Machine Learning Based Classification of Inter-Residue Contacts in Protein Structures

Davide Baggio
davide.baggio.1@studenti.unipd.it

Sebastiano Sanson
sebastiano.sanson@studenti.unipd.it

Abstract

Predicting the type of inter-residue contact in a protein structure is important for understanding its interaction network. In this project, we frame the RING contact classification task as a supervised learning problem, using gradient-boosted trees with XGBoost to predict each contact's classification just by structural features. Models are trained on a large dataset of residue-residue pairs derived from thousands of PDB structures previously labeled. To address class imbalance, we evaluate performance using tailored multiclass metrics. We compare two strategies: (1) a unified multiclass XGBoost model and (2) an ensemble of one-vs-all binary XGBoost classifiers with logistic objectives, analyzing their respective advantages for this biological prediction task.

1. Introduction

Residue Interaction Networks represent a protein's 3D structure as a graph of amino acid contacts, capturing non-covalent interactions based on geometrical and physico-chemical criteria. RING software identifies such contacts from a PDB structure and assigns each a type out of 10 distinct classes. Traditionally, RING classification relies on geometric rules; here, we instead train a data-driven model to predict the RING type of a contact from the features of the interacting residues. Predicting contact types by statistical methods can help validate and complement physics based annotations. In this study we use XGBoost to classify contact types, aiming to reproduce RING's labeling from structure-derived features. We leverage the provided training data and compare a multiclass classification approach to a one-vs-all scheme.

2. Dataset

The dataset comprises a collection of example from 3,914 PDBs, containing one line per residue–residue contact, with a highly imbalanced distribution of RING types. Each record lists identifiers of the two residues and a variety of structural features for each "source" and "target" residue, plus the RING interaction type label (Table 1).

Contact type	Count
HBOND	1,055,929
VDW	737,0610
PIPISTACK	38,283
IONIC	35,391
PICATION	8,885
SSBOND	2,100
PIHBOND	1,790
Unclassified	1,089,547

Table 1: Distribution of contact types in the dataset.

2.1. Data Preprocessing

The raw feature files were merged into a single DataFrame. We first handled missing and duplicated data: any contact with a missing Interaction label was relabeled as "Unclassified".
In the multiclass approach (Section 3.1), we removed duplicate residue pairs with conflicting contact type labels while retaining a single representative instance to prevent data redundancy. When resolving duplicates, we prioritized the most frequent contact type to help balance class distribution.
In the one-vs-all approach (Section 3.2), this deduplication step is unnecessary as each binary classifier is trained independently for its respective contact type, making duplicate entries with conflicting labels unproblematic.
For numerical features, missing values were imputed using column means, resulting in complete data with no remaining null values. Categorical features were converted to integer representations, including label encoding of the DSSP secondary structure states (`s_ss8` and `t_ss8`). The `3di_letter` feature was removed as redundant, since its information is fully captured by the `3di_state`.

2.2. Data Splits

The preprocessed data was then split into:

- **Training**: 81%
- **Validation**: 9%
- **Test**: 10%.

This splitting strategy ensures model evaluation on independent validation and test sets, while preserving the original class distribution through stratification.

2.3. Feature Engineering

For each contact (residue pair) we used structural descriptors of both the source and target residues. These include:

- **Secondary structure (DSSP)**: 8-state labels for each residue (columns `s_ss8`, `t_ss8`), later encoded as integers.
- **Solvent accessibility (RSA)**: relative solvent-accessibility values (`s_rsa`, `t_rsa`).
- **Backbone dihedral angles**: phi and psi angles for each residue (`s_phi`, `s_psi`, `t_phi`, `t_psi`).
- **Atchley factors**: five physiochemical factor scores per residue (`[s_a1, ..., s_a5]`, `[t_a1, ..., t_a5]`).
- **3Di structural state**: the 3Di alphabet state index for each residue (`s_3di_state`, `t_3di_state`).

In this step, pairwise feature operations were systematically applied to corresponding features between source and target structures. For each matched feature pair, four mathematical operations were computed:

- *sum*: Addition of source and target feature values
- *product*: Multiplication of source and target feature values
- *absolute difference*: Absolute value of the difference between source and target feature values
- *average*: Mean of source and target feature values

These operations were implemented to generate composite features that capture different aspects of the structural relationships between source and target. The rationale was to explore whether linear combinations of these mathematical transformations could enhance model performance by providing complementary information about structural similarities and differences.

