Meta-DPI: A Computational Metamethod for Predicting Protein-Protein Interfaces

Thesis Submitted in Partial Fulfillment
of the Requirements
of the Jay and Jeanie Schottenstein Honors Program

Yeshiva College
Yeshiva University
May 2020

Mordechai A. Walder

Mentor: Dr. Rajalakshmi Viswanathan, Professor and Co-chair
Department of Chemistry

Table of Contents

Abstract

Protein-protein interactions (PPIs) regulate many biological processes that are integral for the survival, function, growth, and evolution of organisms. Determination of the amino acid residues at these interaction sites, or interfaces, enhances our understanding of the molecular mechanisms by which proteins carry out their functions, and it facilitates the development of therapeutics by identifying critical sites for disrupting protein function. Due to the time-consuming and costly nature of experimental approaches for identifying interacting residues, computational methods are employed to streamline the process. This work describes the development of meta-DPI, a computational metamethod for predicting protein-protein interfaces that integrates the orthogonal prediction methods **D**ockPred, **P**redUs 2.0, and **I**SPRED4. Cross-validation experiments on two datasets containing a total of 223 protein complexes illustrate that meta-DPI significantly outperforms each of its three constituent methods (DockPred, PredUs 2.0, and ISPRED4) as measured by both single-threshold and threshold-free evaluation metrics. The enhanced predictive power of meta-DPI demonstrates how metamethods create improved interface predictors, which in turn, enable molecular mechanisms to be understood more thoroughly and drug targets to be identified more efficiently.

I. Introduction

A. Protein-Protein Interactions

Proteins have a diverse range of functions in living organisms. Different proteins catalyze biochemical reactions, function as hormones that maintain homeostasis, transport molecules around the body and through cell membranes, and are an essential part of the immune system's response to fight infection. Proteins are polymers consisting of amino acid subunits linked together by peptide bonds. There are twenty different amino acids typically utilized to make proteins in cells, each with a common backbone and a variable side chain. The unique amino acid sequence of a protein is known as its primary structure, and the hydrogen bonds between atoms of a protein's backbone give rise to its secondary structure elements, like alpha helices and beta sheets. The tertiary structure of a protein arises due to interactions between amino acid side chains that yield the folding patterns that define a protein's three-dimensional structure. The amino acid residues located on a protein's surface are able to interact with other molecules, including other proteins, while internal residues are not able to interact with other molecules. A

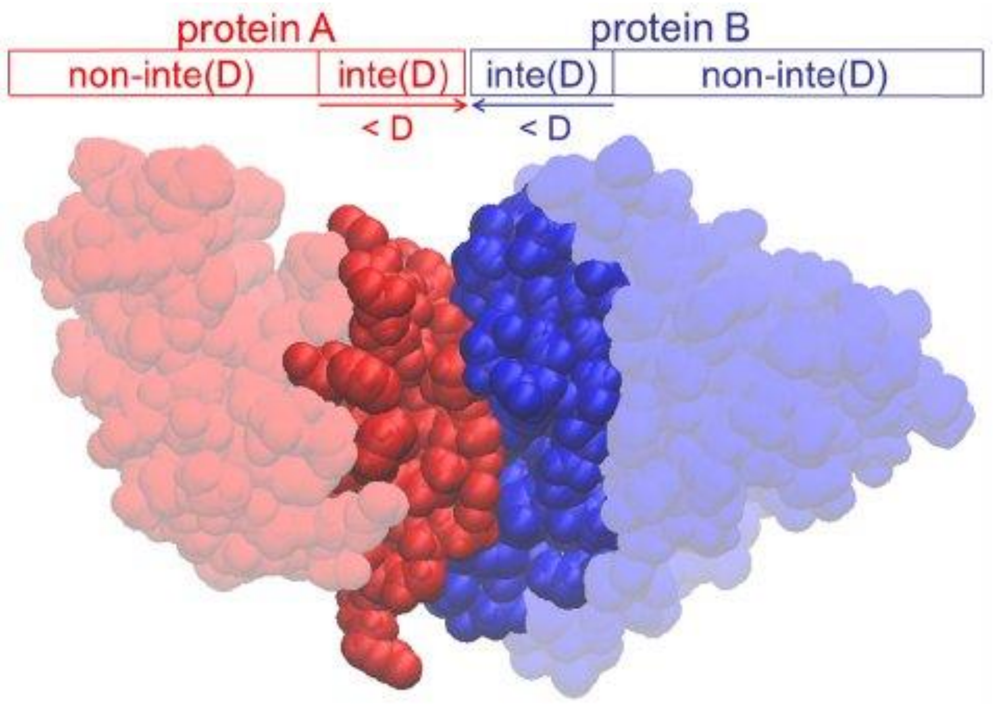


Fig 1. Definition of the interface region inte(D) of a protein-protein complex. The dark red interface region inte(D) of protein A is defined as the continuous protein surface region comprising all residues with at least one atom within a distance D from protein B. All other residues of protein A belong to the complementary light red region, non-inte(D). (Based on [1])

generic protein-protein interaction is illustrated in Figure 1, in which two proteins (A and B) interact with each other to form a structured protein complex. The binding interface of protein A is the cluster of its surface residues that participate in binding with protein B, and a residue is classified as being at the interface if one of its atoms is within a threshold distance D (usually 4.5-8 Å [2]) from the binding partner [1].

Knowledge of the partners with which a protein forms a complex is key to understanding a protein's function and for precise characterization of broad protein interaction networks. Identifying a protein's interface residues is important in determining the molecular mechanism by which a protein executes its function [3]. Modification of the residues at the binding interface can disrupt or promote a PPI, and knowing which residues are interfacial is critical to discerning how mutations affect a PPI. Molecular targets serve as the framework of modern drug discovery, which aims to identify therapeutic agents that can selectively modulate disease-specific molecular pathways [4]. PPIs play a central role in the progression of many disease states, which is why they have become an emerging class of targets for drug discovery. Thus, determining the interfacial residues of PPIs is a critical step in the process of identifying potential drug targets.

One such example of a therapeutic drug that targets a PPI is ipilimumab, which is a monoclonal antibody that targets the CTLA-4:B7 (receptor: ligand) PPI. T cell homeostasis is maintained by the homologous ($\approx$ 30% sequence identity) costimulatory receptors CTLA-4 and CD28 on the cell's surface, and each of them binds to the B7-1 and B7-2 ligands that are on the surface of antigen-presenting cells [5]. Upon ligand binding, CD28 sends signals that increase T cell proliferation and cytokine secretion, while CTLA-4 sends signals that downregulate T cell response. The FG loop ($_{99}$MYPPPY$_{104}$) on CTLA-4 contributes approximately 80% of the interfacial contacts and 90% of the binding energy with B7 ligands. The FG loop is part of

CTLA-4's P3 epitope that binds to ipilimumab, which indicates that direct steric competition allows ipilimumab to inhibit B7 ligands from binding to CTLA-4 (Figure 2). Through inhibition of the CTLA-4:B7 PPI, ipilimumab removes the downregulation of cytotoxic T cells, which boosts the immune response against cancerous cells. Ipilimumab has been approved to treat metastatic melanoma, and it is currently being studied in clinical trials to treat other types of cancer. Ipilimumab is a prime example of how knowing interfacial residues of a PPI can lead to the development of a therapeutic drug that targets the PPI.

