Classification of contacts in protein structures

Veronica Bedin, Alessandro Canel, Emanuele Pase, and Lorenzo Riccò
(Dated: July 4, 2023)

The aim of the project is to create a software in order to classify contacts inside a protein structure. The software takes a PDB file as input and is able to generate in output a table in which every residue-residue interaction of the protein has a specific propensity of belonging to the different contact types considered, following RING guidelines. We have focused our attention in the development of different Machine Learning methods through two principal paths: on one hand we have tried to use classical supervised methods such as Decision Tree, K-Nearest Neighbor and Logistic Regression tuning and combining them in different ways, on the other we have boost a Multi Layer Perceptron, which has resulted as the best option. The work has been divided, in a balanced way, through each component of the group. Even if it is difficult to define a straight line, the project has been displaced in the succeeding way: Veronica and Lorenzo have focused their attention to the features analysis and to develop the first path followed as well as writing the report, Alessandro has succeed in the MLP deployment , Emanuele has dedicated himself in the assembly of the whole software and in the definition of the documentation. The collaboration between each component of the group and the continuous exchange of ideas has represented the strength of the work.

FEATURE ANALYSIS

The dataset we worked on is characterized by a list of 25 pre-calculated features. The decision of using the pre-calculated freatures was made after considering the possibility of exploring alternative approaches: the other main idea was to enhance the model with as additional features the embeddings produced by ProtTrans. However, due to perceived complexity and potential management difficulties, we did not delve into the matter. We report in a brief the feature considered:

- $ss8$: secondary structure 8 states (DSSP)
- rsa: relative solvent accessibility
- up, $down$: half sphere exposure up, down
- phi, $spsi$: phi and psi angles
- $ss3$: secondary structure 3 states (from angles)
- $a1, ..., a5$: Atchley feature from 1 to 5

These features are present both for the source residue with prefix $s\_$ and target residue with prefix $t\_$.
The target value is the *interaction* feature with which these two residues bind. The values follows: HBOND stands for hydrogen bonds, VDW for Van der Waals interactions, SSBOND for disulfide bridges, IONIC for salt bridges, PIPISTACK for $\pi - \pi$ stacking and finally PICATION for π-cation.

At first, we had a look at the distribution of the interactions in our dataset in 1. The high unbalance suggested that over or under sampling solutions will need to be inspected.

We have plotted also the correlation matrix 2 to have a first look at the properties of our features. There, We can see that some features like t_up and t_rsa, s_up and s_rsa, s_a3 and s_a5, t_a3 and t_a5 are highly correlated.

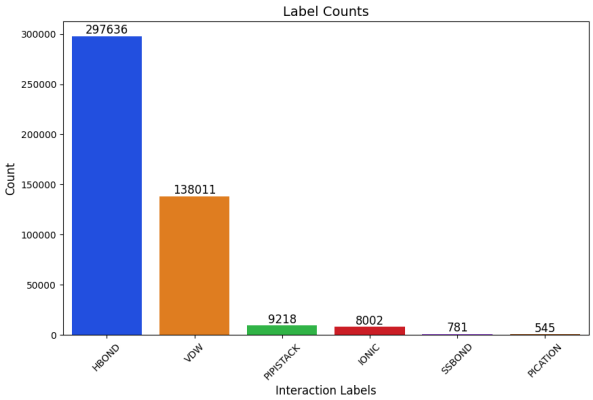

FIG. 1: The number of instances for each one of the possible interactions.

This suggests that they provide similar or redundant information to our model, so we could try to drop one for each couple during training.

After this we plotted the frequencies of the values of each feature for every different interaction. This allowed for a quantitative insight into the importance of the features. All the plots show the source amino acid but the same conclusions are valid for the target one.

The analysis of 3 for the feature s_down shows that the distribution is the same for all the different kind of interactions. Hence, the model will likely not be able to use this feature to predict the interaction.