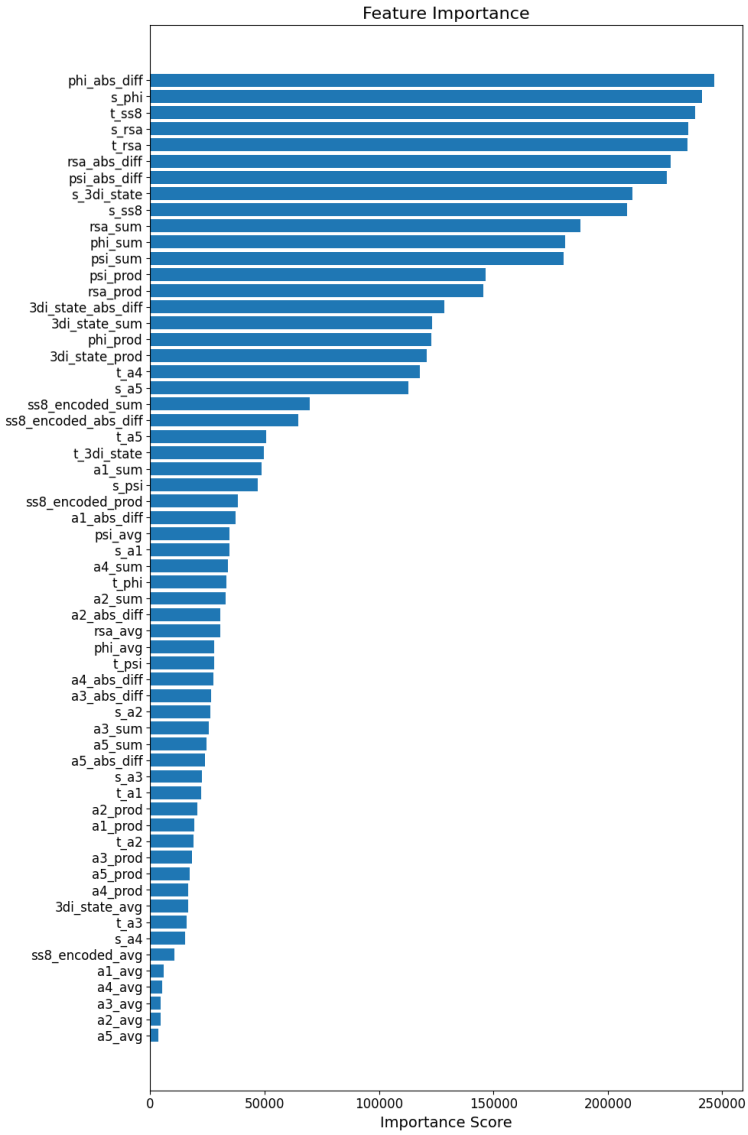

Figure 1: Feature importance plot for multi-class model.

Notably, the engineered features generated during the feature engineering phase demonstrated measurable contributions. As illustrated in Figure 1, the composite features derived from pairwise mathematical operations appear among the informative predictors, indicating their utility in capturing relevant structural relationships.
However, despite the contribution of some engineered features, the performances are not improved, as you can see in the Section 5.

2.4. Balancing dataset

Due to the highly imbalanced nature of the data set (with some classes such as 'Unclassified' and 'HBOND' dominating), we applied class balancing techniques. SMOTE[1]

was used to generate synthetic examples for the minority classes, ensuring that each class had a more balanced representation exclusively in the training set, keeping validation and test sets unmodified. This technically helps to prevent the model from being biased towards the majority class and improves its ability to generalize across all contact types. The sampling strategy used is manually setting the values in order to oversampling the minority classes, obtaining the following distribution:

Contact type	Count
HBOND	545,494
VDW	561,545
PIPISTACK	200,000
IONIC	200,000
PICATION	150,000
SSBOND	40,000
PIHBOND	40,000
Unclassified	882,533

Table 2: SMOTE distribution of contact types.

In both the multiclass and One-vs-All approaches, we initially employed XGBoost's `scale_pos_weight` parameter[3] to address class imbalance, setting its value for each class as $sum(negativeinstances)/sum(positiveinstances)$. However, after empirical evaluation, we observed superior performance when using the `compute_sample_weight`[2] function from scikit-learn, which dynamically estimates class specific sample weights during training. This adjustment was adopted in both final implementations.

3. Approaches

XGBoost was chosen over other machine learning alternatives due to its proven dominance in structured tabular data tasks, since the dataset consists of engineered structural features that are heterogeneous and non-linear in their relationships. Gradient boosted trees excel at capturing such complex interactions through sequential error correction, often outperforming simpler models like logistic regression and even neural networks. Additionally, the dataset exhibits significant class imbalance and XGBoost's inherent mechanisms allows robust handling of skewed distributions make it an appropriate choice. The framework employs parallelized gradient boosting, which iteratively combines weak learners to form a highly accurate ensemble model, while efficiently optimizing computational performance through distributed processing.
The following subsections provide a detailed discussion of both approaches.

