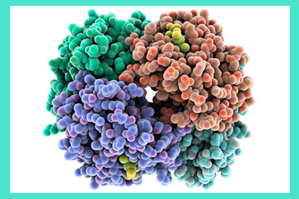
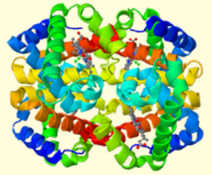
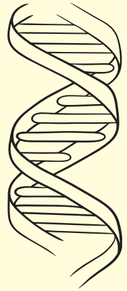


Project 6 : Protein Classification



Adam Boumessaoud, Antoine Grislain, Océane Li, Thomas Louvet

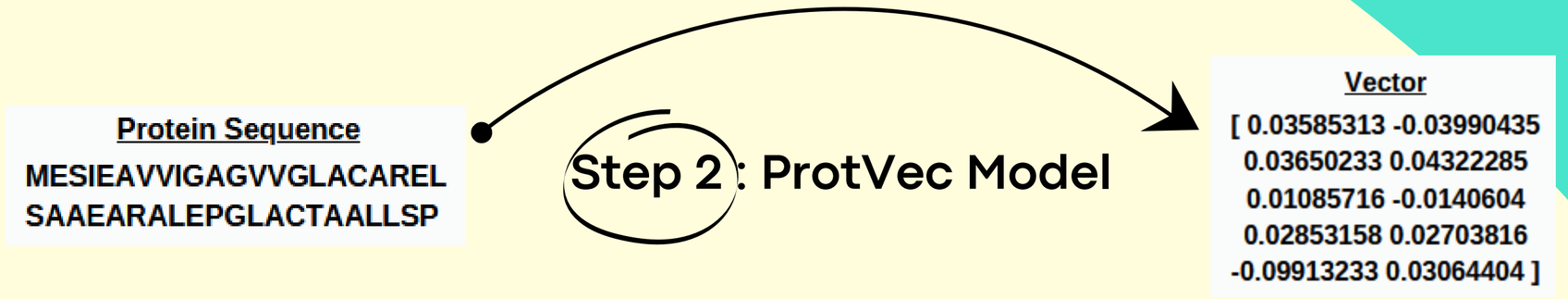
Proteins fulfill a multitude of functions within organisms, with their utility primarily encoded in specific fragments known as active sites. These segments are typically conserved within the protein sequence and have often been identified and classified based on the **InterPro annotation system**, also known as domains (characterized by codes such as IPR000000). The aim of this project is to predict protein function using Deep Learning and Machine Learning techniques based on their sequences.



Step 1

Before applying classification models, it is essential to prepare the data for analysis. To achieve this, we retrieved protein sequences from the public database **SwissProt**. InterPro domains provide crucial information on the functional and structural characteristics of proteins, facilitating their classification and analysis across various biological contexts.

We filtered the proteins to retain only those belonging to the top 100 most frequently observed codes within the entire SwissProt dataset. We also removed all of the InterPro codes pertaining to anything other than a domain and we decided not to take into account the sequences that did not possess any code.



The concept of ProtVec involves representing each protein sequence as real-valued vectors in a multidimensional space. For instance, manipulating the **vectors of 3-grams** (sequences of 3 amino acids) allows the inference of similarities or functional properties between different proteins. To embed these 3-gram vector representations, a Skip-gram neural network is utilized. The objective function to maximize is as follows :