Even feature s_ss3 shows no differences between the interactions. These considerations suggest to drop s_ss3, t_ss3, s_down, t_down in our training.

In opposition to 3, figure 4 shows difference between interactions in the distribution profile: this seems like a good feature to take into consideration.

The same happens when considering 5: e.g. we can

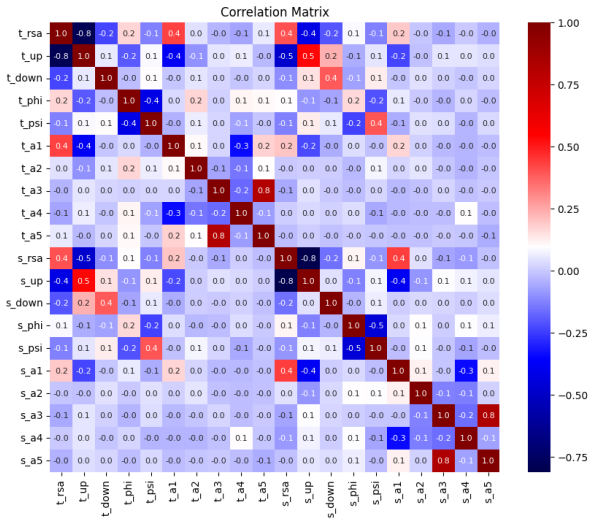

FIG. 2: The correlation matrix of our features. *s_ss*8 and *s_ss*3 are not plotted as they are categorical variables.

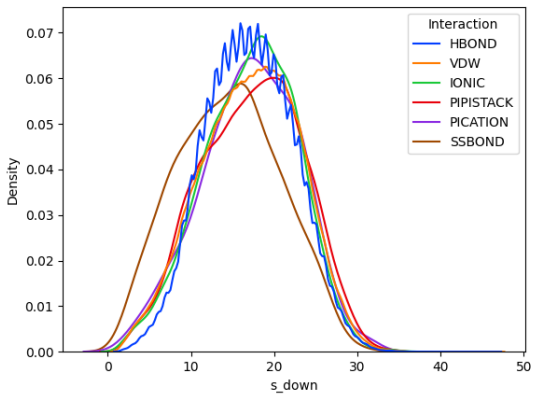

FIG. 3: The frequency of appearance of *s_down* feature. The curves represent the relative distribution of data points for each interaction type.

see that in more than 50% of the hydrogen bonds, the secondary structure is an alpha-helix -labelled with *H*-. The SSBOND interaction deserves special attention: only the cystine amino acid can from this particular bond and that is why it was excluded from the plotting of the Atchley values (e.g. 4).

This finals considerations about the features characteristics have been tested mainly in the develop of the MLP classifier.

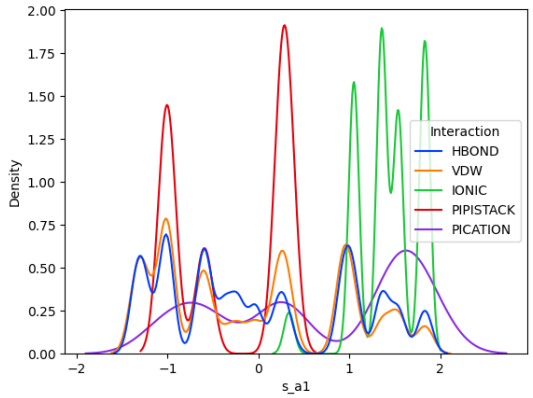

FIG. 4: The frequency of appearance of *s_a*1 feature. The curves represent the relative distribution of data points for each interaction type. The SSBOND interaction is excluded from the plot.

