Characterizing Protein-Protein Interaction Sites from Sequence Data using Tree Ensemble Approaches

Group 11

Gyula Maloveczky (2865813), Jacob M. Reaves (2865836), Shabnam Shajahan (2852254), Ana Esteve Garcia (2725481), Jasmine Y. Chen (2748748)

Course: Scientific Machine Learning
Course Code: XM_0138
Date of Submission: 28/03/2025

Abstract

Essential biological processes often depend on protein-protein interaction (PPI) interfaces. Examples include signal transduction, enzymatic activity, and immune responses. Interface residue properties can provide insight into protein function, cellular mechanism, and disease pathways. Experimentally identifying interface residues is expensive and time-consuming, therefore machine learning (ML) methods using sequence-based and structural features have been developed to predict the interface residues. In this study, ML interface residue prediction is used to investigate properties of interaction residues and explore which residue window size, 3, 5, 7, or 9, provides the best model performance. We compared decision tree, random forest, and XGBoost models, choosing to use random forest due to its performance and interpretability. Feature window sizes were then compared using the random forest model. The results suggest that PPI interface residues are generally defined by low windowed PSSM scores, low hydrophobicity, and high surface accessibility. The best performing window size found was window size 9, however, our tests also suggest combining the windowed scores with the non-windowed scores is detrimental to the overall performance of a model, which was unexpected. Low windowed PSSM scores are expected, as it implies diverse interface residue neighborhoods. Low hydrophobicity contradicts what was thought previously, as interface residues are expected to be hydrophobic. High surface accessibility is as expected, since residues usually need to be on the protein surface to interact with other proteins. 9 was the greatest window size we tested, and so provided the most information about the residue neighborhood of all window sizes, therefore we also expected it to perform best compared to other window sizes. Notably, our model underperformed compared to the PIPENN model, with an ROC-AUC of only 0.652. Therefore, the defining features of interface residues found by this study may only represent surface-level patterns.

1 Introduction

Protein-protein interactions (PPIs) are fundamental to numerous biological and cellular processes, including transcription regulation, signal transduction, enzymatic activity, and immune responses [Stringer et al., 2022]. Understanding these interactions provides insight into protein function, cellular mechanism, and disease pathways. PPIs are also crucial in drug discovery, as disrupting or enhancing specific interactions can lead to targeted therapies [Zhou et al., 2016, Xue et al., 2015]. PPIs occur through a specific region known as the interface, which facilitates the binding between proteins. The residues found in the interface are often characterized by distinct physicochemical properties such as hydrophobicity, polarity, and solvent accessibility [Xue et al., 2015]. Compared to non-interface residues, interface residues are more likely to be partially solvent-accessible, as they need to interact with their binding partners while maintaining structural stability. They also tend to be enriched in specific amino acids, such as aromatic (tryptophan, tyrosine, phenylalanine) and certain polar residues (arginine and histidine) as well as most hydrophobic residues, which contribute to interaction specificity and stability [Xue et al., 2015, Yan et al., 2008]. A schematic representation of protein interactions, highlighting interface residues, is shown in **Figure 1**. Experimentally, identifying the residues in the interface is expensive and time-consuming as it uses techniques like X-ray crystallography or Nuclear Magnetic Resonance spectroscopy, therefore computational methods using sequence-based and structural features have been developed to predict the interface residues [Stringer et al., 2022, Xue et al., 2015, Cheng et al., 2021]. With the availability of large, labeled datasets and the inherent patterns in residue-level biochemical properties, Machine Learning (ML) offers a powerful approach for predicting interface residues. Prior studies have shown that evolutionary and physicochemical properties extracted from sequences can effectively distinguish interface residues from non-interface residues [Xue et al., 2015]. A key aspect in this approach is the use of windowed features, where feature values are averaged over neighboring residues to capture both local and broader sequence contexts. Multiple window sizes (e.g., 3, 5, 7, or 9 residues) can be used simultaneously to integrate both localized and more global sequence information. However, in this report, we focus on selecting a single window size to balance interpretability and performance, as previous work has shown that using all window sizes does not necessarily lead to significantly better results [Sikić et al., 2009].

Our dataset consists of 65,150 residues annotated with structural, physicochemical, and evolutionary characteristics. The dataset includes protein sequence information (e.g., residue identity, position, and protein length), physicochemical properties (e.g., surface accessibility and hydropathy index), secondary structure prediction probabilities (beta sheet, alpha helix and coil/ loop) and evolutionary profiles represented by position-specific scoring matrices (PSSM). Several features, namely relative and absolute surface accessibility, PSSMs, and secondary structure annotations (sheet, helix, coil), are averaged using a sliding window approach, where values are averaged over different window sizes. The target variable indicates whether a residue belongs to a protein-protein interface, based on the annotation criteria from BioLip [Yang et al., 2013], that is, one of the atoms of the residue is closer to an atom of a ligand than the sum of the two atom's Van der Waals radii plus 0.5Å in their 3D structure. The clear labels make the dataset well suited for supervised machine learning. However we have to note that this annotation (and thus our model) makes a closed world assumption. By training a model on this data we assume that the negative labels are accurate, however we have no way of knowing if it is true, since it is possible that the negatively labeled residues are also part of interfaces, but those interfaces have not been described yet.

In this report, our objective was to explore which window size provides the best model performance and investigate underlying biological patterns in the interface residues and their context. Specifically, we seek to determine whether unique feature patterns distinguish interface residues from non-interface residues. By addressing these questions, we aim to enhance the understanding of interface residues and improve computational predictions for PPIs, for example helping to identify key interaction sites that could serve as potential drug targets, as well as providing information on general properties of these sites, which can inform general properties of potential drug molecules, making High-throughput target-based screenings and docking simulations more targeted and efficient.

2 Methods

2.1 Data Preprocessing

Data quality significantly influences the performance of a model and its ability to generalize. The initial critical step after data collection is to ensure the integrity of the dataset by applying a range of preprocessing techniques [Albahra et al., 2023].

We determine our choice of preprocessing tasks based on Exploratory Data Analysis

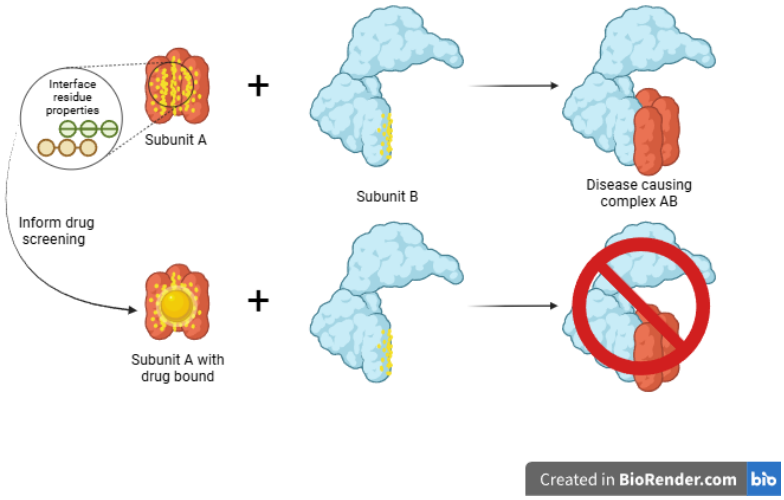


Figure 1: **Biological Background Schematic** showing the relevance of learning interface residue properties to inform drug screening

(EDA) and the ML task that we abstract our research questions to. Therefore, we perform some of the most common data preprocessing strategies that deal with tabular data including checking for missing values and the detection of outliers. Additionally, we perform feature encoding to transform one of our categorical features to a numerical feature. While ML models like decision trees could in theory handle categorical features, considering the specific implementations for our task, we resorted to convert them to numerical features. Finally, we perform a range of resampling strategies to compare with the cost-sensitive learning approach of using class weights and select the one that efficiently handles class imbalance for our chosen data set.