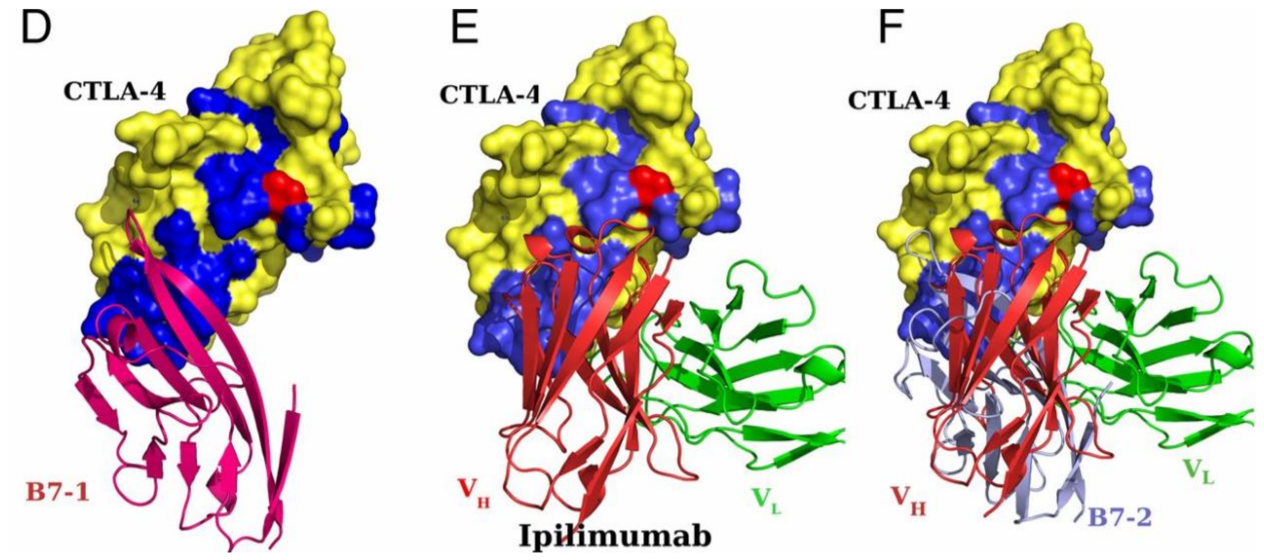


Fig 2. Mechanism of ipilimumab targeting PPI. (*D*) Mode of B7-1 (pink) interaction with the MYPPPY-loop surface. (*E*) Mode of CTLA–4 and ipilimumab interactions. (*F*) Superposition of the CTLA-4:B7-2 and CTLA-4:ipilimumab structures, based on the CTLA-4 component in each complex, is presented. These superpositions indicate that ipilimumab and the B7 ligands compete for overlapping binding surfaces on CTLA-4. (Based on [5])

B. Experimental Approaches for Identifying Protein-Protein Interfaces

Commonly used low-throughput methods for determining PPI residues include X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy. X-ray crystallography determines the structure of a protein complex based on the diffraction pattern generated from passing X-rays through a crystallized sample. Thus, it requires a high-quality crystal sample to specify the complex structure with sufficient resolution. Since integral membrane proteins and large dynamic complexes are effectively impossible to crystallize, their structures cannot be determined by X-ray crystallography. NMR spectroscopy does not require that samples be crystallized, but the large amount of sample required limits NMR to only being able to determine the structure of small protein complexes. Cryo-electron microscopy does not necessitate a large amount of sample or protein crystallization, so it has become the dominant technique for structural determination of protein complexes over the past few years as it allowed previously inaccessible proteins to be structurally understood [6]. However, due to the low-throughput and costly nature of these experimental approaches, computational prediction methods are employed to streamline the process of identifying the interfacial residues of PPIs.

C. Computational Methods for Predicting Protein-Protein Interfaces

The two main strategies implemented in computational prediction methods are intrinsic based approaches and template-based approaches. Intrinsic-based methods train machine learning algorithms on a dataset of experimentally determined PPIs in order to create a model that relates sequence and structural features with the likelihood for residues to be at the interface. Sequence features include hydrophobicity, amino acid interface propensity, physico-chemical properties, and evolutionary conservation, and structural features include secondary structure, solvent-accessible surface area, and geometric shape. As input, intrinsic-based predictors plug a query protein's sequence and structural features into their models, and as output, they assign interface prediction scores to each query protein residue. While intrinsic-based methods have been steadily enhanced over the past 20 years, their future improvement is limited because further combination of existing features and classifiers has little impact on performance [2].

Template-based methods exploit the fact that binding interfaces are conserved among homologous and structurally similar complexes. Homologous template-based predictors build a multiple sequence alignment (MSA) of a query protein to identify the query's homologues with a known complex structure. The homologues' interfaces are mapped onto the query protein, and the residues are assigned interface prediction scores according to that mapping. Structural neighbor-based predictors find proteins with a known complex structure and have a globally similar fold (neighbors) to the query protein. The structural neighbors' interfaces are mapped onto the query protein, and the residues are assigned interface prediction scores according to that mapping. While template-based methods are highly effective for query proteins that have close homologues or neighbors with a known complex structure, their performance is limited by the relatively small number of resolved complex structures [2].

D. Metamethods and meta-DPI

To overcome the limitations of intrinsic and template-based methods, metamethods that integrate orthogonal (based on different non-overlapping features) predictors can be developed to enhance prediction performance. Meta-PPISP is one such metamethod that combined the predictors cons-PPISP [7], Promate [8], and PINUP [9] through linear regression analysis [10]. The construction of meta-PPISP was not ideal because it only combined complementary intrinsic-based approaches, and it did not combine a template-based approach with an intrinsic-based approach. Additionally, it employed linear regression analysis for method combination, instead of using logistic regression analysis, which is more effective for discrete categorical data like a residue's interfacial score. In order to create a robust metamethod with enhanced prediction performance, I developed meta-DPI, which implements logistic regression analysis to integrate the orthologous interface predictors DockPred (docking-based), PredUs 2.0 (template-based), and ISPRED4 (intrinsic-based).

II. Background Methodology

A. DockPred

It has previously been shown [11] that substrate and non-substrate small organic molecules have a tendency to bind to similar, energetically favorable sites on a target protein ("sticky" sites) regardless of their relevance to it. DockPred was created to test the hypothesis that proteins have a generic interaction site at which cognate and non-cognate ligands bind, just as it had been observed for small molecule "sticky" sites. The success of DockPred demonstrated that non-cognate ligands preferentially bind to the cognate binding site of a target protein [12].

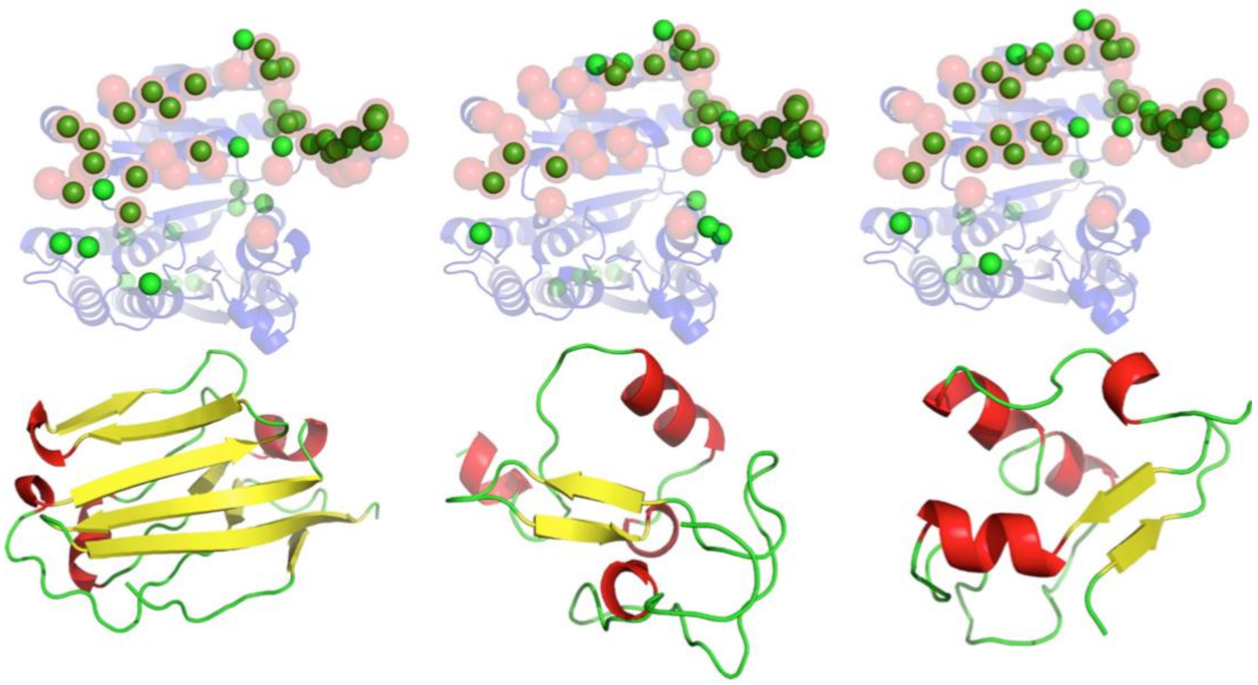


Fig 3. Binding supersite of 1cnz.A. Three non-cognate ligands (lower row, from left to right, PDB codes: 2jjs.C, 2v86.A, 3h33.A) that share no detectable sequence or structure similarity to the cognate ligand, are docked extensively on the surface of the receptor (upper row, 1cnz.A). In the upper row, ribbon model in transparent blue shows the receptor structure. The annotated functional site in the receptor is shown using red transparent spheres for the Cα atoms. The predicted functional site residues, as defined by the corresponding ligand probes underneath, is shown using green spheres for the Cα atoms. (Based on [12])

There were two datasets of query proteins used in the development of DockPred. One dataset contained 108 proteins from the Docking Benchmark (DB) database, and the other contained 133 different proteins from the NOX database [13, 14]. 50 DB proteins and 41 NOX proteins belong to the top 12 CATH superfamilies (Table 1). The DB database was constructed to have a benchmark set of proteins on which to compare the efficacy of docking algorithms [14]. The NOX database was constructed to develop a support vector machine algorithm that distinguishes between obligate, non-obligate and crystal packing interactions [13]. The DB and NOX databases contain 230 and 243 non-redundant protein-protein complex structures along with the structures of their unbound components, respectively. For DockPred, the Contacts of Structural Units (CSU) program was used to define interface residues from the complex structures of the query proteins [15]. If any atom in a query residue was within 3.5Å of an atom in the query's partner protein in the complexed structure, the residue was considered to be at the binding interface. This yielded the experimentally annotated interface residues for each query protein in the dataset.

