A Neural Network Approach to Classification of Protein Residue Interactions

Andrea Auletta
andrea.auletta@studenti.unipd.it

Marco Bernardi
marco.bernardi.11@studenti.unipd.it

Niccolò Zenaro
niccolo.zenaro@studenti.unipd.it

Abstract

This paper presents a machine learning approach to classify protein residue interactions within the framework of Residue Interaction Networks (RINs). Utilizing the RING tool, we obtained a dataset containing over 2.4 million residue-residue contacts, categorized into several interaction types. Given the inherent class imbalance in the dataset, with a significant overrepresentation of certain interaction types such as hydrogen bonds and van der Waals interactions, multiple techniques were explored to address this imbalance. These included undersampling, oversampling using the Synthetic Minority Over-sampling Technique (SMOTE), and class merging strategies. We implemented and compared several models, including XGBoost and neural networks, focusing on their ability to accurately predict the interaction type. The challenges posed by the overlap of classes in the feature space are discussed, alongside the preprocessing steps taken to enhance model performance. The results of our analysis highlight the effectiveness of the proposed methods and provide insights into the complexities of classifying protein residue interactions.

1. Introduction

Residue Interaction Networks (RINs) are a representation of the non-covalent interactions between amino acid residues within a protein structure, derived based on their geometrical and physico-chemical properties. These networks provide a detailed mapping of intra-protein contacts, which are crucial for understanding the structural and functional dynamics of proteins. RING (`https://ring.biocomputingup.it/`) is a computational tool that facilitates the analysis of these networks by processing Protein Data Bank (PDB) files (`https://www.rcsb.org/`) to identify and classify residue-residue interactions within a given protein structure. RING categorizes these interactions into distinct contact types, including Hydrogen Bonds (HBOND), Van der Waals interactions (VDW), Disulfide Bridges (SBOND), Salt Bridges (IONIC), π-π Stacking (PIPISTACK), π-Cation Interactions (PICATION), Hydrogen-Halogen Interactions (HALOGEN), Metal Ion Coordination (METAL_ION), π-Hydrogen Bonds (PIHBOND), and a category for Unclassified Contacts.

This project is centered around the development of a predictive model that can infer the RING classification of residue contacts using statistical or supervised learning approaches, as opposed to purely geometrical methods. The objective is to design a program that calculates the likelihood or propensity of a residue-residue interaction belonging to each contact type defined by RING, starting from the structural data of the protein.

2. Data Source

The dataset utilized in this study comprises a collection of training examples derived from 3,299 Protein Data Bank (PDB) structures. Each PDB structure is represented by a separate file, which provides detailed information on the residue-residue contacts identified within the corresponding protein. The files are available for download and are organized such that each file contains a tab-separated table of interactions for a single PDB structure.

Across all 3,299 PDB structures, the dataset contains a total of 2,476,056 residue-residue contacts, distributed among various interaction types as follows:

Each file is formatted as a tab-separated table, consisting of columns that provide essential details for each contact. The columns include residue identifiers, which follow the same naming conventions as those used in BioPython, along with several pre-calculated features. The final column in each table specifies the type of interaction, categorizing the contact according to the RING-defined classifications.

Contact Type	Count
HBOND	901,814
VDW	640,469
PIPISTACK	32,965
IONIC	30,355
SSBOND	1,792
PICATION	7,623
PIHBOND	1,836
Unclassified	860,202

Table 1: Distribution of contact types across the dataset.

2.1. Data Preprocessing

The files containing the information for each PDB structure were merged into a single dataframe, creating the dataset for the model. This dataset underwent a preprocessing phase, which involved several crucial steps for data cleaning and preparation.

First, the null values present in the dataset were replaced with the mean of the corresponding column values. This operation ensured data continuity, minimizing the impact of missing values on the model's performance.

Furthermore, all rows lacking a classification, i.e., without a value in the "interaction" column, were reclassified under the "Unclassified" category. This update involved modifying the values in the "interaction" column, ensuring that every interaction in the dataset was consistently labeled according to the contact categories defined by RING.