Missing values in the dataset can arise due to several reasons including data unavailability, and improper data entries. These issues mostly depend on the source of the dataset and can cause data inconsistency, thus affecting the overall performance of the ML algorithm [Hasan et al., 2021]. Upon inspecting our dataset, we found no missing values.

Outliers refer to data points that deviate from the rest of the data distribution [Ansari et al., 2024]. EDA and visualization of the statistical distribution of the set of characteristics in our data suggested the presence of extreme values. Considering the characteristic tail in our distribution, we recognize that log normal distributions often contain large values as part of their natural spread. Therefore, rather than viewing these extreme values as potential outliers, and discarding them, we consider these values to hold essential information with respect to our research questions and retain them in our dataset.

Categorical encoding involves the transformation of categorical data to numerical data without any distortion to the data distribution. This is particularly important because categorical features are not handled by the ML algorithms that we propose. Here, we replace one of our features 'sequence' that describes the 'Amino acid type' directly with the 'Eisenberg Hydrophobicity scales' [Eisenberg et al., 1984]. We opt for label encoding instead of one-hot encoding because there is a natural ordering to the amino acids that is highly relevant to our classification task. The Eisenberg scale is also very similar to the hydropathy indices in the dataset corresponding to the ordering and identities of the amino acids. The only difference is that certain amino acids are grouped together. Consequently, we drop the hydropathy index from our dataset as it is now redundant [Rego et al., 2021].

Class imbalance is one of the most challenging tasks in ML because the accuracy and reliability of a trained model depends on the balance of class distributions in the dataset [Werner de Vargas et al., 2023]. To check if our dataset suffers from class imbalance, we first visualize the class distribution of our target variable. 'Class 0' outnumbered 'Class 1' in our dataset 9:1. Hence, we tested several data-level and algorithmic-level strategies to handle this class imbalance in our data set.

Data-level class imbalance strategies include resampling data before applying ML algorithms through undersampling the majority class, oversampling the minority class or using hybrid resampling techniques. The methods that we ap-

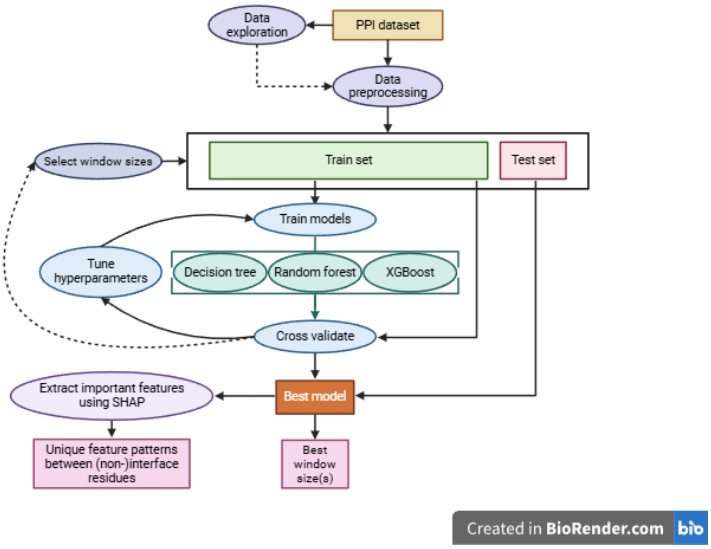


Figure 2: **Workflow Schematic** outlining steps of the study

ply here are Random Undersampling, Synthetic Minority Oversampling Technique (SMOTE), SMOTE-Tomek, and Adaptive Synthetic Sampling (ADASYN), from the imblearn python libary [Lemaître et al., 2017]. Further, we also test and compare them with the cost-sensitive learning approach, an algorithmic-level class imbalance strategy which uses algorithms that are optimized to handle class imbalance [Werner de Vargas et al., 2023].

We compared the Receiver Operating Characteristic - Area Under the Curve (ROC-AUC, also referred to as simply AUC) values for each of these resampling methods and observed that the 'Cost-sensitive' learning approach slightly outperformed other methods (**Figure S12**). As a result, we proceed to use this as our strategy to handle class imbalance. We used a random forest (RF) model to evaluate the different class imbalance strategies. To optimize the hyperparameters in the case of each of the strategies we have used our own implementation of grid search described under the hyperparameter optimiziation subsection.

2.2 Cross-Validation and Train/Test Split

After the EDA decided to define the way we were going to selected feature sets and models. For this end, we used cross-validation (CV); to estimate the performance of different models trained on different feature sets. Our first step was to split our dataset into train (80%) and test (20%) sets, both making sure to have the same proportion of target variable in each set and to have all the residues belonging to one protein end up in the same set. The latter is important to avoid data leakage, as adjacent residues have information about each other (mostly due to the window features). For reproducibility, our train and test sets can be found in our GitHub repository.

We then used the training data for the train and validation set (also 80/20). Using an appropriate CV scheme is important for setting the hyperparameters of our model as well as making decisions about features, like feature selection or feature engineering. We used k-fold CV with 5 as the number of folds. K-fold CV involves splitting the data into k equal-sized partitions called folds, and then doing k iterations, in each iteration evaluating on one fold while training the data on the rest of the folds (training on k-1). In the next iteration using a different fold for evaluation. After k iterations, the scores for each fold are averaged.

We also have made sure that no residue groups from the same protein end up in different folds, just like in the train-test split. For this purpose GroupKFold was used from the sklearn library [Pedregosa et al., 2011]. We then verified if the ratios of the classes of the target variable in each fold were similar (**Figure S5**). The ratio of the classes of the target variable in each fold is almost identical.

2.3 Machine Learning Algorithms

For the different models, we chose to evaluate three different tree-type ML algorithms: Decision Tree (DT), Random Forest (RF), and XGBoost (XGB) [Pedregosa et al., 2011, Chen and Guestrin, 2016]. We chose tree models due to their high interpretability, as we were interested in which features were most important for defining interface residues.

We used the DT as a baseline, as it is less powerful than the ensemble tree-based models. It is a hierarchical model that splits data into branches based on feature thresholds, creating a tree-like structure where each node represents a decision rule, and the leaves represent final predictions. It is prone to overfitting but provides interpretability and ease of use.

RF is an ensemble learning method that constructs multiple decision trees using bootstrapped samples and random subset of features, then aggregates their predictions to improve accuracy and reduce overfitting. This approach enhances generalization by reducing variance of single the single learners, while maintaining interpretability.