Table 1. [12] Number of Proteins in Each Dataset Belonging to Each CATH Superfamily

Predominant Fold (CATH Superfamily)	DB Dataset	NOX Dataset
Rossman Fold	7	20
Immunoglobulin Like	33	7
TIM Barrel	1	8
Four Helix	-	2
Trefoil, Acidic Fibroblast Growth Factor	1	2
Alpha-beta Plaits	1	-
OB Fold	3	1
Jelly Roll	3	-
Globin Like	1	-
Alpha-beta Barrel	-	1

A group of 13 non-cognate ligand probes were chosen to be used for DockPred based on their lack of sequence similarity to known cognate ligands of the query proteins. The two docking programs ZDOCK and GRAMM were used to generate 2000 docked complexes for each uncomplexed query protein with each of the 13 non-cognate ligand probes. The CSU program was then used to analyze each of the docked complexes to identify the residues at the interface of each complex. A residue at the *i*th position of the query protein in the *k*th docked complex was denoted as R_{ik}. $I(R_{ik}) = 0$ for residues not at the interface of a docked complex, and $I(R_{ik}) = 1$ for residues at the interface of a docked complex. For each residue at position *i* in the query protein, a Residue Interface Frequency (RIF) was calculated by summing over all docked complexes according to the formula

$$N_i = \sum_{k=1}^{2000} I(R_{ik}).$$

The top ranking residues with the largest N_i values of a query protein were considered to be the residues predicted to be at the binding interface [12].

B. PredUs 2.0

The first version of PredUs, which was developed in 2011, made interface predictions for a query protein based on the known binding interfaces of the query's structural neighbors. Two proteins are considered to be structural neighbors if their three-dimensional structures are similar. The secondary structure elements of two proteins can be similar without being composed of the same amino acids, so a prediction based only on structural similarity does not consider the query amino acids' tendency to participate in binding. PredUs 2.0 was created in 2015 to address the flaw in making a structural template-based prediction alone. Using a Bayesian approach, PredUs 2.0 combines an amino acid interface propensity score with the template-based score of PredUs [3].

The original PredUs program used the structural alignment program Ska [16] to identify a query protein's structural neighbors. Protein structural distance (PSD) is a measurement that quantifies the structural similarity between two proteins. It is calculated by superposing the two proteins' structures in a manner that minimizes the root-mean-square deviation of the amino acids' alpha carbons [17]. PredUs employed a PSD cutoff of 0.6 so that close and remote structural neighbors could be found. Neighbors that have a known complex structure were retained and ranked according to their structural alignment score. Using the program cd-hit [18], neighbors with a sequence similarity larger than 40% were grouped together, and only the protein with the higher PSD was kept. PredUs used the transformation relating the neighbor to the query protein to place the neighbor's binding partner in the query's coordinate system. If the neighbor's binding partner was within 5Å of a query residue's heavy atom (C, N, O), PredUs incremented the residue's contact frequency score weighted by the PSD score between the neighbor and query [19]. After the transformation process was completed for every structural neighbor that was retained, each query residue has a total contact frequency score that sums the weighted scores from all the transformations.

PredUs used a support vector machine (SVM) algorithm to generate its template-based prediction score. Each residue r_i on the query's surface was combined with the 14 closest surface residues to form r_i's surface patch. A 31-element profile was assigned to the patch, and it was composed of each residue's contact frequency score and solvent accessible surface area (ASA), as well as the highest contact frequency score in the protein. The SVM mapped each patch profile to vectors in high-dimensional space, and it created a hyperplane that separated the vectors corresponding to interface residues from the vectors corresponding to non-interface residues. PredUs calculated an interfacial score for each residue based on its profile vector's distance above or below the SVM hyperplane [19].

PredUs 2.0 calculated the interface propensity of each residue type *r* using a set of 2,766 heterodimeric complexes that had less than 40% sequence redundancy. Interface propensity was expressed as the relative ASA (RASA) contribution from residues of type *r* in protein-protein interfaces relative to their RASA contribution to the whole protein surface:

$$Propensity(r) = \frac{\left(RASA_r^{interface}/RASA_{all}^{interface}\right)}{\left(RASA_r^{surface}/RASA_{all}^{surface}\right)}$$

$RASA_{rX}$ represents the sum of RASAs of all type *r* residues with characteristic X (interface or surface) in all proteins from the set of heterodimeric complexes. $RASA_{allX}$ represents the sum of RASAs of all residue types with characteristic X in all proteins from the dataset. RASA of protein residue is defined as the residue's ASA as part of the protein, normalized by the residue's area as part of an ALA-r-ALA tripeptide. Residues were considered to be at a protein's surface if they had a RASA value of at least 0.05.

Since a residue's interface propensity can vary with its RASA value, PredUs 2.0 calculated a weighted interface propensity score based on RASA values. The weighted propensity (WP) of a query protein residue r_i of type *r* was calculated as follows:

$$WP(r_i) = Propensity(r) \times P_r(I|RASA)$$

The term $P_r(I|RASA)$ represents the probability that a type *r* residue is at the interface, given its RASA value. Two scores were calculated for the surface patch associated with residue r_i; one (WPA) was the average of the WPs of the patch residues, and the other (JP) was the joint probability for patch residues to be at the interface given their RASA values. The number of times a residue r_i appeared in the top 15 patches ranked by WPA was denoted *n*, and the number of times a residue r_i appeared in the top 15 patches ranked by JP was denoted *m*. The single patch score assigned to residue r_i was ($n + m$). Using a naïve Bayes approach, PredUs 2.0 generated a likelihood ratio (LR) from the original PredUs and a LR from the propensity patch score. The final interface score that PredUs 2.0 assigns to each residue is the product of LR_{PredUs} and LR_{patch} [3].

C. ISPRED4

ISPRED4 is one of the best performing intrinsic-based protein binding interface predictors currently available. It was developed by training an SVM model on a dataset (DBv5Sel) of 314 different monomer chains with complex structures that had been resolved by X-ray crystallography. Interface residues were defined as those that lost at least 1Å^2 of ASA (computed with the DSSP program [20]) when transitioning from a protein's unbound to complex form. In the SVM model, each of the training proteins' surface residues were represented by a 46-dimensional feature vector consisting of 10 different groups of descriptors (Table 2). ISPRED4 combined its SVM model with a Grammatical-Restrained Hidden Conditional Random Field (GRHCRF) to account for possible correlations between neighboring surface residues. For a given query protein, ISPRED4 calculated interface prediction scores by plugging the query residues' feature vectors into its trained SVM/GRHCRF model [21].

The feature vector included 34 sequence-based features that formed 5 groups of descriptors. The sequence profile descriptor represented evolutionary information for each primary sequence position of a query protein. PSI-BLAST [22] was used to search the Uniprot Reference Cluster 90 database [23] for sequences similar to the query protein sequence, and the output served as the 20-dimensional sequence profile vector. Based on the sequence profile, a conservation score descriptor was calculated using the normalized Shannon's entropy equation [24]. The interface propensity descriptor was calculated for each residue type r using the log-ratio of its interface frequency to its surface frequency. The 10 orthogonal properties introduced by Kidera *et al.* were incorporated into a group of 10 residue properties descriptors, which reflected the physico-chemical nature of each residue type *r* [25]. A multiple sequence alignment (MSA) for each query protein was generated using HHblits aligner against the UniprotKB database [26]. Based on the MSA, the PSICOV [27] and MI methods were each used to calculate two co-evolutionary scores, and each method's scores formed a group of two descriptors.

