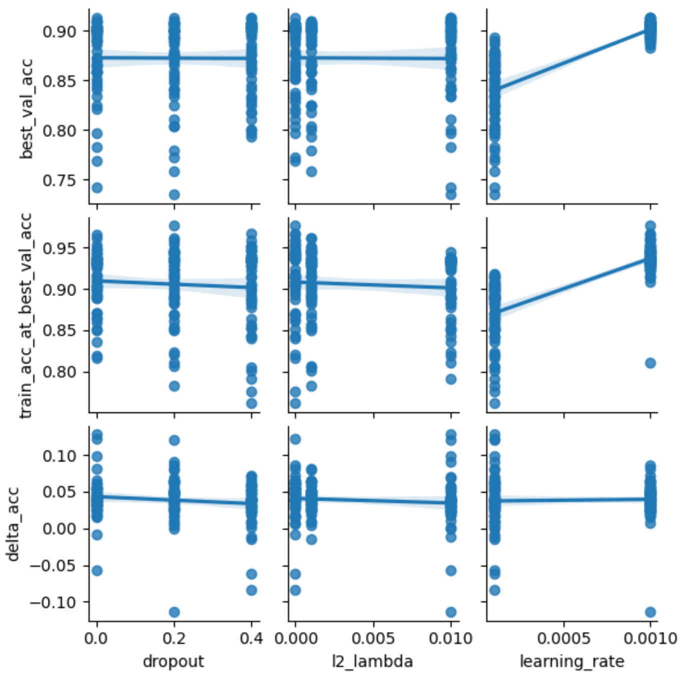
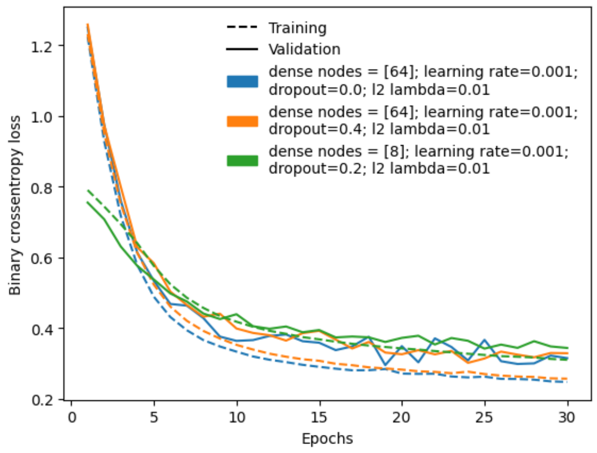
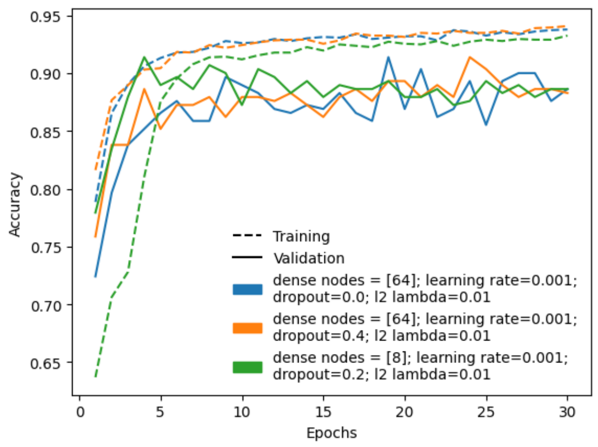


Figure 2: Dependencies on the hyperparameters obtained with GridsearchCV.



(a) *Loss plot*



(b) *Accuracy plot*

Figure 3: Comparison of the three best results for hyperparameter tuning.