XGB is an optimized gradient boosting algorithm that builds trees sequentially, where each tree corrects the errors of the previous one using a weighted approach. We used this method in addition to random forest as it is more powerful for capturing potential complex non-linear patterns in our data.

2.4 Feature Selection and Engineering

Removing unimportant features and engineering new potentially informative features can improve model performance. A further reason to remove certain features is to make our model more interpretable, by removing correlated features, or features that we are not interested in.

RLength: In our preliminary model, Rlength was the most important feature by a wide margin (**Figure S9**). The reason for this is that there is no direct proportion between length and number of interacting residues (**Figure S7**), meaning that there is an inverse proportion between length and the frequency of interacting residues (**Figure S6**). This means that each residue is much more likely to be an interface in a smaller protein and that is why we see the high importance for length. Probably this is a bias in the data and not something related to biological reality. In the dataset, there are only proteins which have at least one interaction site. If we were to look at random proteins, we would probably see that many of them do not have (experimentally described) interaction sites at all. And most probably shorter proteins are more likely to have 0 interaction sites than longer ones, if we would add these proteins to the dataset we would probably see a change in the relationship between length and interface residue frequency. Another source of bias is that longer proteins are generally harder use in X-ray crystallography especially in interaction with another protein, this can lead to longer proteins having less described interaction sites. Since most of the effect of length is likely due to biases and not biological reality we removed it as a feature. Even if there is a biological reason for the relationship, we are more interested in the properties and local context of the interfaces. Normalized length is this feature rescaled, so we removed that as well.

Metadata Features: The dataset description suggests that certain features are not to be treated as input features in our model, so we removed these, namely: aa_ProtPosition, domain, uniprot_id.

Normalized Hydropathy Index: We removed this feature as it contains the exact same information as our label encoding of the amino acids (hydrophobicity_scores), with exception that our feature has a unique value for each amino acid and the hydropathy index has the same value for: Q, N, D and E (otherwise it has unique values for each amino acid as well). Thus our feature has more information than the original (Note that the exact values don't matter for tree-based models, only their order).

Windowed Features: The windowed features are highly correlated to each other between windows. In our final model we used features averaged at window size 9 (WM9) in addition to the unwindowed (simple) scores as this size slightly outperformed every other window size (and window size combination), this was tested with an RF model, evaluating with 5-fold CV and optimizing the hyperparameters for every different window selection. (**Figure 3c**). Based off the preliminary results, some scores were selected for the validation on the test set. The WM3, WM5, WM7 and WM9 scores were selected and compared against the baseline, which contained only the simple scores (not windowed). Furthermore, it was hypothesized that decorrelating the windowed features could yield extra performance because the window features would only encode the scores at certain positions (e.g. WM3 would only encode -1 and +1 positions, WM5 -2 and +2, etc.). Lastly, due to the correlation in the feature importances between the simple scores and the windowed scores, a model containing only WM9 scores (as opposed to a mix of simple and WM9 scores) was also included. Every window includes the windows that are smaller then them and this explains part of the correlation between them. To counteract this from each feature we subtracted the one that is 2 smaller than it, multiplying each by their length (since the numbers are averages, so a window of size 9 has a sum of variable i equal to 9 times the average) and dividing it by two (eg. $W_{i9,decorrelated} = \frac{W_{i9}\times 9 - W_{i7}\times 7}{2}$). Where W_{i9}, W_{i7} are feature i averaged at window size 9 and 7 respectively, In general: $W_{ik,decorrelated} = \frac{W_{ik}\times k - W_{i(k-2)}\times (k-2)}{2}$), where $k \in \{3,5,7,9\}$. This decorrelating step has largely decreased

correlations between windows (**Figure S15**). Since this step only improved the ROC-AUC to a negligible extent we used the normal (without decorrelation) window size 9 for our final model, since it is easier to interpret and interpretability is key for our biological research question.

2.5 Hyperparameter Optimization

After choosing feature sets and some preliminary models, we optimized for the hyperparameters for them. The optimization is key in finding the hyperparameters of the model that perform the best on the given data (tuned and untuned decision tree results shown in **Figure S8**). The hyperparameters mostly relate to model complexity, but also influence other aspects of the model. For this we used Bayesian Optimization (BO) from sci-kit optimize [Head et al., 2021], which is more efficient than Randomized or Exhaustive Grid Search. Each iteration of BO search takes into account the parts of the hyperparameter space that have already been explored and makes an informed choice of the next combination to use. We used 50 iterations for each model, using 5-fold GroupKFold (described above in detail), and optimized for ROC-AUC, because this metric considers the balance between both false positive and true positive rate, making it ideal for evaluating unbalanced datasets.

For the class imbalance methods, we implemented our own hyperparameter search method, since we had to make sure that we use oversampling methods like SMOTE in each iteration of the k-fold CV, instead of using it on the entire dataset before the CV. This is because some of the oversampling approaches create new samples based on several preexisting samples. So, if we first oversample and then do k-fold CV, any given fold will likely contain data that were oversampled based on samples from another fold. This data leakage between folds would highly bias our results, so we implemented our own grid search, where we test all hyperparameter combinations in a given hyperparameter space, using a modified 5-fold CV, where we do an over/under sampling step in each iteration of the k-fold CV. Alternatively, we could have only oversampled in each fold however, that would have led to an underestimation of the performance of these methods, as we would have only oversampled on one fifth of the our data (compared to 4/5 in the case above), which would have likely resulted in worse performance (using more data for oversampling is beneficial). This is why we opted for the computationally more expensive, but more accurate, approach of resampling in every iteration of the CV.

The hyperparameter spaces we used for each model can be found in our GitHub repository as well as the best hyperparameters for all of our models. We have re-optimized the hyperparameters for every different window size combination selection. The optimal hyperparameters for the final random forest model were the following (using WM9 and Simple features):

- Criterion was 'gini
- Max Depth was '5'
- Max Features was 'log2'
- Minimum of Samples per Leaf was '67'
- Number of Estimators was '50'
- Class Weight was 'balanced' (Cost sensitive learning)

2.6 Interpreting Predictions with SHAP

SHAP (Shapley Additive Explanations) was used to assess feature effects and importances by quantifying the contribution of each feature to the model's predictions [Lundberg and Lee, 2017]. To ensure a robust estimation of feature effects and counteract the correlations between our features we utilized interventional feature perturbation, where feature values were systematically altered while keeping the marginal distributions intact, for this we used 10000 background samples randomly sampled from our train set. This method provides a more accurate measure of feature importance by isolating the true impact of each variable, reducing biases that arise from correlated features. We trained a SHAP TreeExplainer on our model and the background examples. Then 2000 test examples (from our test set) were randomly sampled and SHAP values were calculated based on these. SHAP summary plots were generated to visualize the distribution and magnitude of feature impacts across all predictions. We have visualized complex, non-linear and interdependent relationships between features and how they influence the prediction, by SHAP scatter and dependence plots. This approach allowed for a more detailed assessment of how individual features influenced model outcomes, helping us understand which features define interface residues.