Table 2. [11] ISPRED4 Groups of Feature Descriptors

Descriptor	Program(s) Used	Number of Features
Sequence profile	PSI-BLAST	20
Conservation score	PSI-BLAST	1
Interface propensity	In-house script	1
Residue properties	In-house script	10
Mutual Information / PSICOV	HHBlits	2
Depth indexes	PSAIA	3
Protrusion indexes	PSAIA	4
Secondary structure	DSSP	3
Average B-Factor	In-house script	1
RSA difference	DSSP, SABLE	1

ISPRED4's feature vector also included 12 structure-based features that comprised 5 groups of descriptors. ISPRED4 used the PSAIA toolkit [28] to compute protrusion and depth indexes for surface residues. Protrusion indexes consisted of a group of four descriptors, and depth indexes contained a group of three descriptors. Using the DSSP program, residues were assigned to one of three secondary structure classes: helix (H, G, I), strand (E, B), or coil (T, S). A group of three descriptors was computed for each residue r_i, representing the frequency of helical, strand, and coil residues in its surface patch. An average B-factor descriptor was computed for a surface residue by averaging the B-factors for its individual atoms. The dRSA descriptor was calculated by subtracting a residue's observed RSA value from its predicted RSA value (SABLE predictor [29]).

III. Methods

A. Meta-DPI

Meta-DPI was developed on the two datasets of query proteins used in the development of DockPred. The DB dataset contained 107 protein chains from the Docking Benchmark database, and the NOX dataset contained 116 protein chains from the NOX database (see Appendix A) [13, 14]. Each residue of every query protein was assigned an interface score i of either 0 (non-interface) or 1 (at interface) based on the experimentally determined complex structures. DockPred, PredUs 2.0, and ISPRED4 were used to analyze the non-complexed query protein chains. The three methods generated prediction scores ($0 \leq p \leq 1$) for each residue of every query protein; the higher the score assigned to a residue, the more likely it was to be at the interface. Appendix B includes each method's prediction file for an example query protein.

A logistic regression model was employed to determine how to combine the prediction scores of the three methods into a metamethod prediction score. Although linear regression has been used for some previous metamethods, such as meta-PPISP [10], logistic regression was chosen for meta-DPI because the interface score (dependent variable) can only have discrete values (0 or 1). Figure 4 illustrates how a logistic regression model fits discrete categorical data better than a linear regression model. The function used in a logistic model is:

$$\boldsymbol{P}\left(i = 1 \mid x_1, x_2, \dots x_j\right) = \frac{1}{1 + e^{-(b_0 + \sum b_j x_j)}}$$

where $\boldsymbol{P}\left(i = 1 \mid x_1, x_2, \dots x_j\right)$ refers to the probability that $i = 1$ given the values of $x_1, x_2, \dots x_j$. The target dependent variable is i, and the explanatory independent variables are $x_1, x_2, \dots x_j$. Based on previously collected data for i and x_j, maximum likelihood is used to estimate the parameters b_0 and b_j (“fitting” the logistic model). The likelihood function is a function of the unknown parameters, and it represents the joint probability of observing the obtained data. The parameters are estimated by setting each parameter’s partial derivative to zero (maximization), which results in a system of equations that can be solved iteratively with a computer program [30].

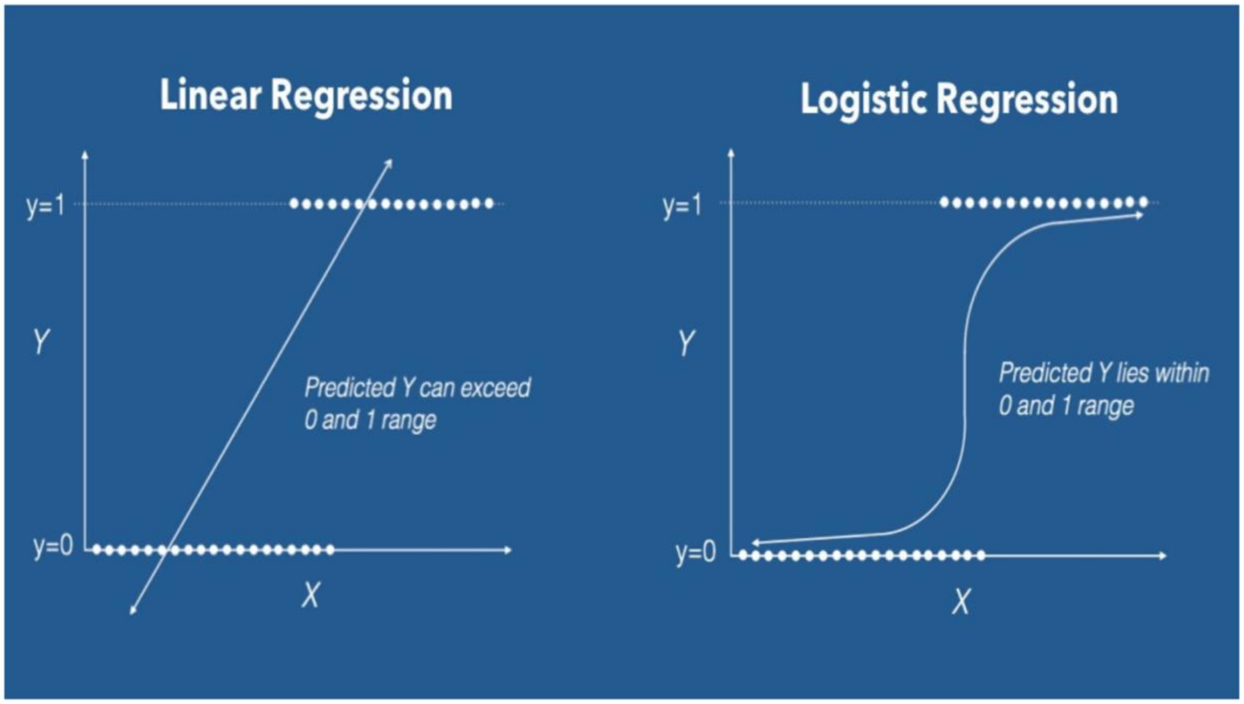

Linear Regression
y=1
Y
Predicted Y can exceed 0 and 1 range
y=0
X
Logistic Regression
y=1
Y
Predicted Y lies within 0 and 1 range
y=0
X

Fig 4. Linear Vs. Logistic Regression. The dependent variable Y only has discrete values of 0 and 1. The linear regression model does not fit the discrete data well, and it predicts Y values outside of the 0-1 range. The logistic regression model does fit the discrete data well, and it predicts Y values within the 0-1 range. (Based on [29])

Each query protein residue served as a data point on which the logistic regression model was trained; the prediction scores from the individual methods (DockPred, PredUs 2.0, & ISPRED4) were the independent variables, and the interface score was the dependent variable. The logistic regression model was trained on each set of proteins separately in order to cross-validate the results. The coefficients generated from training the model on DB proteins were used to calculate metamethod prediction scores for NOX proteins, and the coefficients generated from training on NOX proteins were used to calculate metamethod prediction scores for DB proteins. The logistic function was used to calculate the metamethod prediction scores as follows:

$$p_{meta-DPI} = \frac{1}{1 + e^{-(b_0 + b_{DockPred} p_{Dockpred} + b_{PredUs} p_{PredUs} + b_{ISPRED} p_{ISPRED})}}$$

where b_{PredUs} refers to the coefficient generated for PredUs 2.0, and p_{PredUs} refers to the prediction score assigned by PredUs 2.0 for a given residue, and the other coefficients similarly describe the other two methods.

In order to acquire a better insight into how each individual method contributed to meta-DPI, three additional metamethods were created by removing DockPred, PredUs 2.0, or ISPRED4. Meta-DI combined DockPred and ISPRED4, meta-DP combined DockPred and PredUs 2.0, and meta-PI combined PredUs 2.0 and ISPRED4. The cross-validation and calculation of metamethod prediction scores was performed in the same manner described above for meta-DPI. The python script written to execute the logistic regression analysis is included in Appendix C.

B. Evaluation of Prediction Methods

Each query protein is composed of n_+ experimentally-determined interfacial residues in the positive class and n_- experimentally-determined non-interfacial residues in the negative class.

Since protein residues belong to one of two categories, interface prediction methods are considered binary classifiers. After a classifier generates prediction scores for a query protein, the query residues are divided into m_+ predicted interfacial residues and m_- predicted non-interfacial residues. One way to split a protein's residues is to sort them according to their prediction scores (p), and the top k number of residues are assigned to the m_+ class and the remaining residues to the m_- class. Another approach is to assign residues for which $p \geq T$ to the m_+ class and residues for which $p < T$ to the m_- class (T refers to the p threshold value).