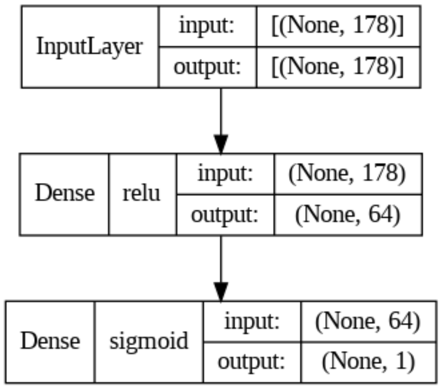
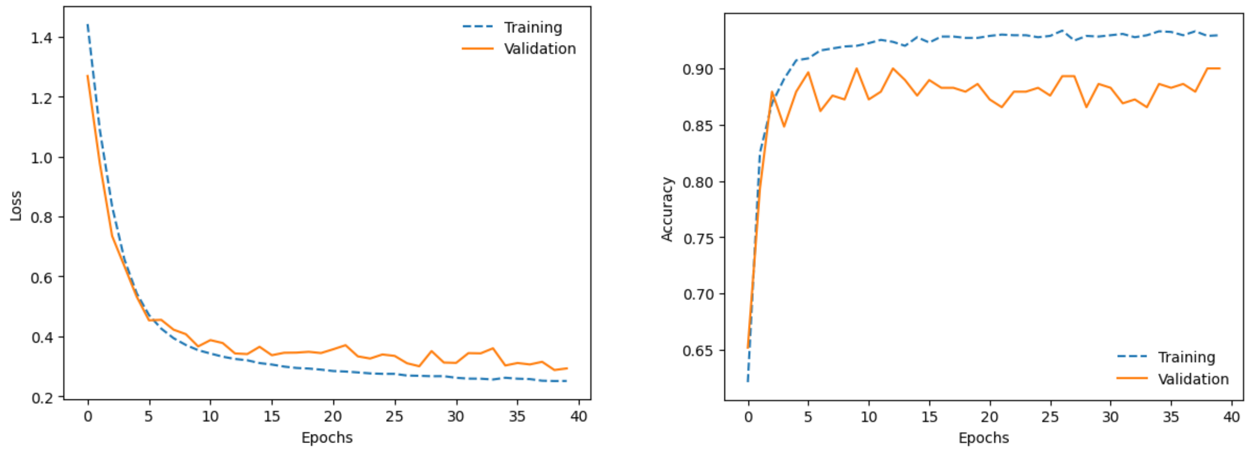


Figure 4: Network architecture of one of the best performing models.



(a) *Loss plot* **(b)** *Accuracy plot*

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

3.3 Alternative Methods

We implemented other two methods to compare the results obtained with the MLP one. These two are the k-Nearest Neighbours (k-NN) and the Logistic Regression (LR).

3.3.1 *k*-Nearest Neighbours (*k*-NN)

In k-NN classification, the output is a member of a class. An object is classified as belonging to the class most common among its k nearest neighbors, where k is a positive integer, typically small [14]. In our project, k = 4 was the one with the best results, in agreement with [5].

3.3.2 Logistic Regression (LR)

Logistic Regression is a classification technique which uses a logistic function to model the dependent variable [15]. For the LR classifier we used the same parameters as in [5]: therefore, we set `solver='lbfgs'`, `multiclass='ovr'`, `penalty='l2'` and `n_jobs=-1`.

4 Results

In supervised machine learning there are several ways to evaluate the performance of a classifier. We define the correct positive predictions as *TP* (true positive), the correct negative predictions as *TN* (true negative), the wrong positive predictions as *FP* (false positive) and the wrong negative predictions as *FN* (false negative). The confusion matrix for binary classification is built from these parameters [16] and it gives an overview of the performance of the model. In Figure 6 the Confusion Matrices with the corresponding Normalized Confusion Matrix for the three classifiers are shown. Each confusion matrix was computed from the testing set using the Sci-kit learn confusion matrix display tool.

The following five performance metrics have been identified by the literature to be significant for binary classification tasks [16, 5].

Accuracy It is a standard metric 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 $Sen. = \frac{TP}{TP+FN}$ and corresponds to the true positive rate.

Specificity It is defined as $Spe. = \frac{TN}{TN+FP}$ and corresponds to 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 defined as the area under the ROC. It is a good performance metric if the goal of the model is to perform equally well on both classes [17].