2.7 Bootstrapping Scores from the Validation

To obtain an accurate representation of the scores obtained by validating on the test set, we performed bootstrapping with resampling on the predicted scores (n=5000). For this purpose

'resample' was used from the sklearn library [Pedregosa et al., 2011]. To calculate the standard error of the mean (SEM) and the 95% confidence interval (CI) later using the scipy library we needed to know if the distribution of the score was normal. For that end a Kolmogorov-Smirnov normality test [Virtanen et al., 2020] was performed. The SEM and CI were calculated assuming a normal distribution for all scores in the end as there was no significant deviation from the normal distribution (P=.258 - P=1.000).

3 Results

3.1 Machine Learning Algorithm

Figure 3a shows that on CV XGB and RF outperformed the DT algorithm as expected. We also used a DT model without hyperparameter tuning as a baseline, which outperformed the optimized DT by a wide margin. Since XGB and RF performed similarly, we chose to use RF to test window sizes, as it is simpler and requires less optimization than XGB.

3.2 Window Size

As seen in **Figure 3c**, WM9 appears to outperform most other window sizes, though its performance is comparable to the WM3 + WM9 model. Notably, performance is postively correlated with window size. Since including more window sizes often times reduced performance, only one windowed feature was chosen for evaluation (**Figure 3b**). The results show that the model with decorrelated features performed the worst, even worse than the baseline (Simple) feature set. From our preliminary CV results (**Figure S11**) we expected the performance to be around 70% AUC for the WM9 model. The rest of the results do align with our preliminary CV results, but interestingly, the WM9 model performs best when the simple features are excluded. We also see that there is an insignificant performance change between the Simple and Simple + WM3, the Simple + WM7 and Simple + WM9 model, as indicated by the SEM error bars (red).

3.3 Final Model

Our final RF model used data with non-windowed and nine-windowed features to predict PPI interface residues, and was ultimately used to determine feature importance. Its performance on the test data was 0.652 ROC-AUC, 0.193 precision, and 0.634 recall.

3.4 Feature Importance

We extracted feature importance using SHAP values, shown in **Figure 4**. The SHAP results show that the three most important determinants of PPI prediction are windowed PSSMs, hydrophobicity, and surface accessibility.

3.4.1 Windowed PSSM Scores

Several windowed PSSM features show the highest absolute SHAP values, indicating strong influence on model predictions. Notably, windowed PSSMs greatly outnumber non-windowed PSSMs in the list of most important features, highlighting the influence of residue neighborhood conservation, in general higher values averaged PSSM values correspond to lower SHAP values. In particular, hydrophilic residues cysteine (C), serine (S), glutamine (Q), and hydrophobic residue proline (P), exhibit negative SHAP values when their windowed PSSM scores are high.

A closer look at these four features is shown in **Figure 4b**. Residues cysteine (C) and proline (P) show a negative relationship between SHAP values and windowed PSSM. In contrast, residues serine (S) and glutamine (Q) display an initial increase in SHAP value as their windowed PSSM score increases, before their SHAP values drop sharply as PSSM score continues to increase, this pattern can be observed in case of other amino acids as well, namely: R, G, F, N, M, A, T, and Y.(**Figure S14**).

3.4.2 Hydrophobicity

Hydrophobicity scores also show notable influence on PPI interface residue predictions, with low hydrophobicity scores linked to high SHAP values. This means hydrophilic residues are more likely to be predicted as interface residues, contrary to what we expected. To further examine this relationship, we look at **Figure 4c**, where it can be seen that hydrophilic residues generally have higher surface accessibility than hydrophobic residues. In other words, hydrophilic residues are more likely to be on the surface of the protein, which may explain why they are more likely to be predicted as interface residues.

We also note that Arginine, represented by the leftmost line of points in **Figure 4c**, has the highest and most positive impact on interface residue prediction by a large margin compared to other amino acids.

3.4.3 Solvent accessibility

Normalized absolute solvent accessibility is also among the top-ranked predictors. Higher surface accessibility values correlate with positive