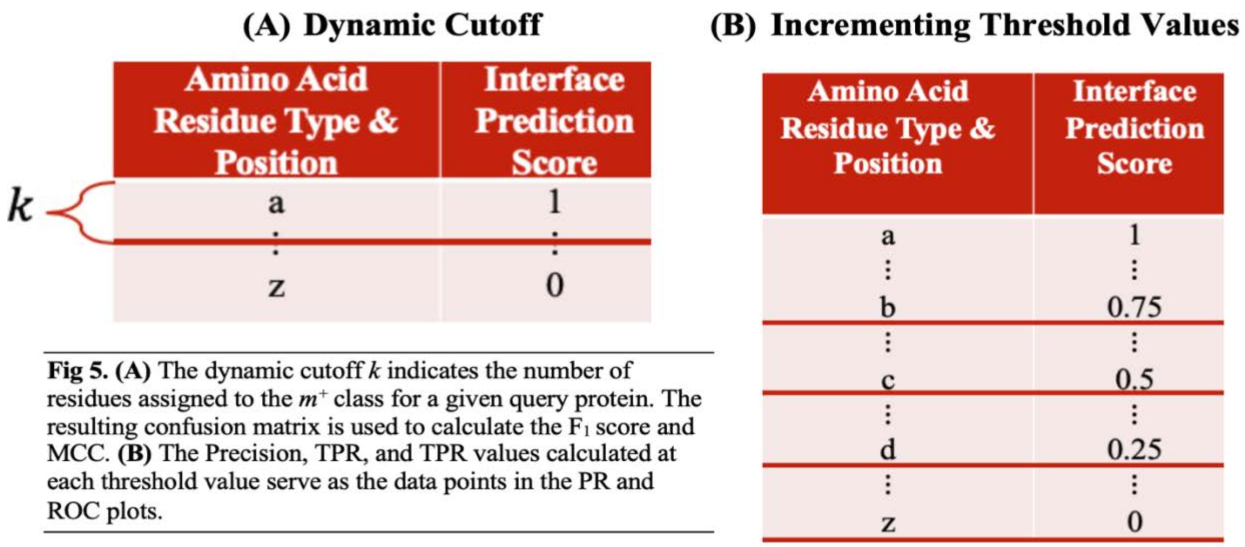
Once a classification method has been applied to the proteins in a dataset, each residue can fall into one of four categories: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). TP refers to predicted interfacial residues that were experimentally determined to be interfacial $(m^+ \cap n^+)$, while FP refers to predicted interfacial residues that were experimentally determined to be non-interfacial $(m^+ \cap n^-)$. TN refers to predicted non-interfacial residues that were experimentally determined to be non-interfacial $(m^- \cap n^-)$, while FN refers to predicted non-interfacial residues that were experimentally determined to be interfacial $(m^- \cap n^+)$. The four binary classification outcomes can be presented in a 2 x 2 confusion matrix $\boldsymbol{CM} = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$. When creating a binary classification model, the goal is to maximize the number of TP and TN, while minimizing the number of FP and FN. A perfect classification generates the confusion matrix $\boldsymbol{CM} = \begin{pmatrix} n^+ & 0 \\ 0 & n^- \end{pmatrix}$, and a perfect misclassification generates the confusion matrix $\boldsymbol{CM} = \begin{pmatrix} 0 & n^+ \\ n^- & 0 \end{pmatrix}$ [31].

A variety of classifier evaluation metrics can be calculated from confusion matrix values (Table 3). Precision $\left(\frac{TP}{TP+FP} = \frac{m^+ \cap n^+}{m^+}\right)$ refers to the fraction of a classifier's m_+ class that also belongs to the n_+ class, and recall $\left(\frac{TP}{TP+FN} = \frac{m^+ \cap n^+}{n^+}\right)$ refers to the fraction of the n_+ class included in a classifier's m_+ class. The F1 score is the harmonic mean between precision and recall, so it summarizes a classifier's performance in generating its m_+ class. The MCC is the only metric that achieves a high score only if a classifier correctly predicted the majority of the n_+ class and the majority of the n_- class. Since the F1 score is independent of the number of TN and changes when the n_+ and n_- classes are swapped, it is a less informative than the MCC metric.

Table 3. Binary Classifier Evaluation Metrics

Evaluation Metric	**Formula**
Precision	$\frac{TP}{TP+FP}$
Recall / True Positive Rate	$\frac{TP}{TP+FN}$
False Positive Rate	$\frac{FP}{TN+FP}$
F1 Score	$\frac{2 \times Precision \times Recall}{Precision + Recall}$
Matthews Correlation Coefficient	$\frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

For a given query protein, the number of residues k to assign to the m_+ class was determined using the dynamic cutoff formula [3] proposed by PredUs 2.0: $k = 6.1N^{0.3}$, where N refers to the number of query protein surface residues (Figure 5A). The values of k for the query proteins in our dataset is included in Appendix A. F1 and MCC scores were calculated for each individual prediction method and metamethod for all query proteins in the DB and NOX datasets. Global F1 score and MCC, which reflect a prediction classifier's performance on all of the proteins in a dataset as a whole, were subsequently calculated. The perl6 scripts written to ascertain the F1 score and MCC are included in Appendices D and E.


(A) Dynamic Cutoff

Amino Acid Residue Type & Position	Interface Prediction Score
a	1
⋮	⋮
z	0

(B) Incrementing Threshold Values

Amino Acid Residue Type & Position	Interface Prediction Score
a	1
⋮	⋮
b	0.75
⋮	⋮
c	0.5
⋮	⋮
d	0.25
⋮	⋮
z	0

Fig 5. (A) The dynamic cutoff *k* indicates the number of residues assigned to the m^+ class for a given query protein. The resulting confusion matrix is used to calculate the F1 score and MCC. **(B)** The Precision, TPR, and TPR values calculated at each threshold value serve as the data points in the PR and ROC plots.

The Receiver Operator Characteristic (ROC) and Precision-Recall (PR) curves give an overview of a classifier's performance over a range of thresholds. The ROC plot shows pairs of true positive rate (TPR) and false positive rate (FPR) values at all possible thresholds, and the PR plot displays pairs of precision and recall values at all possible thresholds. TPR is equivalent to recall, and FPR $\left(\frac{FP}{TN+FP} = \frac{m^+ \cap n^-}{n^-}\right)$ refers to the fraction of the n_- class included in a classifier's m_+ class. The area under the curve (AUC) is a metric that quantifies the results in ROC and PR curves. A random unskilled classifier will yield a ROC plot of $y = x$ (TPR = FPR), which has a ROC AUC of 0.5. A PR plot of $y = \frac{n^+}{(n^+ + n^-)}$ will be generated by a random unskilled classifier, which has a PR AUC of $\frac{n^+}{(n^+ + n^-)}$. Generally, a small fraction of a protein's residues appears at its interface, so $n^+$and n^- are imbalanced classes ($n^+ \ll n^-$). Figure 6 illustrates the difference between the confusion matrices of balanced and imbalanced classes. Since precision can differentiate between a classifier's performance on balanced classes versus imbalanced classes, unlike TPR and FPR, PR curves are more informative than ROC curves for a problem like interface prediction that involves imbalanced classes [32].

For each prediction method, threshold values (T) were incremented by 0.01, starting from 0 and ending at 1 (Figure 5B). Residues with prediction scores $p \geq T$ were assigned to the m_+ class, and confusion matrices were generated accordingly at each T value. TPR, FPR, precision, and recall were calculated at every T value. The TPR and FPR at a single T value served as a data point on the ROC plot, and the precision and recall at a single T value served as a data point on the PR plot. Using this approach, ROC and PR curves were generated to display each prediction method's performance on the DB and NOX datasets. The AUC for the ROC and PR plots were approximated using the trapezoidal rule. Appendix F contains the perl6 scripts written to increment T values and calculate the evaluation metrics for the ROC and PR curves.