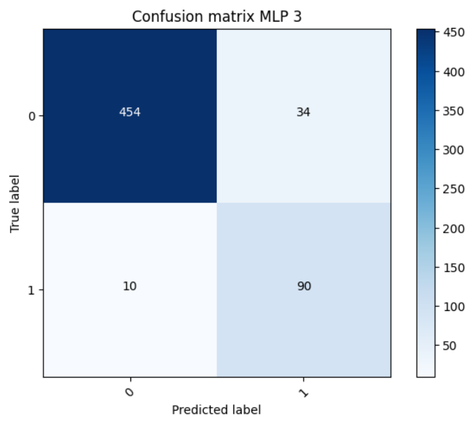
F-score It takes into account both sensitivity and precision: $F{-}score = 2 \times \frac{Prec. \times Sen.}{Prec. + Sen.}$.

Table 3 shows the performance metrics achieved by the three classifiers. The best results are obtained using the Multi-Layer Perceptron classifier. We then compared the results with the ones presented in Table 3 in [5]. For the MLP classifier, we obtained slightly better values for accuracy, sensitivity and F-score, while specificity and AUC_ROC are respectively 1% and 2% worst.

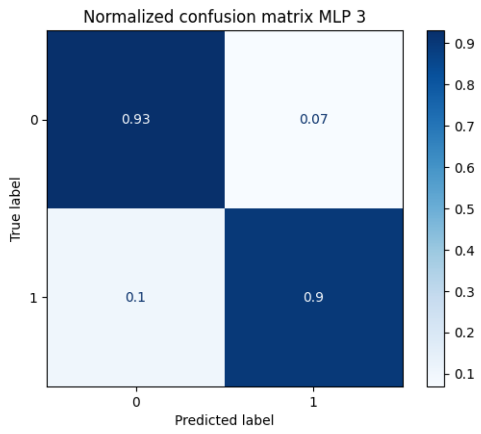
The performance metrics achieved by the *k*-NN classifier are definitely better than the ones presented in [5]. In our case, the F-score and the accuracy are about 15% higher, only the specificity is 8% worst.

For what concern the LR classifier, the results are very similar, except fot the higher sensitivity and lower specificity in this work (about 7%).

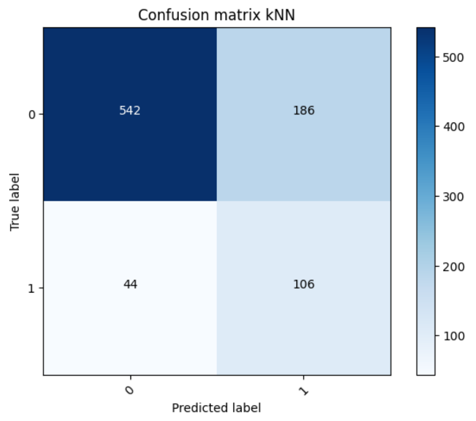
In Figure 7 the ROC curves for the three classifiers are compared.