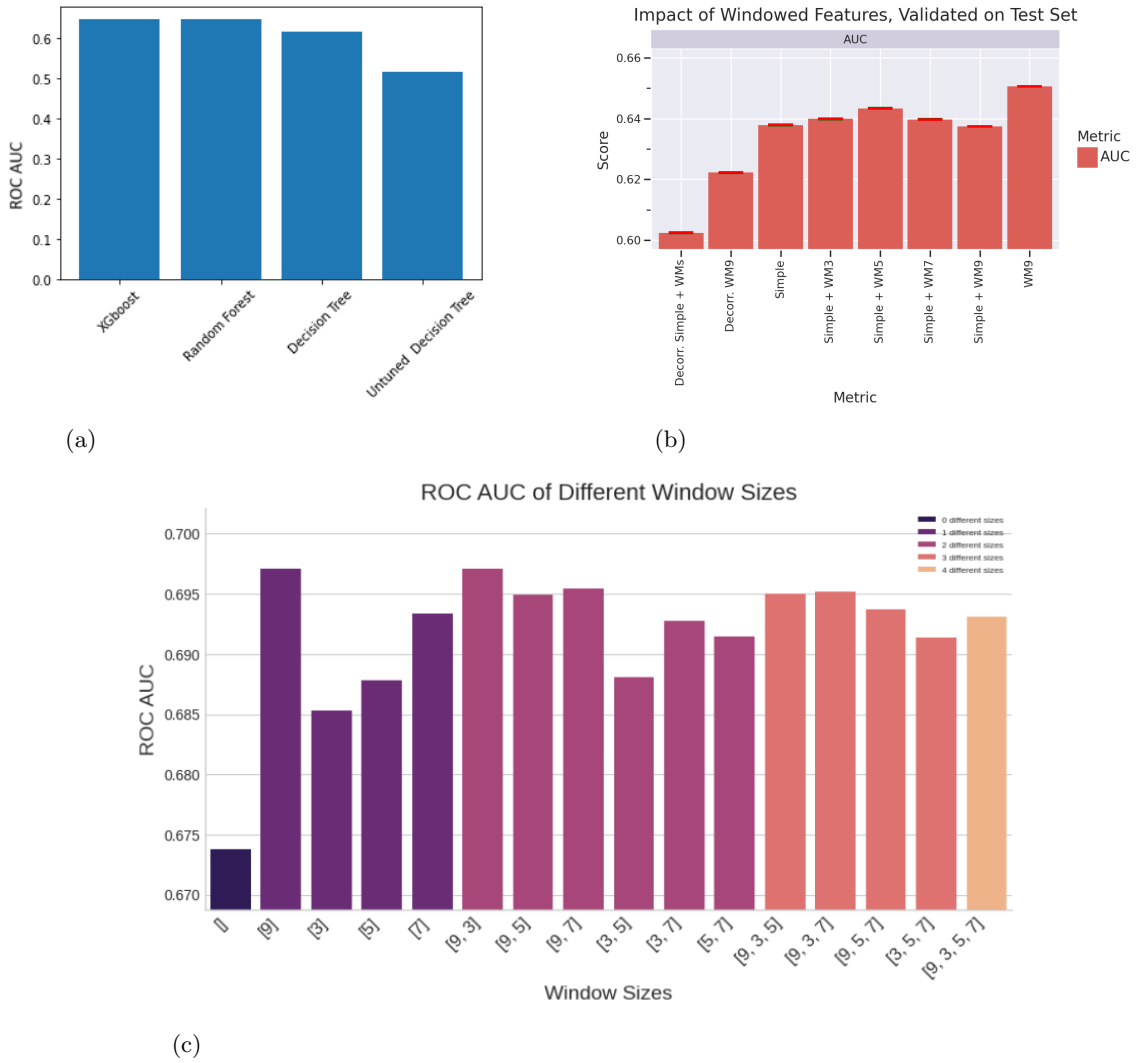


Figure 3: **Model Evaluations** (a) **Model Comparison** using 5-fold CV. (b) **Impact of the Windowed Features**, zoomed in on AUC score, selection based on the figure in c, The red error bars are the standard error of the mean and the green error bars are the 95% confidence interval (barely visible). (c) **Comparison of ROC-AUC Scores Across Different Window Sizes** for random forest during cross validation. Colors represent number of different window sizes used. All models also included non-windowed features (i.e. a window size of 1).

SHAP values, indicating that surface residues are more likely to be part of a PPI interface. This aligns with our expectations, as interface residues must be solvent-accessible in their unbound state to engage in interactions.

4 Discussion

We aimed to predict PPI interfaces using a machine learning approach, focusing on the effects of different window sizes and feature importance. Our main models had AUC and F1 scores of 0.695, 0.146 (9WM), 0.637, 0.147 (simple + 9WM) 0.622, 0.096 (decorrelated all WMs). It is relevant to compare our results to [Stringer et al., 2022] as they used the same dataset for testing their models as we did (except for the fact that they used a bigger subsample of the same dataset). They achieved AUC of 0.755 and F1 of 0.339 with *esnep_p*, their PPI specific neural network. Recently, a newer version of the same model was published incorporating protein language models [Thomas et al., 2025], achieving AUC of 0.8. Our model's lower performance is expected as the other methods use more complex models (neural networks), that also directly consider sequential information (RNNs and CNNs). They also trained on a bigger dataset (over 6800 proteins, compared to our train set of around 180 proteins). Our F1 score in particular underperformed theirs, this could be because we optimized for AUC during the hyperparameter optimization.

Different window sizes were used simultaneously, while including multiple window sizes may add information, it also introduces correlations between features, making interpretation more

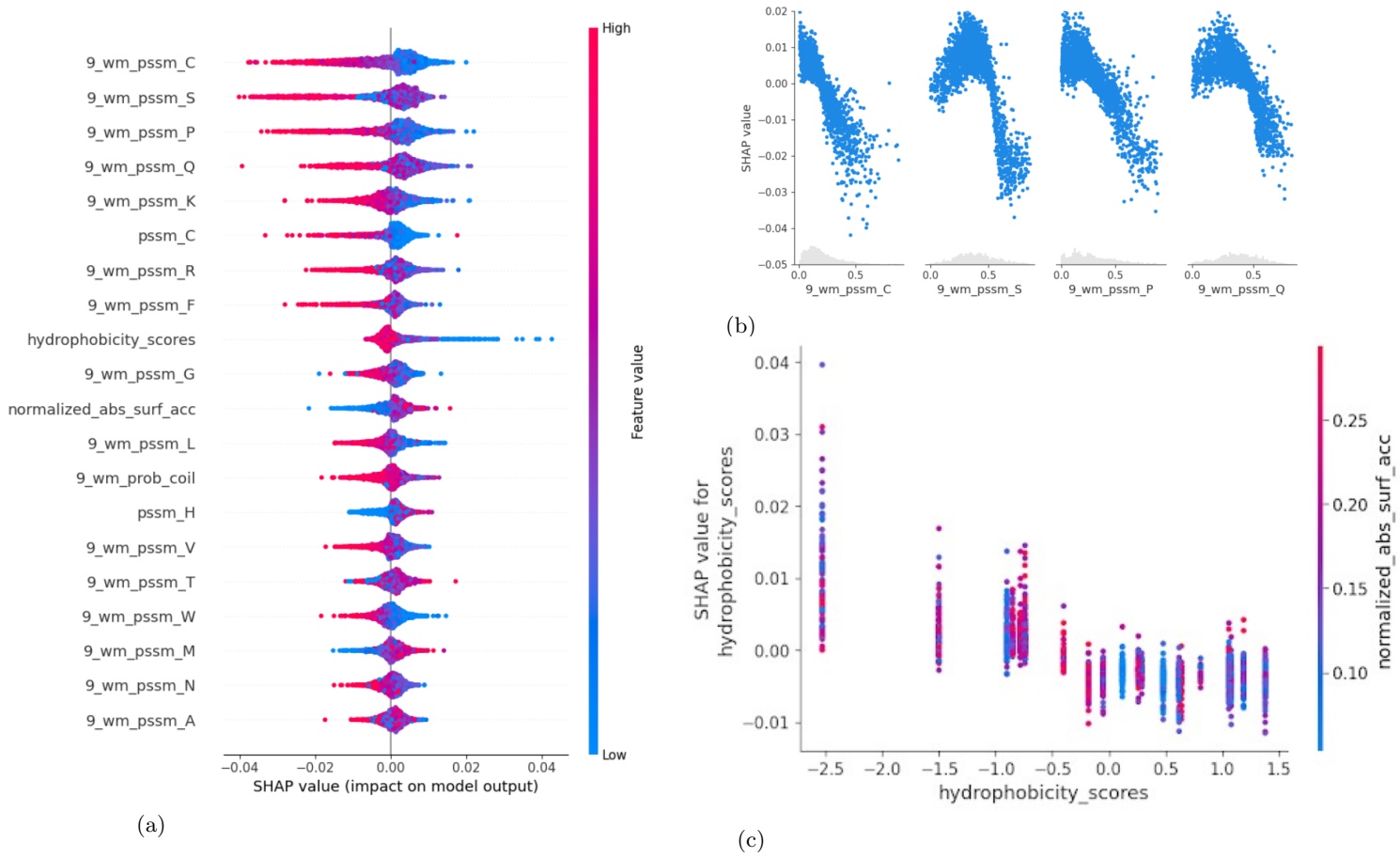


Figure 4: (a) **SHAP Summary Plot** showing feature importance, where each point represents an individual prediction. Features are ranked by importance, with color indicating feature value (red = high, blue = low). Higher SHAP values indicate a stronger positive impact on the model's interface residue prediction, and vice versa. (b) **Top Four Features** Each point is an individual prediction. Higher SHAP values indicate a stronger positive impact on the model's interface residue prediction, and vice versa. Underneath the blue points is the grey marginal distribution. (c) **Hydrophobicity SHAP Plot**, with each point representing a single prediction. Each amino acid is defined by its hydrophobicity score, as represented by the vertical lines of points. The corresponding amino acids from left to right are R, K, D, Q, N, E, H, S, T, P, Y, C, G, A, M, W, L, V, F, I. Hydrophilic residues have negative scores, while hydrophobic residues have positive scores, with higher absolute values indicating stronger hydrophobicity/philicity. The colors indicate absolute surface accessibility (ASA), with warmer colors corresponding to higher ASA.