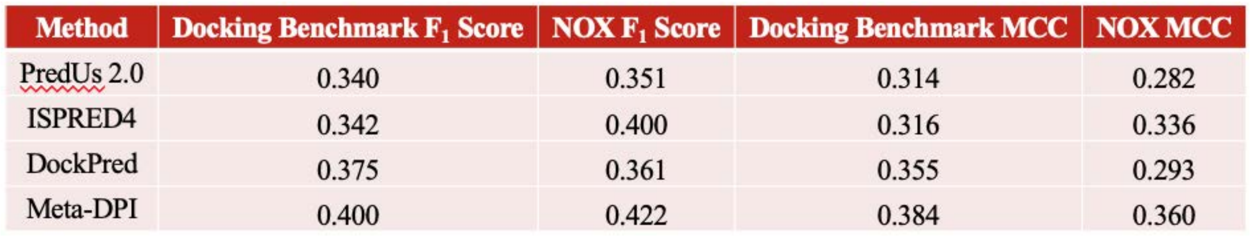
$$\boldsymbol{CM} = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix} \qquad \boldsymbol{CM}\ (\boldsymbol{Balanced}) = \begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix} \qquad \boldsymbol{CM}\ (\boldsymbol{Imbalanced}) = \begin{pmatrix} 3 & 2 \\ 6 & 9 \end{pmatrix}$$

Fig 6. Both confusion matrices have 20 total elements. The balanced CM has 10 elements in the n^+class and 10 elements in the n^- class. The imbalanced CM has 5 elements in the n^+class and 15 elements in the n^- class. (Based on [31])

IV. Results & Discussion

A. Single Threshold Evaluation Metrics: F_1 Score & MCC

Table 4. F_1 Score & MCC of Individual Classifiers and meta-DPI for DB and NOX Databases



Method	Docking Benchmark F_1 Score	NOX F_1 Score	Docking Benchmark MCC	NOX MCC
PredUs 2.0	0.340	0.351	0.314	0.282
ISPRED4	0.342	0.400	0.316	0.336
DockPred	0.375	0.361	0.355	0.293
Meta-DPI	0.400	0.422	0.384	0.360

In developing meta-DPI, I aimed to create a protein interface predictor that performed better than the currently available methods. A logistic regression model was used to train meta-DPI on the interface prediction values of DockPred, ISPRED4, and PredUs 2.0 for proteins in the DB and NOX datasets separately. A cross-validation experiment was performed by using the logistic regression results of each training set to generate meta-DPI prediction values for proteins in the other set. In order to compare the performance of meta-DPI relative to its constituent methods, single-threshold evaluation metrics were calculated for the proteins in each dataset as a whole.

In Table 4, the Docking Benchmark F_1 score and MCC reflect the performance of each classification method on the proteins in the DB dataset as a whole, while the NOX F_1 score and MCC reflect each classification method's performance on the proteins in the NOX dataset as a whole. The meta-DPI prediction values for DB query protein residues, which were calculated using the coefficients obtained from the logistic regression performed on the NOX dataset, were used to calculate the DB F_1 score and MCC for meta-DPI. The NOX F_1 score and MCC for meta-DPI were calculated in the opposite manner. As shown in Table 4, meta-DPI outperformed each of its constituent methods according to both evaluation metrics, regardless of the dataset on which it was trained.

For most query proteins, the number of dynamic cutoff residues was slightly higher than the number of experimental annotated residues (see Appendix A). For the NOX query protein 2PCB.A, which had 25 more cutoff residues than annotated residues (31 and 6, respectively), each classifier had a lower precision than its average precision and a higher recall than its average recall. On the other hand, for the NOX query protein 1EFV.A, which had 28 fewer cutoff residues than annotated residues (31 and 59, respectively), each classifier had a higher precision than its average precision and a lower recall than its average recall. Thus, the number of cutoff residues relative to annotated residues directly influences the two components of the F_1 score, precision and recall.

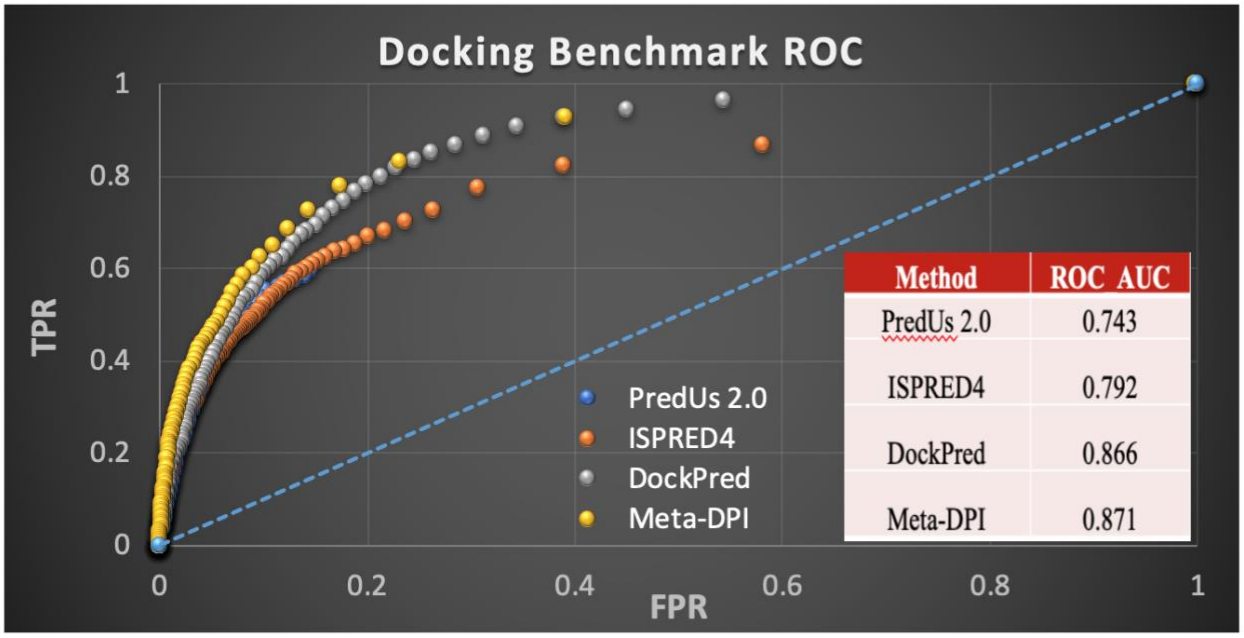
When comparing each classifier's average MCC with their MCC for the aforementioned NOX query proteins, no consistent pattern of outliers emerges. This is because the MCC depends on all four categories of the confusion matrix equally. Since MCC is not influenced by the difference between the number of cutoff and annotated residues, it is a more informative metric than F_1 score, especially for query proteins with an unknown complex structure and unknown number of annotated residues.

Table 5. F_1 Score & MCC of All Metamethods for DB and NOX Databases

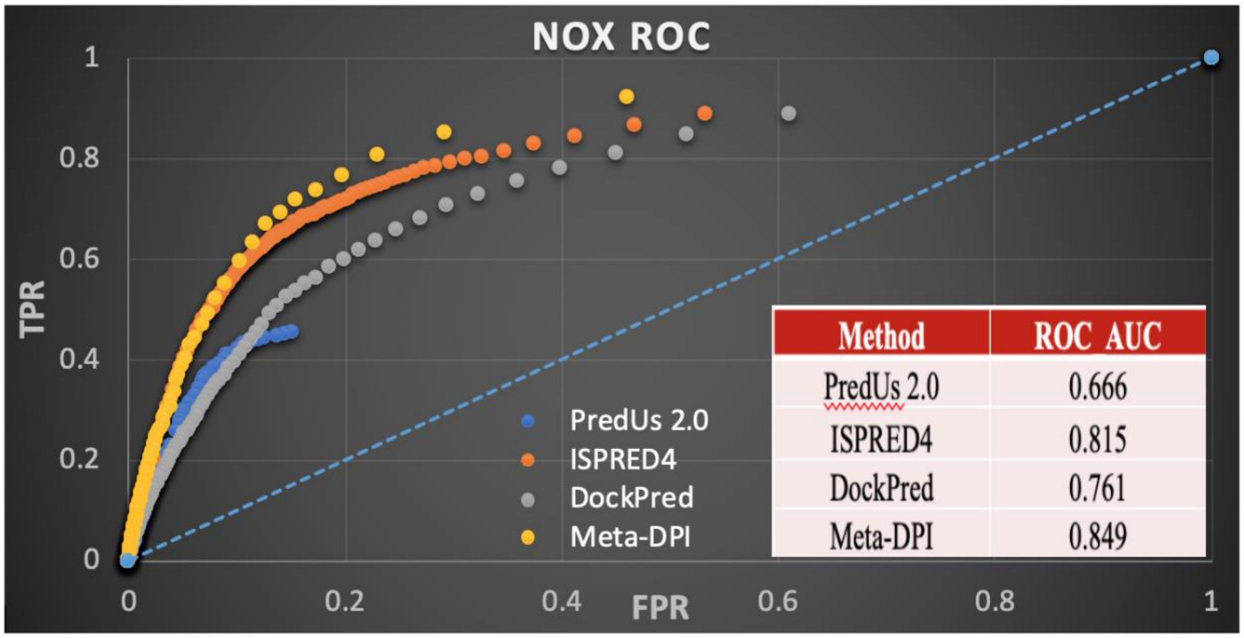
Method	Docking Benchmark F_1 Score	NOX F_1 Score	Docking Benchmark MCC	NOX MCC
Meta-DI	0.391	0.418	0.373	0.356
Meta-DP	0.398	0.378	0.382	0.311
Meta-PI	0.367	0.418	0.346	0.356
Meta-DPI	0.400	0.422	0.384	0.360