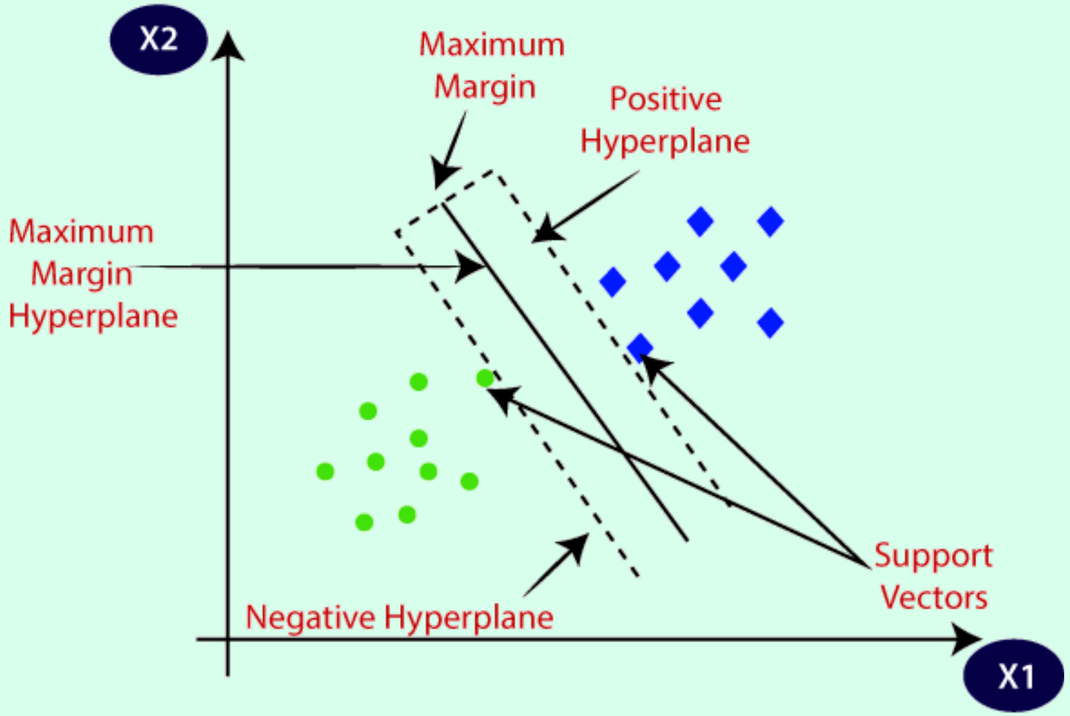
$$argmax\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{-c\leq j\leq c,\, j\neq 0} log\, p(w_{i+j}|w_i)\right),\ \text{with}\ \ p(w_{i+j}|w_i) = \frac{exp(v'^{T}_{w_{i+j}} v_{w_i})}{\sum_{k=1}^{W} exp(v'^{T}_{w_k} v_{w_i})}$$

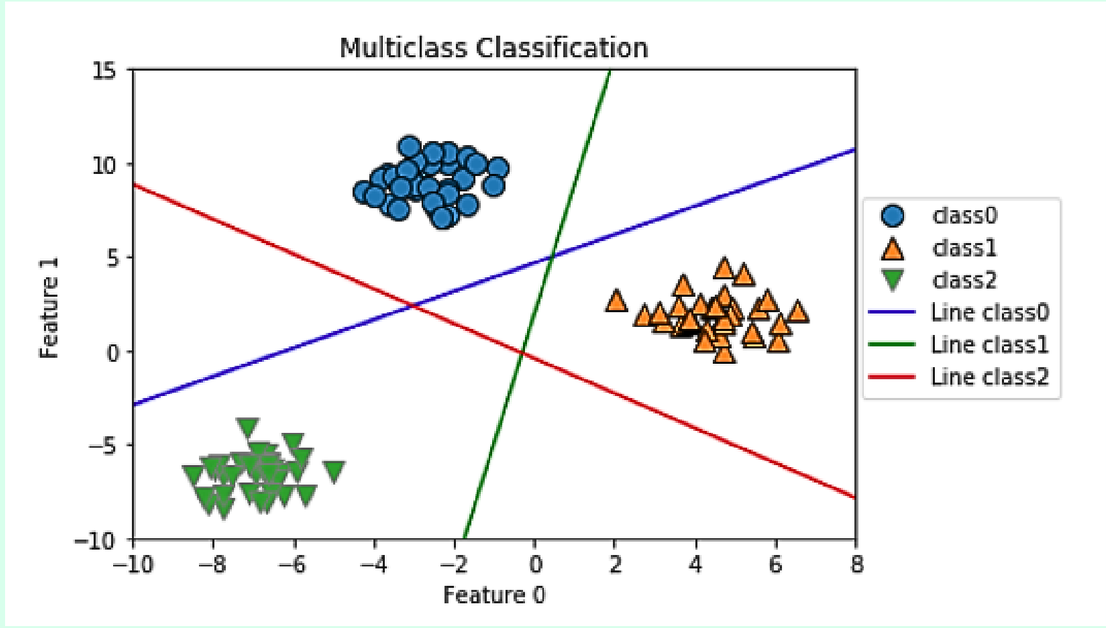
The softmax function converts the scores (dot product between embedding vectors) into probabilities, ensuring that the sum of the probabilities of all 3-grams in the lexicon equals 1. The dot product represents the similarity between the embedding vectors of the context and target 3-grams. Higher similarity results in a higher conditional probability .