3.1. Multi-class Model

A single XGBoost classifier was trained using the `multi:softprob` objective function[3] to perform 8-way classification, distinguishing between seven distinct contact classes and an "Unclassified" category.

3.1.1 Parameters[3]

The model hyperparameters were optimized through manual tuning and included:

- `max_depth: 15`
  Maximum depth of a tree, increasing this value will make the model more complex and more likely to overfit. We found the best value to be 15, as we obtained the best metrics;
- `learning_rate: 0.03`
  This shrinkage parameter controls how much each new tree contributes to the ensemble. We set it to 0.03 in order to make the boosting process more conservative;
- `subsample: 0.8`
  This parameter controls the fraction of training instances randomly selected for growing each tree, introducing diversity, reducing overfitting but maintaining stable gradients:
- `num_boost_round: 10000`
  Number of maximum iterations allowed, however with `early_stopping_rounds` of 50, it has never reached such threshold.

Model evaluation was conducted using *multiclass logloss* and *multiclass classification error rate* metrics.

3.2. One-vs-All Ensemble Model

In parallel, we implemented an ensemble of eight binary XGBoost classifiers, with each classifier dedicated to one specific contact class. For each class i, training samples belonging to class i were labeled as positive (1), while all remaining samples were labeled as negative (0). Each binary classifier utilized the `binary:logistic` objective function[3].

3.2.1 Parameters[3]

The binary models shared common hyperparameters:

- `learning_rate: 0.03`
  Maximum delta step we allow each leaf output to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.

- `num_boost_round: 2500`
- `early_stopping_rounds: 50`
- `max_depth: 15`

Model evaluation was conducted using *Area Under the Curve (AUC)* metric. During prediction, each of the eight binary classifiers output a probability score, and the final class assignment was determined by selecting the class with the highest probability across all binary models.

3.3. Model Comparison and Hyperparameter Optimization

Both methodologies employed a consistent train/validation split derived from the provided dataset. Rather than conducting exhaustive automated hyperparameter search, we performed targeted manual adjustment of key parameters (tree depth and regularization terms) in order to maximize the above mentioned metrics. The implementation details for both approaches are documented in the accompanying computational notebooks, with complete parameter specifications and training procedures available in the source code.

3.4. Feature selection

Feature selection was performed to evaluate whether retaining only the most informative features would enhance model performance. This approach involved identifying and preserving features with the highest predictive importance while removing all remaining features from the dataset. However, across all experimental conditions tested, feature selection consistently resulted in degraded model performance compared to the full feature set. These results suggest that the complete feature space contributes meaningful information to the classification task.

4. Model Evaluation Metrics

We evaluated performance using metrics that account for class imbalance and multi-class scoring. As specified in the project requirements, we computed:

- **Balanced accuracy**: the average of recall (true positive rate) over all classes;
- **Matthews Correlation Coefficient** (MCC): a correlation coefficient between true and predicted labels. It ranges from -1 (inverse prediction) to 1 (perfect), with 0 meaning randomness;
- **ROC-AUC**: area under the Receiver Operating Characteristic curve, values below 0.5 indicates bad predictions, 0.5 no discriminative power and 1 a perfect separation.
- **Average Precision** : area under the precision-recall curve, indicative of performance on positive class detection.

5. Testing and results analysis

Model	**BAL Acc**	**MCC**	**AUC-ROC**	**AVG Prec**
McC Duplicates	0.6832	0.2495	0.8600	0.3269
McC No Duplicates	0.7296	0.2986	0.8804	0.5592
McC SMOTE No Duplicates	0.6230	0.2909	0.8804	0.5688
OvA	0.7141	0.3153	0.8697	0.3492
OvA FE SMOTE	0.6727	0.2953	0.8681	0.3253

Table 3: All metrics obtained.

After training, we evaluated both models on held-out test set. The Multiclass XGBoost model with no duplicates achieved overall the best metrics among all our trials. The One-vs-All ensemble performed slightly better on balanced accuracy, MCC and AUC-ROC, however obtaining a significantly lower average precision. Overall, both models performed substantially better than random chance, although their numeric scores indicate only moderate predictive power.