3. Approaches

3.1. Balancing the Dataset

As observed in Section 2, the dataset exhibits a significant imbalance, with a disproportionately large number of HBOND, VDW, and Unclassified interactions compared to the other contact types. Various techniques have been employed to address this imbalance, and these methods are detailed in the subsequent sections.

Undersampling The dataset was balanced by reducing the number of samples in the classes with a higher frequency to match the size of the class with the fewest samples, or to a lower, predefined number. This reduction was performed randomly. However, this approach presents the drawback of discarding a substantial amount of data, which may result in the model's inability to fully learn the distinguishing features of the classes with a larger number of samples.

Oversampling - SMOTE The Synthetic Minority Oversampling Technique (SMOTE)[1] was applied to generate synthetic samples for the minority classes, thereby increasing their representation in the dataset. This method involves calculating the difference between a sample and its nearest neighbors, multiplying this difference by a random number between 0 and 1, and then adding the result to the original sample to create a new synthetic instance. To manage computational resources and minimize the risk of introducing excessive noise into the dataset, the size of the minority classes was increased by approximately tenfold.

Class Merging To further address the class imbalance in the dataset, several strategies were explored:

- Merging all majority classes into a single class;
- Merging all minority classes into a single class;
- Implementing a combination of the two approaches above.

Challenges in Classification As detailed in the following sections, the model struggles to accurately distinguish and classify samples from the majority classes, despite the abundance of data relative to the minority classes. This difficulty arises from the significant overlap between classes when visualized in a 2D feature space, where the classes appear highly intermixed as we can see in the image 1. Initial efforts were directed toward addressing this specific challenge. In the 2 we can see also that both the majority and minority classes are really mixed (this graphic has been done by taking a subset composed for each classes, all the subset have the same dimension).

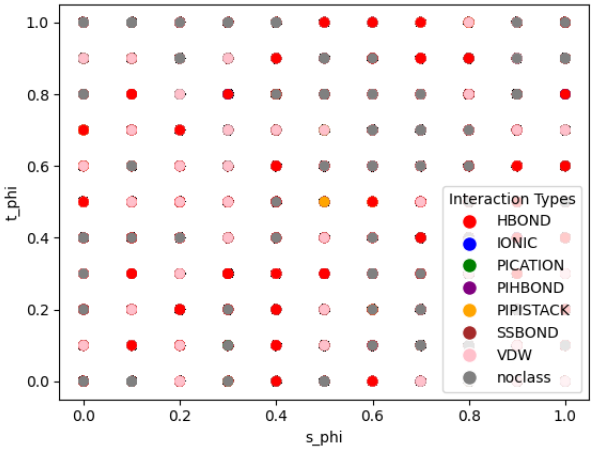

Figure 1: Scatter plot of the dataset

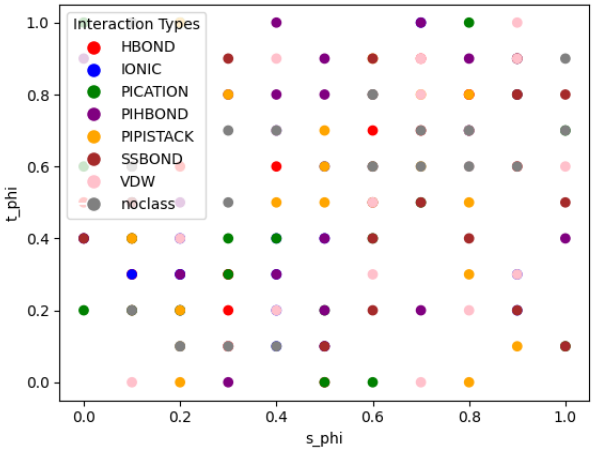

Figure 2: Scatter plot of a sampled dataset

3.2. XGBoost

XGBoost is an implementation of a gradient boosting[1] algorithm and it constructs a series of decision tree and improves the model each iteration. We tested this model both with the initial dataset and the dataset with the SMOTE applied.