Thus, we opted for the **Word2Vec model**, which, once trained, can perform all the functions of a ProtVec model. After this sequence encoding step, our data is pre-processed by ProtVec and ready to be used for training classification models in the next stage.

Step 3: Machine Learning Methods

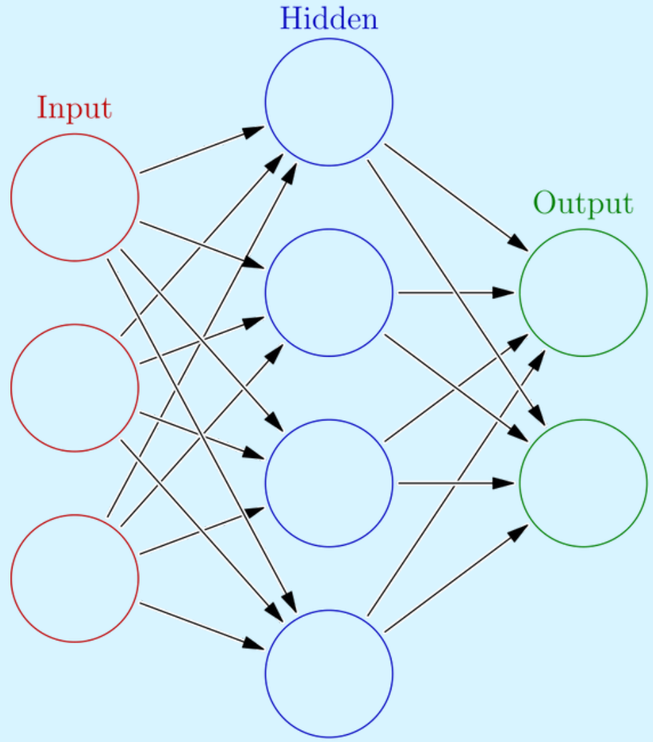
First of all, we attempted a machine learning approach using a **Support Vector Machine** (SVM). Usually, SVMs are used in mono-label cases (only one of n labels is selected). However in our case, quite a few sequences relate to multiple labels at the same time. To circumvent this issue, we used the One Versus Rest strategy : we transformed this complicated exercise into a hundred binary ones. Each InterPro code then gets its own SVM telling us if it is, or not, present in the sequence.