The primary challenge stemmed from the dataset's severe class imbalance, particularly in predicting minority categories. The 'VDW' class proved to be the most problematic one, as it was frequently misclassified as either 'HBOND' or 'Unclassified' due to distribution skew. For the minority classes, we observed consistent overfitting on 'PICATION' and 'SSBOND', while 'PIHBOND' remained poorly predicted regardless of the adjustments made. Another major difficulty emerged in combining predictions from the eight One-vs-All classifiers. While individual models achieved strong performance (with average accuracies around 80%), their ensemble suffered significant degradation in performance. This decline reflects the inherent complexity of calibrating confidence scores across multiple specialized classifiers, a known limitation in multi-model systems where decision boundaries may conflict.

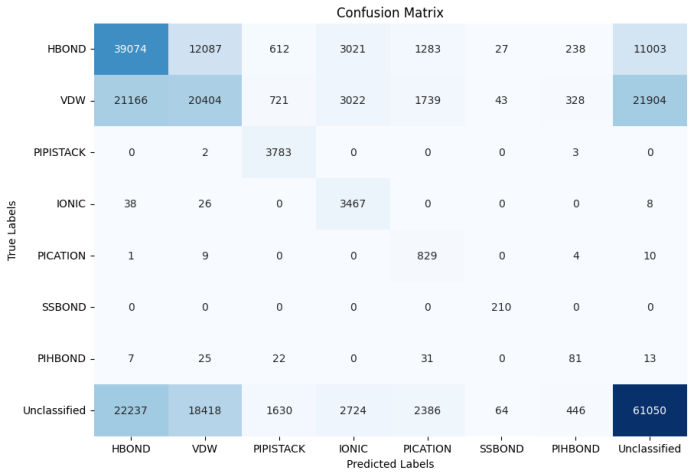

Figure 2: Confusion matrix for the best-performing multi-class model.

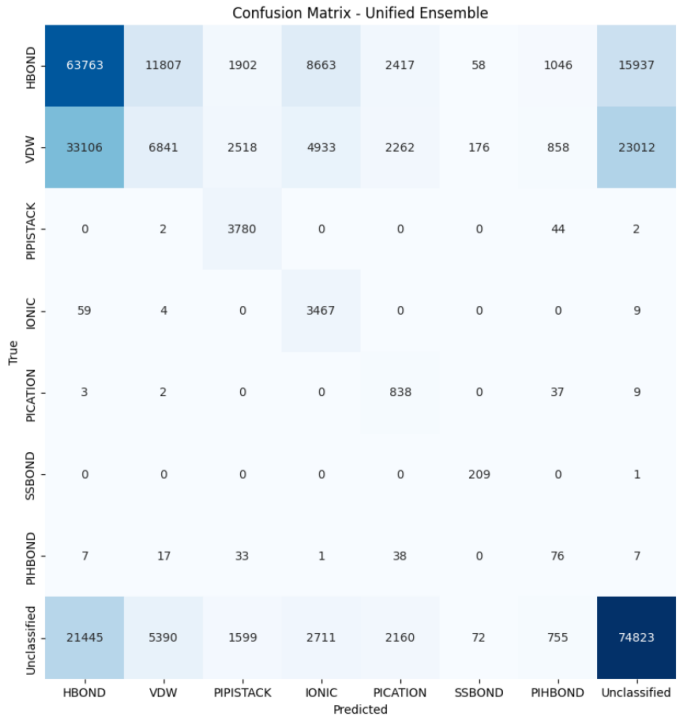

Figure 3: Confusion matrix for the best-performing OvA model.

6. Conclusions

In conclusion, gradient-boosted trees can learn to predict residue–residue contact types from structural features with performance well above random, confirming that the features (DSSP, RSA, angles, Atchley, 3Di) carry information about interaction class. Both a single multiclass model and a one-vs-all ensemble achieved similar results (Balanced Acc $\approx$ 0.71, MCC $\approx$ 0.30, AUC-ROC $\approx$ 0.85) in this dataset. However in the multiclass model, the average precision's metric shows a significant improvement in the chosen metrics.

When applied in a real use case scenario, the multi-class model tends to default to predicting 'Unclassified' when uncertain, rarely assigning borderline cases to classes like 'HBOND' or 'VDW'.

On the other hand, the OvA model's predictions are more diversified than multi-class ones, so it seems that this approach, at inference time, performs better.

While XGBoost shows promise for this task, accurately distinguishing abundant but ambiguous interactions such as 'Unclassified' remains challenging. Future work could explore additional input features or alternative model architectures to improve the classification of such borderline contacts.

References

[1] Jason Brownlee. *SMOTE for Imbalanced Classification with Python*. URL: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.

[2] scikit-learn developers. *compute_sample_weight*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_sample_weight.html.

[3] xgboost developers. *XGBoost Parameters*. URL: https://xgboost.readthedocs.io/en/stable/parameter.html.