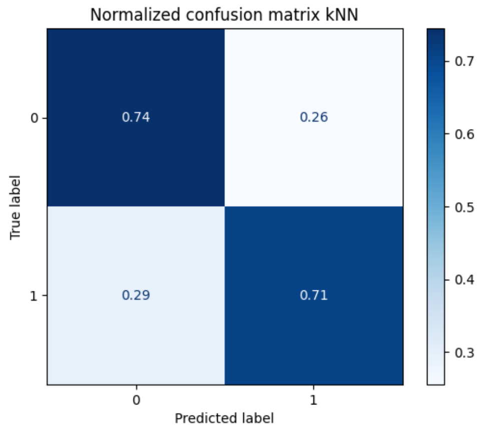
(a) *Confusion Matrix MLP*



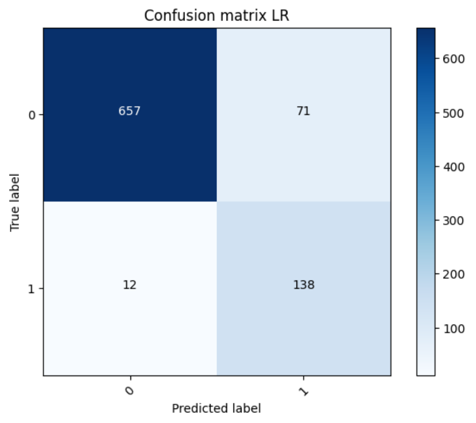
(b) *Normalized Confusion Matrix MLP*



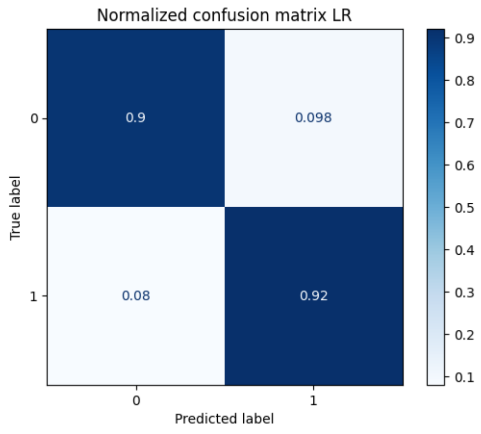
(c) *Confusion Matrix k-NN*



(d) *Normalized Confusion Matrix k-NN*



(e) *Confusion Matrix LR*



(f) *Normalized Confusion Matrix LR*

Figure 6: Confusion Matrices for the MLP classifier (**a**) - (**b**), for the *k*-NN classifier (**c**) - (**d**) and for the LR classifier (**e**) - (**f**).

	Accuracy	Sensitivity	Specificity	AUC-ROC	F-score	Performace avg.
MLP	0.93	0.9	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**

Table 3: Performances of the MLP, *k*-NN and LR classifiers.

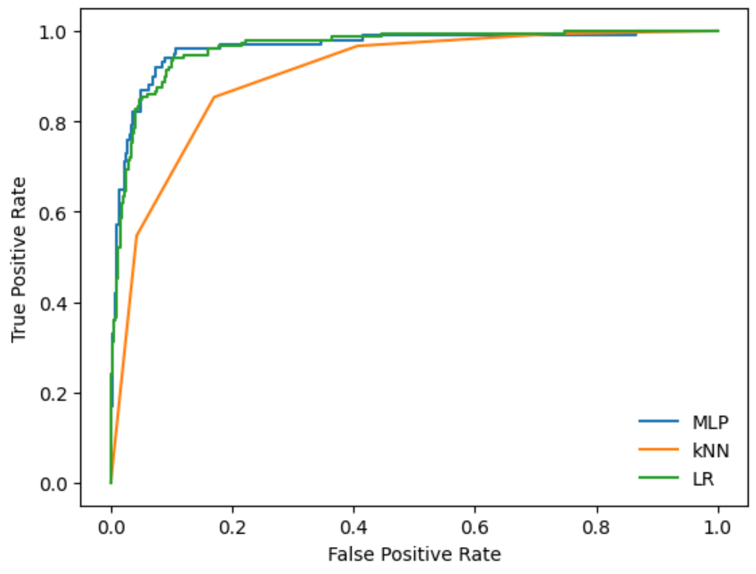


Figure 7: Comparison of the ROC curves for the three classifiers.

5 Summary

This work shows the possibility to study and predict the HIV-1 cleavage sites given the combination of sequence information, including amino acid binary profiles, bond composition, and physicochemical properties. The best performances are obtained via the Multi-Layer Perceptron classifier and similar results can be obtained with the Logistic Regression classifier. The *k*-NN classifier is the worst among the methods analyzed in this work.

The results could be improved by using a larger data set, that can be implemented via data augmentation techniques.

References

[1] M. Z. Yousaf, S. Zia, M. E. Babar, and U. A. Ashfaq, "The epidemic of HIV/AIDS in developing countries; the current scenario in Pakistan", Virology Journal **8**, 10.1186/1743-422X-8-401 (2011).

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

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

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

[5] 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).

[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*, 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] S. Chiarella and M. Turchini, *Code repository for the project*, (2023) https://github.com/martinaturchini/HIV-1-cleavage-project.git.

[11] *Minmaxscaler vs standardscaler – python examples*, (2023) https://vitalflux.com/minmaxscaler-standardscaler-python-examples/#Why_is_Feature_Scaling_needed.

[12] *Having an imbalanced dataset?*, (2019) https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb.

[13] *Hyperparameter tuning for machine learning models*, (2017) https://www.jeremyjordan.me/hyperparameter-tuning/.

[14] *K-nearest neighbors algorithm*, (2023) https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[15] *Perfect recipe for classification using logistic regression*, (2020) https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592.

[16] 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).

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