challenging. In our final evaluation we included a model that was only trained on the WM9 features, and it seemed to outperform all other feature sets (**Figure 3b**). This is expected from the perspective that the WM9 features hold the most information because they include the largest distance away from the sample residue in their mean. However, when we included the simple scores, the performance dropped significantly. This is unexpected because if those features are less informative then the windowed features, you would expect the model to mostly ignore them. Our hypothesis is that, by adding the simple scores as well, the principle that makes RFs potent was circumvented. Namely, instead of a limited and discriminative feature space per DT in the RF, the model is more likely to have a limited and non-discriminative feature space because the simple and WM9 scores are so correlated. For example, if a model gets assigned the PSSM_G scores and WM9_PSSM_G scores, it has two features which mostly contain similar information, and by extension, less overall information. Another surprising result was that decorrelating the features did not solve this issue, as these models performed even worse than the normal features, despite performing better during the hyperparameter optimization than the normal features (around 70% AUC). That likely rules out the model being the issue, which means that possibly the model just generalizes poorly to the test data (of which the features were also decorrelated). This could be due to the distribution of the data between the train and test or because the positional information gained from the decorrelation makes the model

more prone to overfitting.

For our feature importances we looked at the model with the Simple + WM9 feature set, which was not the optimal model in the end, but we selected it as our final model based on CV. We examined SHAP values of the most important features to determine how they influence the prediction. Interestingly, hydrophobicity was found to be negatively impact the interface formation, which contrasts with traditional expectations and previous research [Yan et al., 2008, Xue et al., 2015], where hydrophobic residues were found to have high interface propensities. One possible explanation is that polar residues are more likely to be exposed on the protein surface, making them more accessible for interactions, while hydrophobic residues are more likely to be buried in the core of the protein. Interestingly, when we controlled for relative surface accessibility the pattern remained the same, so even among the surface accessible residues, which are already more likely to be on the surface, high hydrophobicity still corresponds to lower interface prediction, which is contrary to previous works [Yan et al., 2008], where surface exposed hydrophobic residues correspond to high interface propensities. Probably, our model could not capture this complex interaction between the two features. When examining the SHAP values for individual residues, we observed that they behave differently across different ranges. The general pattern regarding 9 WM PSSMs is that the very high values correspond to negative SHAPs. This could be due to the fact that high values mean high conservation of the given amino acid in all nine position, meaning low sequence diversity in that area. However in a certain range of values certain amino acid PSSMs have positive contribution to interface prediction. These amino acids are: S, Q, R, G, F, N, M, A, T, and Y, with serine (S) and threonine (T) being particularly interesting, since their contribution could be due to their role as phosphorylation sites, since these sites were found to be enriched in PPI interfaces (**Figure 4**) [Nishi et al., 2011], but the rest of these amino acids also have higher propensities according to previous literature [Yan et al., 2008]. Other features investigated corresponded to lower importance according to SHAP values like the secondary structure, however beta sheets have positive impact, which is in line with previous findings [Yan et al., 2008], normalized absolute solvent accessibility was positively correlated with the SHAP values, which indicates that they are likely found in the PPI interface. This aligns with our expectations and with previous literature [Capel et al., 2022], since interface residues need to be exposed to the solvent in their unbound state to take part in interactions.

4.1 Limitations

While our model demonstrates strong performance in identifying key residues contributing to PPI interfaces, there were limitations that need to be considered. The dataset used for training may introduce biases that do not generalize across all protein types. Additionally, the negative correlation of hydrophobicity with interface formation requires further investigation, as it contradicts previous knowledge. Experimental validation of these findings would be necessary to confirm the biological relevance of the patterns detected by our model. Moreover, exploring models that incorporate sequence information, such as Recurrent Neural Networks (RNN), transformers, or Convolutional Neural Networks (CNN), could improve predictive performance and provide deeper insights into PPI interfaces, though their interpretability poses different challenges. Our model was not directly aware of the sequence and only predicted on residue level. Sequential machine learning (Hidden Markov Models) and deep learning (CNN, RNN) methods may perform better on this and similar tasks, as seen in [Stringer et al., 2022]. Finally, using protein language models like ProtT5-XL [Elnaggar et al., 2022] can lead to even further improvements in predictions for this and similar tasks [Thomas et al., 2025], as this models can translate long and complex protein patterns into short biologically relevant embeddings.

5 Conclusion

In this study, we used residue-level information to characterize PPI sites using DT, RF, XGB ML models. Our findings indicate that removing highly correlated and redundant features, particularly in terms of window sizes improved model performance, with a feature set containing correlated WM9 and unwindowed features showcasing the best results. Our final RF model with an AUC of 0.652, however, was found to perform significantly worse than the current state of the art, underscoring the critical role of data quantity and model complexity in achieving high predictive capability. Despite these limitations, our research identified key sequence-derived features - windowed PSSM scores of residues C, S, P, surface accessibility, and hydrophobicity contributing to interaction site prediction with the unexpected observation that higher hydrophobicity correlated with lower prediction scores.

6 Acknowledgements and Author Contributions

AI was consulted for: debugging code or amending tricky code (like plots or Pandas subsets), help with brainstorming about the interpretation of the results, as a tool for finding some relevant research papers, and improving the flow of some long/complex sentences.

AI was not used for: Writing large chunks of code, project architecture, writing multiple sentences/paragraphs in the report.

The distribution of work for the project largely followed like this: Project architecture (Gyula and Jacob), Data preprocessing, over/undersampling code (Shabnam), Model training, optimization, feature/model selection code (Gyula and Jacob), Finding relevant papers (Ana and Gyula), Creating BioRender schematics (Jasmine)

The distribution of writing the report largely followed like this: Abstract (Jasmine), Introduction (Ana), Methods (Gyula, Shabnam, Jasmine, Jacob), Results (Jasmine, Jacob), Discussion (Ana, Gyula, Jacob), Conclusion (Shabnam), Proofreading/Final touches (Everyone)

7 Availability

Our scripts are available on the following GitHub repository, managed by Jacob Reaves.

References

[Albahra et al., 2023] Albahra, S., Gorbett, T., Robertson, S., D'Aleo, G., Kumar, S. V. S., Ockunzzi, S., Lallo, D., Hu, B., and Rashidi, H. H. (2023). Artificial intelligence and machine learning overview in pathology & laboratory medicine: A general review of data preprocessing and basic supervised concepts. In *Seminars in Diagnostic Pathology*, volume 40, pages 71–87. Elsevier.

[Ansari et al., 2024] Ansari, S., Nassif, A. B., Mahmoud, S., Majzoub, S., Almajali, E., Jarndal, A., Bonny, T., Alnajjar, K. A., and Hussain, A. (2024). Impact of outliers on regression and classification models: An empirical analysis. In *2024 17th International Conference on Development in eSystem Engineering (DeSE)*, pages 211–218. IEEE.