3.2.1 Parameter Selection

The parameters that seemed most important to us in the modification were the following:

- Max_depth: this is the maximum depth of the tree, which is a crucial parameter to avoid overfitting. It is setted to 10, which is a good compromise between the model's complexity and the risk of overfitting;
- Learning_rate: setted to 0.1. This is the right value, an higher gives to us bad results and a lower value makes the model too slow;
- num_boost_round: this is the number of iteration. It is setted to 2500. After 2500 iteration is diffucult to see an improvement in the model;
- early_stopping_rounds: this is the number of iteration after which the model stops if it doesn't improve. It is setted to 20.

3.3. Neural Network Architecture

The neural network model implemented in this project is a fully connected feedforward neural network, designed to classify protein residue interactions based on the features derived from the dataset. The architecture, named 'SimpleNN', consists of four layers, each with specific roles and activation functions chosen to optimize the model's performance on this multi-class classification task.

3.3.1 Model Architecture

The model consists of four fully connected layers: an input layer, two hidden layers, and an output layer. The hidden layers use ReLU activation functions to process and refine the features, while dropout layers are applied after each hidden layer to prevent overfitting. The final output layer uses a softmax activation function to produce probability distributions across the target classes, suitable for multi-class classification.

3.3.2 Model Training

For the training process, the Adam optimizer was employed alongside the Cross-Entropy loss function. The Adam optimizer was chosen due to its ability to efficiently handle large datasets and high-dimensional parameter spaces by adapting learning rates based on the estimates of first and second moments of the gradients. The Cross-Entropy loss function was selected because it effectively quantifies the difference between the predicted probabilities and the true class labels, making it well-suited for classification tasks. Training was conducted using mini-batch gradient descent. For each mini-batch, predictions were computed, the loss was calculated, and the model weights were subsequently updated. This approach allows for more frequent updates and efficient handling of large datasets.

4. Model Evaluation Metrics

The quality of the model was assessed using various evaluation metrics, including Matthews Correlation Coefficient (MCC), Balanced Accuracy, and ROC-AUC. These metrics were closely monitored throughout the training process to track the model's performance and progression.

4.1. Rationale for Metric Selection

The chosen metrics are particularly suited for handling imbalanced class distributions. Each metric provides distinct advantages in evaluating model performance:

- **Matthews Correlation Coefficient (MCC):** MCC is a robust metric that considers true and false positives as well as true and false negatives, making it effective for evaluating the performance of models on imbalanced datasets. Its ability to capture the balance between these quantities provides a comprehensive measure of classification quality;
- **Balanced Accuracy:** This metric calculates the average recall per class, addressing the issue of class imbalance by ensuring that each class contributes equally

[1]The boosting is an ensemble learning technique which imporove the performances of the model by combining weak model

to the accuracy score. Balanced Accuracy provides a more nuanced view of the model's performance across different classes, especially when some classes are underrepresented.;

- **ROC-AUC Score:** The ROC-AUC score evaluates the model's ability to distinguish between classes across all classification thresholds. It is particularly useful for imbalanced datasets as it measures the overall capability of the model to rank positive instances higher than negative ones. A higher ROC-AUC score indicates better performance in differentiating between classes;
- **Average Precision:** The Average Precision score summarize the precision-recall curve as the weighted mean of precisions achieved at different threshold. A high value of this metric indicates a better performance of the model in the classification of the true positive instances.

Monitoring these metrics during training allows for a detailed assessment of the model's effectiveness and aids in the identification of potential areas for improvement, especially in the context of imbalanced datasets.

4.2. Feature Importance and Feature Selection

The model XGBoost is also capable of providing the importance of the features. It has three different methods to calculate the importance of the features:

- weight: total number of times a feature is used to split data across all trees;
- gain: average loss reduction gained when using a feature for splitting;
- cover: the number of times a feature is used to split data across trees weighted by training data points.

In the image 3 we can see the score given to each feature and the importance of them for the classification of the interactions. The dataset has been modified by considering all the three method and a different number of feature for each type. The results were not better than the XGBoost model which used the dataset with SMOTE and which considered all the features.