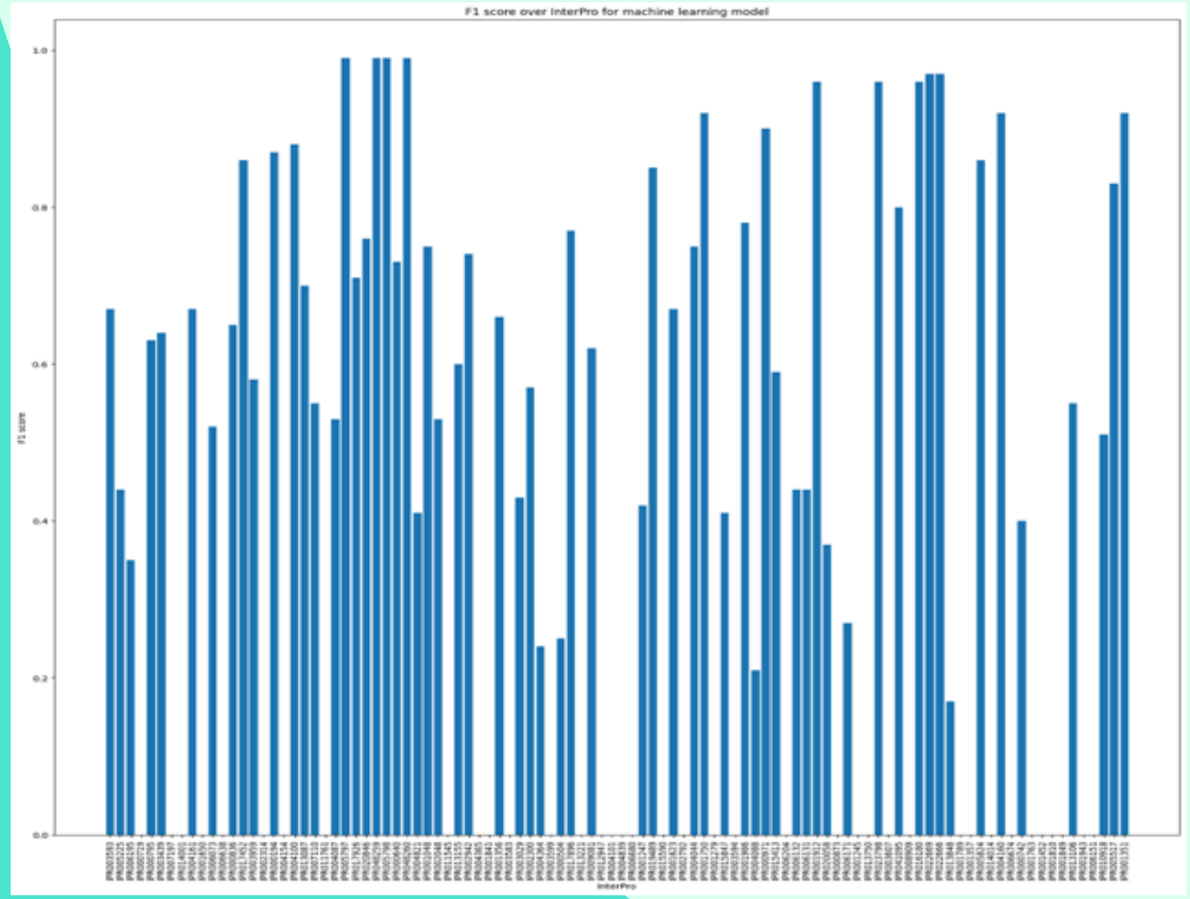
Step 3: Deep Learning Methods

The first step in the process involves preprocessing the data by converting InterPro codes into multi-label indicator vectors, encoding its functional features. The model is structured as a **multi-layer artificial neural network** designed to capture nonlinear relationships among protein sequence features. The first linear layer transforms the input data using weights adjusted during training. To mitigate overfitting, a regularization technique is incorporated through **dropout layers**, where 30% are randomly ignored during training. Between each linear layer, Rectified Linear Unit (**ReLU**) activations introduce non-linearity into the network, enabling the model to learn complex patterns.



Then, we proceed to train it over a defined number of epochs, where each epoch consists of successive iterations through the training data. In each iteration, the model's predictions are compared to the true labels using a loss function (**BCEWithLogitsLoss**). This loss function accounts for the binary distribution of labels and logits generated by the model. Gradients are computed in each iteration and used to adjust the model's weights via the **Adam optimizer**. Once the model is trained, it is evaluated on the independent test set. The model's predictions are compared to the true labels to compute a set of performance metrics such as precision, recall, F1-score, and accuracy.