[Capel et al., 2022] Capel, H., Feenstra, K. A., and Abeln, S. (2022). Multi-task learning to leverage partially annotated data for ppi interface prediction. *Scientific Reports*, 12(1):10487.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

[Cheng et al., 2021] Cheng, F., Zhao, J., Wang, Y., Lu, W., Liu, Z., Zhou, Y., Martin, W. R., Wang, R., Huang, J., Hao, T., Yue, H., Ma, J., Hou, Y., Castrillon, J. A., Fang, J., Lathia, J. D., Keri, R. A., Lightstone, F. C., Antman, E. M., Rabadan, R., Hill, D. E., Eng, C., Vidal, M., and Loscalzo, J. (2021). Comprehensive characterization of protein-protein interactions perturbed by disease mutations. *Nat. Genet.*, 53(3):342–353.

[Eisenberg et al., 1984] Eisenberg, D., Schwarz, E., Komaromy, M., and Wall, R. (1984). Analysis of membrane and surface protein sequences with the hydrophobic moment plot. *Journal of molecular biology*, 179(1):125–142.

[Elnaggar et al., 2022] Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., and Rost, B. (2022). ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):7112–7127.

[Hasan et al., 2021] Hasan, M. K., Alam, M. A., Roy, S., Dutta, A., Jawad, M. T., and Das, S. (2021). Missing value imputation affects the performance of machine learning: A review and analysis of the literature (2010–2021). *Informatics in Medicine Unlocked*, 27:100799.

[Head et al., 2021] Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. (2021). scikit-optimize/scikit-optimize (v0.9.0).

[Lemaître et al., 2017] Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

[Lundberg and Lee, 2017] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

[Nishi et al., 2011] Nishi, H., Hashimoto, K., and Panchenko, A. R. (2011). Phosphorylation in protein-protein binding: effect on stability and function. *Structure*, 19(12):1807–1815.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Rego et al., 2021] Rego, N. B., Xi, E., and Patel, A. J. (2021). Identifying hydrophobic protein patches to inform protein interaction interfaces. *Proceedings of the National Academy of Sciences*, 118(6):e2018234118.

[Sikić et al., 2009] Sikić, M., Tomić, S., and Vlahovicek, K. (2009). Prediction of protein-protein interaction sites in sequences and 3d structures by random forests. *PLoS Computational Biology*, 5(1):e1000278.

[Stringer et al., 2022] Stringer, B., de Ferrante, H., Abeln, S., Heringa, J., Feenstra, K. A., and Haydarlou, R. (2022). Pipenn: protein interface prediction from sequence with an ensemble of neural nets. *Bioinformatics*, 38(8):2111–2118.

[Thomas et al., 2025] Thomas, D. P. G., Garcia Fernandez, C. M., Haydarlou, R., and Feenstra, K. A. (2025). PIPENN-EMB ensemble net and protein embeddings generalise protein interface prediction beyond homology. *Sci. Rep.*, 15(1):4391.

[Virtanen et al., 2020] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

[Werner de Vargas et al., 2023] Werner de Vargas, V., Schneider Aranda, J. A., dos Santos Costa, R., da Silva Pereira, P. R., and Victória Barbosa, J. L. (2023). Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. *Knowledge and information systems*, 65(1):31–57.

[Xue et al., 2015] Xue, L. C., Dobbs, D., Bonvin, A. M., and Honavar, V. (2015). Computational prediction of protein interfaces: A review of data driven methods. *FEBS Letters*, 589(23):3516–3526.

[Yan et al., 2008] Yan, C., Wu, F., Jernigan, R. L., Dobbs, D., and Honavar, V. (2008). Characterization of protein-protein interfaces. *The Protein Journal*, 27(1):59–70.

[Yang et al., 2013] Yang, J., Roy, A., and Zhang, Y. (2013). BioLiP: a semi-manually curated database for biologically relevant ligand-protein interactions. *Nucleic Acids Res.*, 41(Database issue):D1096–103.

[Zhou et al., 2016] Zhou, M., Li, Q., and Wang, R. (2016). Current experimental methods for characterizing protein-protein interactions. *ChemMedChem*, 11(8):738–756.