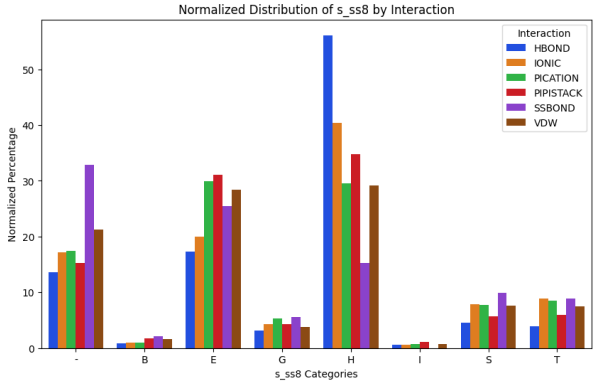

FIG. 5: The probability of appearance of *s_ss*8 feature, normalized for every different interaction.

FIRST PATH: SUPERVISED MACHINE LEARNING METHODS

As mentioned before we have followed two different approaches. In the first one, even without getting in the end reliable results we have understand deeply the characteristics of the problem as well as its main 'braking points'. We have decided to report and argue this part because has taken us a lot of time and thought about and, moreover, has lead us in big improvements in MLP developement phase. Here, we have decided initially to use all the features available.

The approach followed has been to train different methods for different labels (of the target variable *interaction*), using for each one the so called 'one-vs-all (or 'one-vs-rest') technique. The idea has been to decompose a multiclass classification problems into several binary classifications problem for which, every time, the label of interest was settled to 1 and all the others to

-1, grouping the results at the end in a unique model. We have done this effort also in the developing of the Multi Layer Perceptron creating proper Label Encoder in order to deal with the data and the format needed by Pytorch. We have decided to go around with this path following the suggestion of the professor and testing out the fact that in our first trial with the overall set of labels the results were very poor in terms of balanced accuracy, F1 score and MCC coefficient, which have been the major metrics that we considered during this project.

Here we have a really important point to take in considerations. In the initial phase of our work we were not able to understand immediately the importance of using in this context the proper metrics in order to find a good results. We noticed that accuracy was not the best way to asses the performance of the model with supervised methods when we observed that with estimators such as Decision Tree or K-Nearest Neighbor the average accuracy for every type of interaction was in the training phase around 0.87 (Unbalanced dataset can lead to high accuracy because the classifier is biased towards the majority class). So, instead of getting more accurate results we have moved our attention to different measures such as the Matthews correlation coefficient, which is very useful for unbalanced dataset because it provides a balanced measure of the classification performance.

After the exploratory data analysis of the dataset, as a preliminary step of our work, we have directed our attention to oversampling/undersampling techniques. In particular, we have implemented a *RandomOverSampler* method (from *imblearn*) in order to increase the representation of the minority classes. Balancing the dataset using a random oversampler can be done at different stages: we have tried both at the beginning and after train-test split and the label assignment. We have additionally implemented also the $SMOTE$ (Synthetic Minority Over-sampling Technique) technique (always available by *imblearn*), which works by creating synthetic instances along the line segments between neighboring minority class instances. By creating new samples in this way, $SMOTE$ introduces diversity in the synthetic data and helps avoid overfitting issues. This peculiarity represents the main difference between the 2 techniques that we have approached. In the training phase of this part of the project we have adopted the $SMOTE$ option, but the *RandomOverSampler* instance has been a fundamental step in order to understand from a theoretical point of view of to manage data. For the MLP otherwise we have played with the dataset in a different way, utilizing undersampling techniques (by randomly remove rows from the majority classes) mainly in order to make more attempts possible with the several parameters that comes into the scenario for a Perceptron.

We want to underline that, here, we have not considered the idea of using different estimators for different types of bond, instead we have tried to strength as much as possible our pipeline and to reproduce it for every type of bond. This suggestion may represent a step forward in the analysis and resolution of this specific problem.