Metamethods that combined two of the three interface predictors were created to gain insight into each individual classifier's role in meta-DPI. In Table 5, the performance of meta-DPI is compared to the performance of the metamethods that combined two of the three classifiers. Meta-DPI performed better than the other metamethods according to both F_1 score and MCC, regardless of the dataset on which the metamethods were trained. Thus, combining all three methods into a metamethod yielded a classifier superior to the ones obtained when only combining two of the methods. By comparing the results in Tables 4 and 5, it can be seen that the performance of the individual classifiers in a metamethod correlates with the metamethod's performance. The DB F_1 score of DockPred (0.375) was 10% higher than the scores of ISPRED4 (0.342) and PredUs 2.0 (0.340). Therefore, the DB F_1 score of meta-PI (0.367), which combined the lower scoring ISPRED4 and PredUs 2.0, was significantly lower than the scores of the metamethods that included DockPred, meta-DP (0.398) and meta-DI (0.391).

B. Threshold-free Evaluation Metrics: ROC & PR Curves



Method	ROC AUC
PredUs 2.0	0.743
ISPRED4	0.792
DockPred	0.866
Meta-DPI	0.871



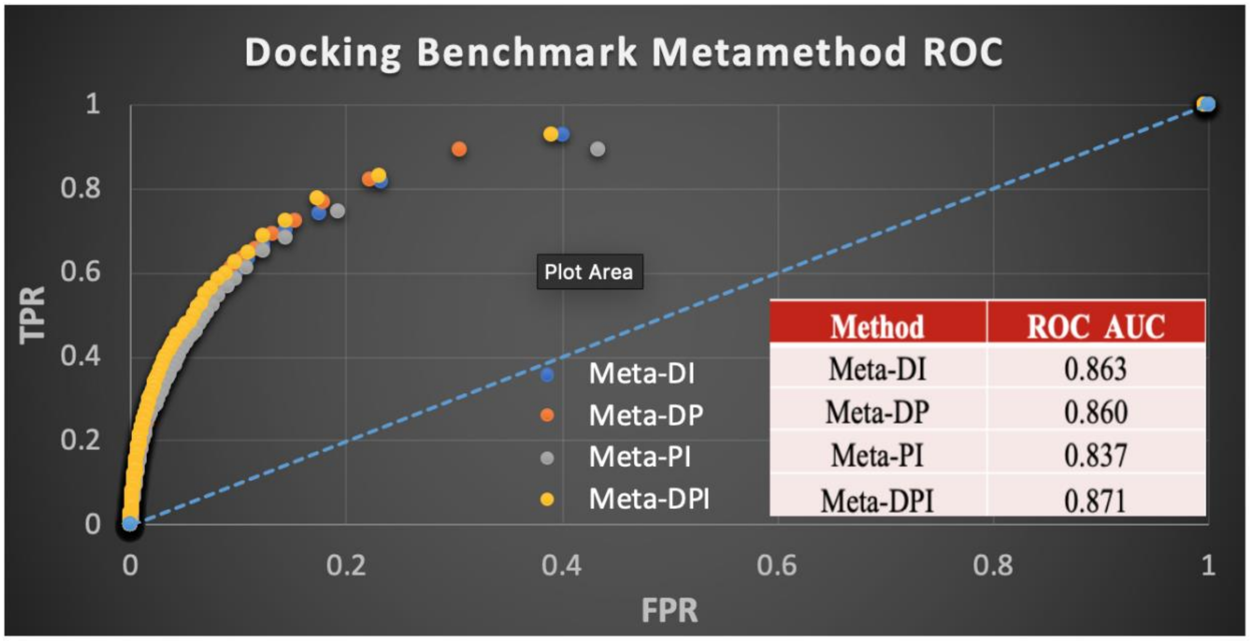
Method	ROC AUC
PredUs 2.0	0.666
ISPRED4	0.815
DockPred	0.761
Meta-DPI	0.849

Fig 7. ROC Graphs for DB & NOX Datasets. Each classifier's AUC value is included in the enclosed table. The dashed line represents the ROC plot of a random unskilled classifier (TPR = FPR), which has a ROC AUC of 0.5.

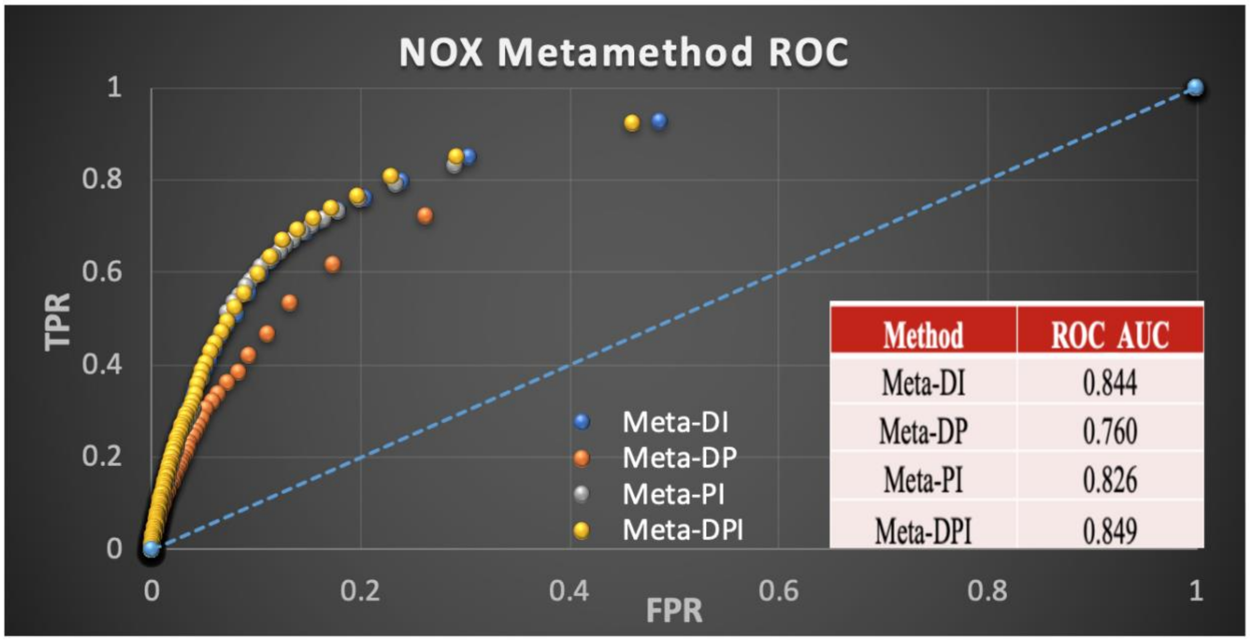
For both the DB and NOX datasets, the ROC AUC of meta-DPI was higher than the ROC AUC of the individual classifiers (Figure 7). While the F_1 score and MCC demonstrated meta-DPI's enhanced ability to classify query protein residues above the dynamic cutoff, the ROC plots confirmed meta-DPI's superior classification of all query protein residues.

The average performance of PredUs 2.0 relative to the second-best performing method was 9.34% lower in terms of ROC AUC, which was significantly larger than its relative poorer performance in terms of F_1 score (1.68%) and MCC (2.19%). This was due to the fact that PredUs 2.0 assigned a prediction score of 0 to a significantly higher percentage of residues than the other methods did, so it had an FPR of 1 at $T = 0.00$ and an FPR of less than 0.2 at $T = 0.01$ (no data points with an FPR between 0.2 and 1). The lack of differentiation between the lowest scoring residues diminished PredUs 2.0's ROC AUC, but it did not affect its performance for single-threshold evaluation metrics. This principle can be illustrated by examining DockPred's ROC plot for the DB dataset. If the 10 data points DockPred had with an FPR between 0.2 and 1 $(0.01 \leq T \leq .10)$ were eliminated by changing the residues with $0.01 \leq p \leq .10$ to $p = 0$, its ROC AUC would drop from 0.866 to 0.821. Thus, ROC AUC is a metric that evaluates classifiers in a somewhat biased manner because it undervalues classifiers that group the lowest scoring residues together.