8 Supporting Information

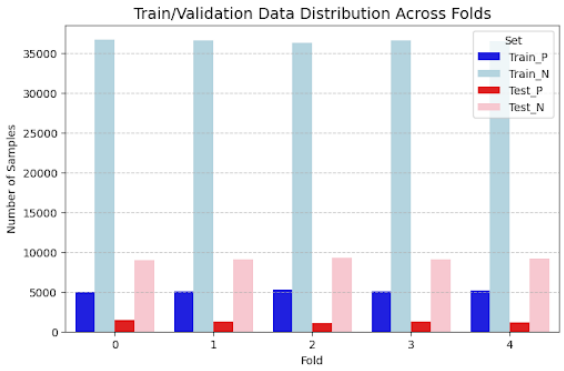


Figure 5: Similar ratio of target classes in each fold during 5-fold CV

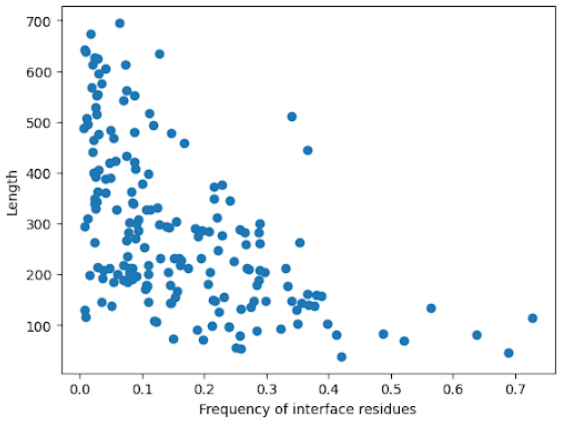


Figure 6: Frequency of number of interface amino acids per protein vs. length

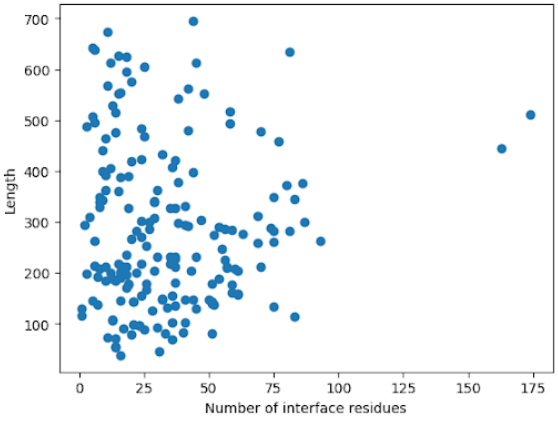


Figure 7: Number of interface amino acids per protein vs. length

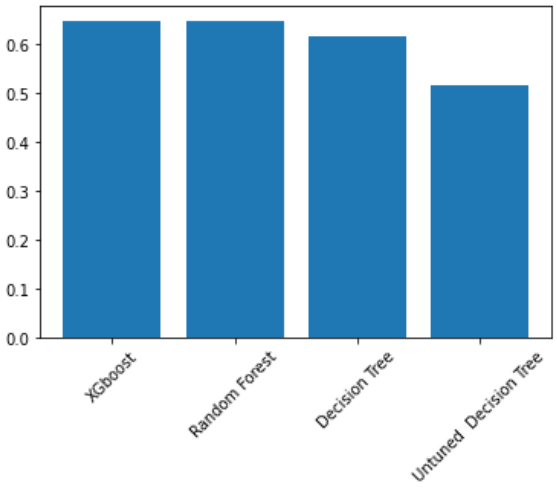


Figure 8: ROC-AUC Comparisons of 4 different models

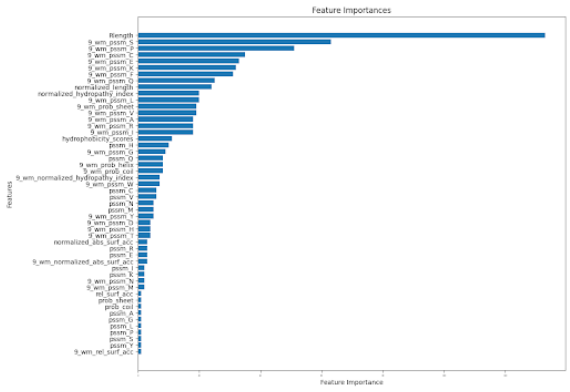


Figure 9: Importances from preliminary random forest model

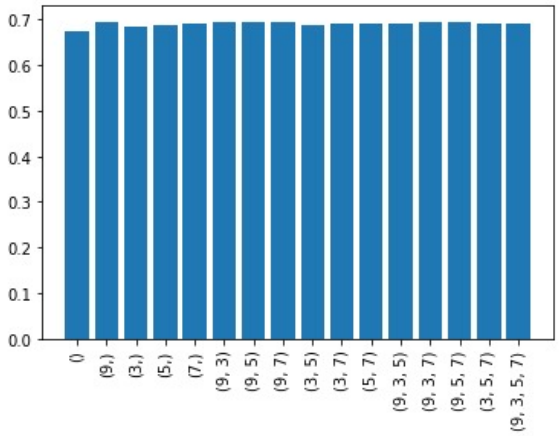


Figure 10: ROC AUCs using different window size combinations on validation set

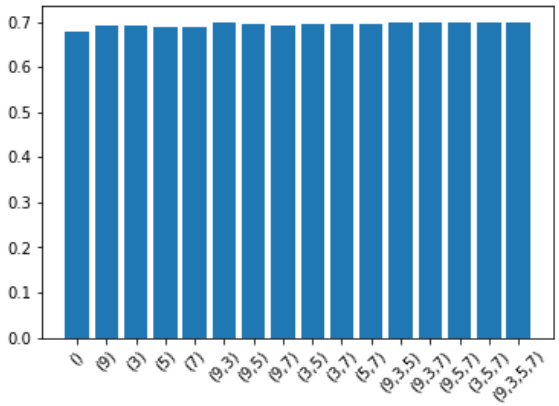


Figure 11: ROC-AUC using different decorrelated window size combinations on validation set

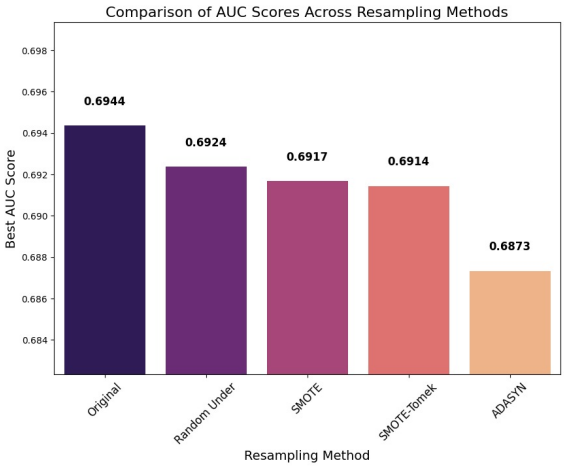


Figure 12: Resampling methods vs. cost sensitive learning (original) ROC-AUC scores

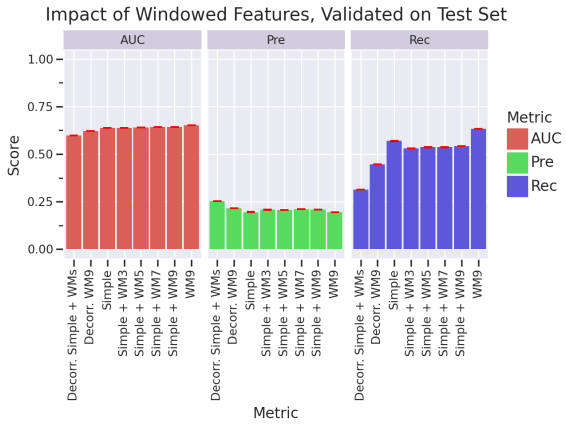


Figure 13: Unzoomed version of the windowed feature performance on the test set. Pre: Precision, Rec: Recall

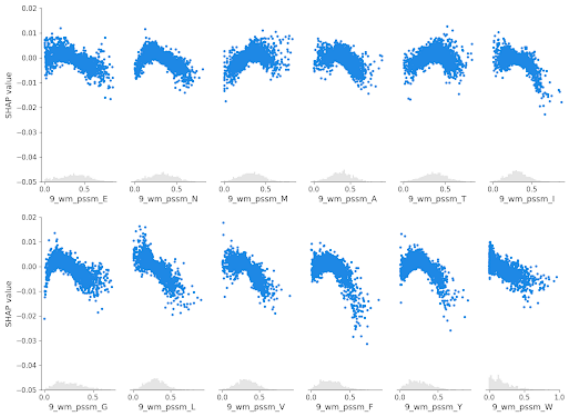
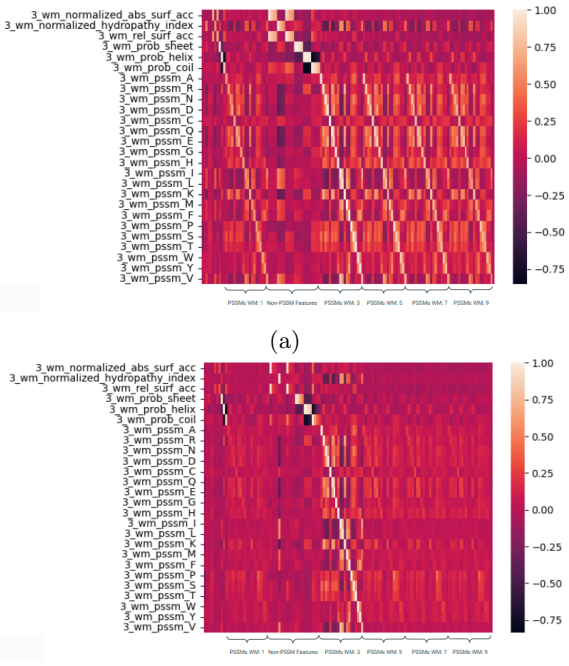


Figure 14: Scatter plot of 9 windowed mean feature values vs. SHAP values, non-linear relationships are apparent



(a)

(b)

Figure 15: A, original correlations between window size 3 and the rest, B, decorrelated correlations between window size 3 and rest of the features