In our project, after analyzing the possible different frameworks, the work of balancing the dataset has been applied as a pre-processing step before dividing it between train and test portions. We have than focused on definition of the models: we have implemented Logistic Softmax Regression, K-Nearest Neighbor and Decision Tree (SVM has been considered as possible model of interest but after some trial has been discard due to time management issue). In order to search for the best combination of hyperparameters in the tuning phase we exploited a Grid Search (thanks to $GridSearchCV$ by *sklearn*) and the metric used to asses the performance of the various models has been balanced accuracy. The Grid Search approach has represented a very time consuming strategy, so we have 'freezed' the data after observing that for at least 3 of the labels that the parameter obtained were quite the same. To get a faster implementation in this phase we have then used a reduced batch of the training set (about 120000 samples per batch). In all the cases Decision Tree has been the model from which we have obtained the best results in this part: with the balanced accuracy the best results were obtained with the PICATION as label of interest and the value was about 0.53. The main problem was about the F1 score for which the values were about 0.1, meaning that the model was not performing well in terms of correctly identifying the minority class

We have then exploited ensemble method in order to obtain better results. Also in this case we have explored several strategy. From one hand we have used a Stacking Classifier: the key idea behind the Stacking Classifier is to leverage the strengths of different base classifiers (the ones cited before) and utilize a meta-classifier (also known as a level-1 classifier) to make a more informed and potentially more accurate prediction. The main drawback of the strategy with a Stacking Classifier approach is time management: it took more than 4 hours in order to train the model using the estimators cited before. Clearly, we have then changed our direction. In a separate manner we have tried with an adaptive boosting ensemble: *AdaBoost*. It works by iteratively training weak classifiers (the base model used was a simple Decision Tree) on different subsets of the training data. In each iteration, *AdaBoost* assigns higher weights to the misclassified samples , forcing subsequent iterations to

focus more on those samples. Last but not least we have also explored the path with a Random Forest estimator.

In order to train the model obtained we have used a cross-validation approach: cross-validation helps to assess the performance and generalization of the model by providing an estimate of its performance on unseen data. We have evaluated the performance of our models using a 5-fold cross validation, every time considering a different label. We have then tested, in each case, the models in a consistent way.

Speaking from an overall point of view (so generalizing the final results for each case), with *AdaBoost* even if we get response during the cross validation phase, the results obtained with the testing set were really discouraging: the balanced accuracy overall was lower than 0.50, the F1 score less than 0.1 and the MCC metric was most of the time a negative value. Using Random Forest, in particular for the HBOND interactions, the results obtained were a bit better touching a maximum for the balanced accuracy (0.65) but there were not significant improvements for F1 and MCC.

Due to the fact that for every single *interaction* label for which we trained and tested our models our results were not encouraging and because of the parallel developing of the MLP, we have not trained the whole model, considering the work too demanding in terms of computational performance.

This part has been useful in order to get a very deep understanding of the theoretical background about the different techniques used: we have tried to implement different approaches for the stages of the pipeline we have followed in order to asses time by time better results and explore possible different solutions to the problem. In order to maximise the performance on what we have developed in the second part we have used part of the results obtained here and through a comparison between us we have decided which procedure would have been the best to devote.

Note: The code of this part has not been reported because has been implemented through several different Jupiter Notebook. In case is required, we will report it at all in a unique file without problem but some modifications will be needed in order to compile it.

SECOND PATH: MLP DEVELOPMENT

In our second strategy we have developed an MLP model and used several techniques to improve the performance and asses the quality of the results, exploiting what we have discovered before. Firstly we have defined a simple Multy Layer Perceptron model using PyTorch: a multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN). This allowed us to specify the architecture of the model and the various hyperparameters, which most impact performance. We have started the work using only 5 nodes and 30 iterations and then we have enriched the model step by step in a coherent way.

Late to handle the data loading and data preprocessing, we implemented a custom Pytorch Dataset class, called ProteinDataset, that allowed us to efficiently and effectively manage data. This helped us a lot reducing the time for the training phase in the different trial that we have done. As cited in the previous chapter, here we have used undersampling techniques.