5. Testing and results analysis

5.0.1 Discussion

The results reported show a substantial difference when dealing with SMOTE dataset and the initial one. Overall XGBoost performs well in both of them, but *balanced*

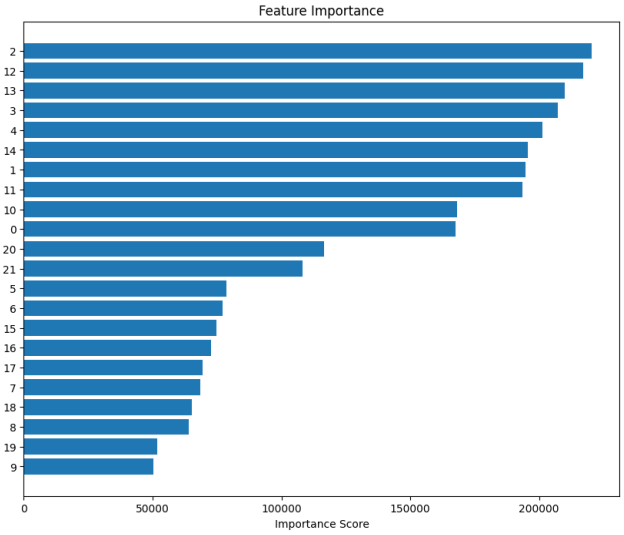

Figure 3: Feature importance given by XGBoost model (weight)

Model	**Bal Acc**	**MCC**	**ROC AUC**	**AVG Prec**
XGBoost (Initial DS)	0.3669	0.2956	0.8741	0.5193
XGBoost (SMOTE)	**0.6835**	**0.4956**	**0.9102**	**0.6106**
XGBoost (Feature Selection)	0.5161	0.3554	0.8516	0.5080
SimpleNN (SMOTE)	0.	0.	0.	0.

Table 2: Results of the models

accuracy and *MCC* see an important improvement using SMOTE, almost doubling their values.

The confusion matrix 4 helps us understanding exactly where the model is doing wrong or right predictions. As we can see, the five less numerous classes are correctly classified by the model; thus the model is able to deal with the imbalance of the dataset. The problem arises when the model has to classify the three most prominent classes, because the '*VDW*' bonds are really misleading for the right classification. That type of bonds are the worst classified of the entire dataset, leading to bad scores (Precision Score for VDW is 0.32) affecting the entire model. In fact, note that the True Positives for *VDW* are around 15.000 examples while the *VDW* classified as *HBOND* are 4 times more numerous and the number of *VDW* classified as *noclass* is 3 times bigger.

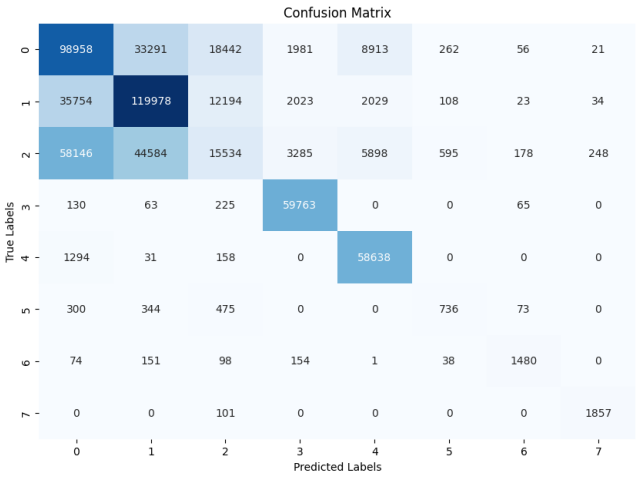

Figure 4: Confusion matrix of the XGBoost model with SMOTE

6. Conclusions

Finally we can say the best approach we've tried have been that one which uses the XGBoost with the SMOTE applied to the dataset as we can see in the table 2. The model has a good performance in the classification of the interactions and the confusion matrix shows that the model is able to classify the interactions in a good way. The feature selection didn't give us better results than the model with all the features. The main problems have been the imbalance of the dataset and the overlap of the majority classes in the feature space. So, if we had more resources we could have thought of other more expensive approaches such as trying to project all the data in a higher dimension space in such a way as to better separate them.

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.