Step 4: ML Results



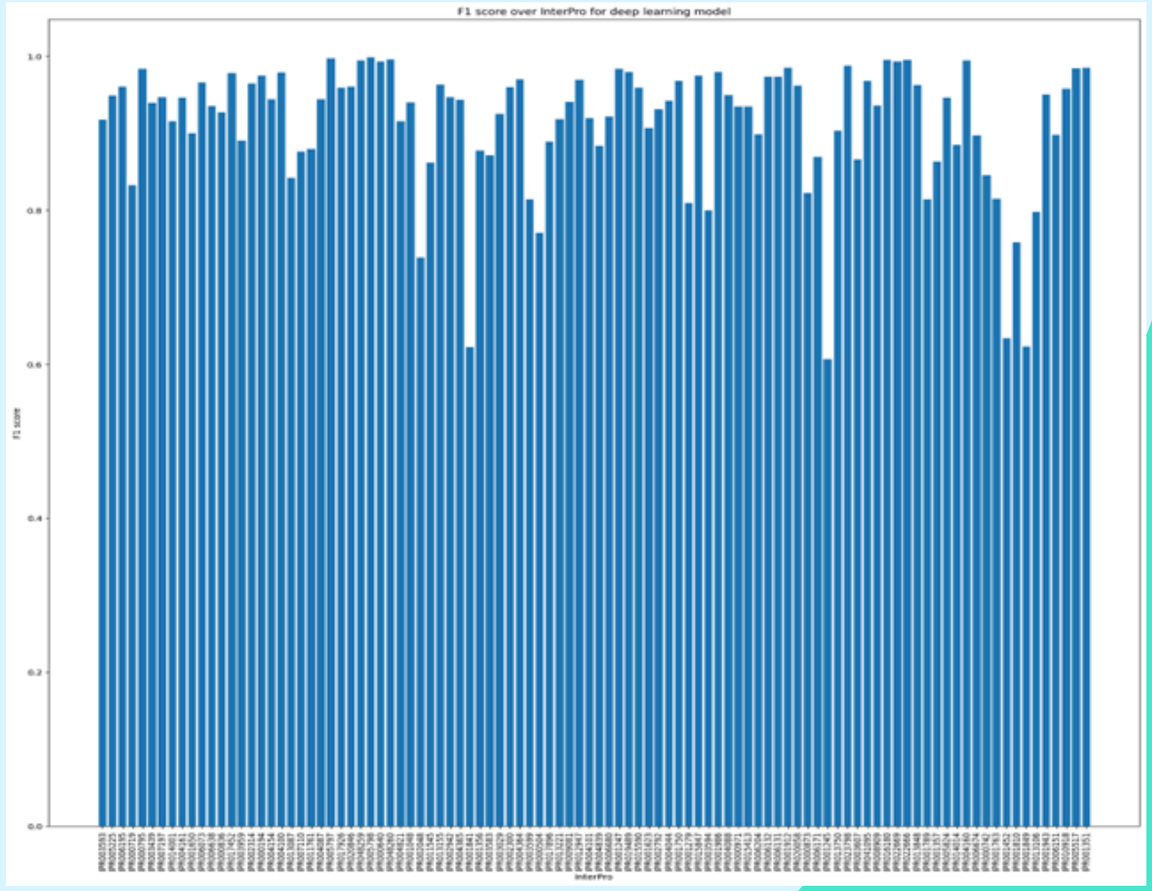
- Accuracy is the most common metric to be used in everyday talk. Accuracy answers the question “Out of all the predictions we made, how many were true?”
- Precision is a metric that gives you the proportion of true positives to the amount of total positives that the model predicts. It answers the question “Out of all the positive predictions we made, how many were true?”
- Recall focuses on how good the model is at finding all the positives. Recall is also called true positive rate and answers the question “Out of all the data points that should be predicted as true, how many did we correctly predict as true?”
- The F1 Score is a measure that combines recall and precision. As we have seen there is a trade-off between precision and recall, F1 can therefore be used to measure how effectively our models make that trade-off. One important feature of the F1 score is that the result is zero if any of the components (precision or recall) fall to zero.

In our case, one of the most interesting statistical values to study is the recall. Sequences rarely relate to more than 5 InterPro codes and, given the large number of possible codes it is much more likely for the model to predict that a sequence does not possess an InterPro code than to predict it does.

Step 4: DL Results

An example is the model predicting that a specific code is never present in our sequences despite being there about 4% of the time. This results in an accuracy of 96%, which sounds amazing at first. However, the recall here would be 0%: the model is incapable of correctly predicting that a code is present when it is. This is why the F1-score is, ironically, a much more precise metric than the precision.

We can see this happening when using the SVM model, as seen in the bar plot. Some of the InterPro codes have an F1-score of 0 for this reason exactly. On the other hand, the Deep Learning model has pin-point accuracy (almost always over 99%) and fairly high F1-scores. The latter are mostly found within the 80% to 99% range.



Conclusion

The deep learning model we created is a lot more consistent than the machine learning model we used. A way to improve it even further would be to first let it predict the less common InterPro codes, as well as when a sequence doesn't have one. Another one would be to find a way to evaluate the possible dependencies between each InterPro code: some might appear more often in the presence of some others. Finding these conditional probabilities might allow the model to make even more accurate predictions.