Moreover, to prevent overfitting with this type of model and improve at the same time the model's ability to generalize the data, we found that implementing an early stopping strategy would have been the most effective way, since that for some label the training data is scarce. It works by monitoring various model performances between epochs and stopping the training in the eventual case that the model stops improving. Early stopping helped us to find a balance between model complexity and generalization by preventing overfitting and ensuring that the model is trained for an appropriate number of epochs.

We also experimented with various hyperparameters to find the best combination for the model. By trying different values for these parameters, we were able to find a combination that resulted in good performance results. In our specific case the optimal training method was a large number of perceptrons in larger batches, with tanh as activation function. This led the output to be a value, for each type of interaction, between -1 and 1 where a value near to 1 means a possible interaction between two different residue.

We have implemented also $RandomForestClassifier$ to find the most relevant features of the model and in order to train the model with better performance and accelerated training time. In other words, we have used Random Forest here to define the best features for each different type of interaction. To evaluate this final model's performance, we calculated used the metric already explained. In summary all these steps helped us develop a multilabel classification model with improved performance on the dataset. Here we report the results obtained.

There has been improvements in the performance of the MLP selecting only the features of interest for each type of contact (defined by 6 and selecting the appropriate parameters: from a 15% of number of ac-

```
Model 0: 12 selected columns
  Names: s_rsa, s_up, s_down, s_phi, s_psi, s_a4, t_rsa, t_up, t_down, t_phi, t_psi, t_a4
  Indices: [0, 1, 2, 3, 4, 8, 10, 11, 12, 13, 14, 18]
Model 1: 3 selected columns
  Names: s_a1, t_a1, t_a5
  Indices: [5, 15, 19]
Model 2: 10 selected columns
  Names: s_rsa, s_up, s_down, s_phi, s_psi, t_rsa, t_up, t_down, t_phi, t_psi
  Indices: [0, 1, 2, 3, 4, 10, 11, 12, 13, 14]
Model 3: 6 selected columns
  Names: s_a3, s_a4, s_a5, t_a1, t_a4, t_a5
  Indices: [7, 8, 9, 15, 18, 19]
Model 4: 6 selected columns
  Names: s_a1, s_a2, s_a4, t_a1, t_a2, t_a4
  Indices: [5, 6, 8, 15, 16, 18]
Model 5: 10 selected columns
  Names: s_rsa, s_up, s_down, s_phi, s_psi, t_rsa, t_up, t_down, t_phi, t_psi
  Indices: [0, 1, 2, 3, 4, 10, 11, 12, 13, 14]
```

FIG. 6: Selected features for every type of contacts.

curate predictions we have passed to a maximum of 30%.

Analyzing the F1 score and MCC there were also some improvements: for the HBOND class the F1 score was about 0.62 and the MCC about 0.22 (best case for this metrics, but a lower accuracy). About PIPISTACK, in order to have a comparison with the preceding path, the F1 score settled to 0.9 but MCC stayed at a value near to 0 (even with different parameters). Every class results can be access through the Jupyter Notebook available.

Watching these little improvements has led us to use the MLP as estimator for our software.

Evaluating the accuracy of the multilabel final estimator has been an hard work but we have noticed the following thing: due to the fact that in the training set a lot of interactions are HBOND or VDW, it is harder for the model evaluate the pattern of this two classes and so it tends assign to every interaction in the protein this two specific bonds.

CONCLUSION

In this report we have explained the paths that we have followed during the project development. We have decide to be more detailed about the first part in order to explain in the best way possible the deeply research that we have done (even without getting the results we were hoping for) and less precise about the MLP because the strategy has been clearly explained also in the code produced for the software realization. We need to emphasize the fact that behind the project there have been a wide study of different possible techniques of Machine Learning as a way of getting more accurate results and to really understand the critical issues that the problem posed. It's been for sure a challenging project that have developed a lot of ideas between us and that we have done in the best way possible.