For both the DB and NOX datasets, the ROC AUC of meta-DPI was higher than the ROC AUC of the metamethods that only combined two classifiers (Figure 8). By comparing the results in Figures 7 and 8, it can be seen that the threshold-free performance of the individual classifiers in a metamethod correlates with the metamethod's performance. ISPRED4's ROC AUC (0.815) for the NOX database was significantly higher than the ROC AUC of DockPred (0.761) and PredUs 2.0 (0.666). Consequently, the ROC AUC of meta-DP (0.760), which combined the poorer performing DockPred and PredUs 2.0, was lower than the ROC AUC of the metamethods that included ISPRED4, meta-DI (0.826) and meta-PI (0.844).

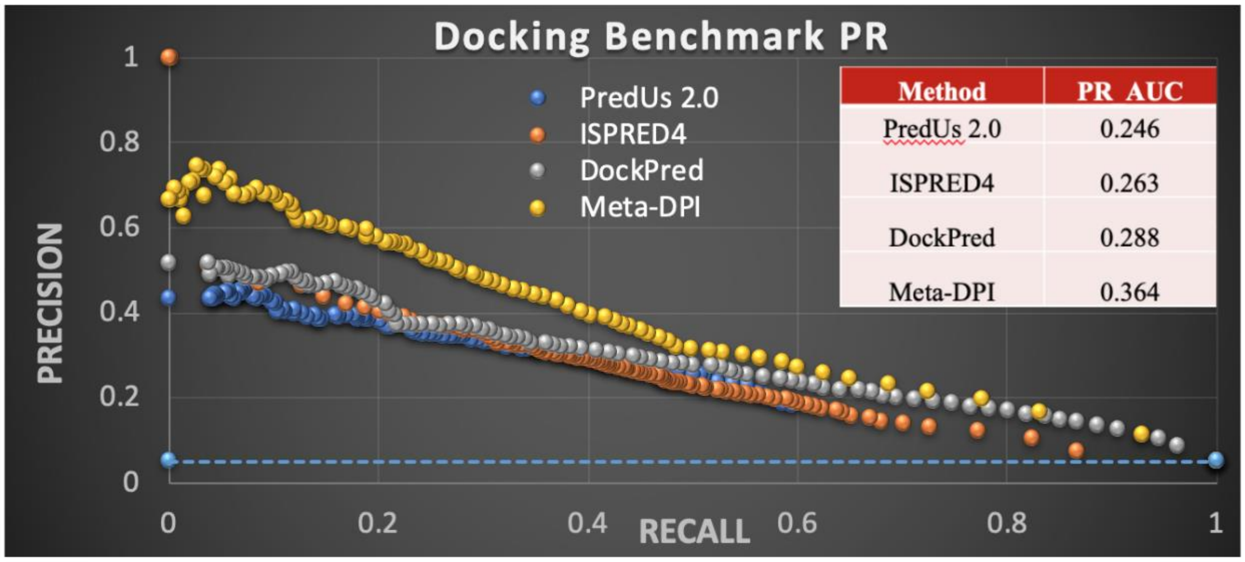


Method	ROC AUC
Meta-DI	0.863
Meta-DP	0.860
Meta-PI	0.837
Meta-DPI	0.871

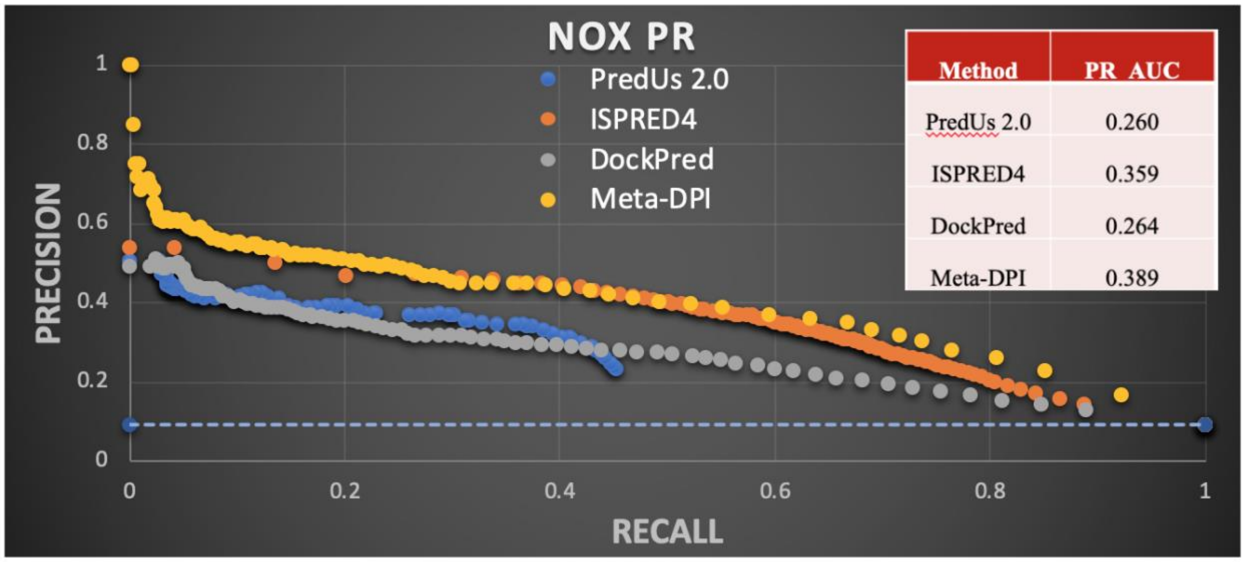


Method	ROC AUC
Meta-DI	0.844
Meta-DP	0.760
Meta-PI	0.826
Meta-DPI	0.849

Fig 8. ROC Graphs for Metamethods for DB & NOX Datasets. Each metamethod's AUC value is included in the enclosed table. The dashed line represents the ROC plot of a random unskilled classifier (TPR = FPR), which has a ROC AUC of 0.5.



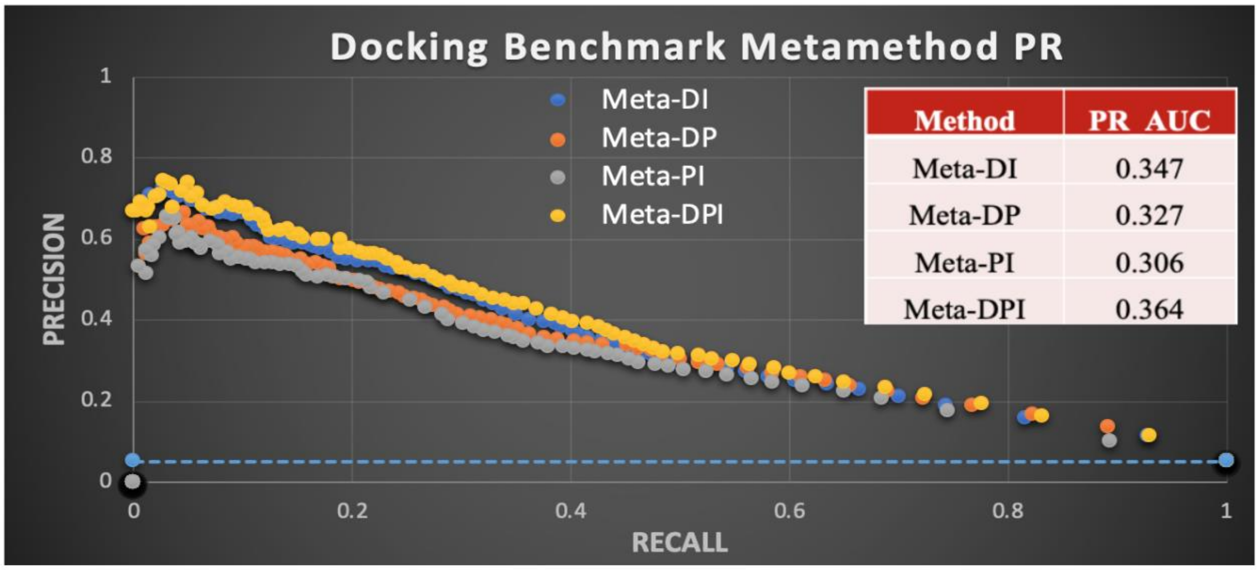
Method	PR AUC
PredUs 2.0	0.246
ISPRED4	0.263
DockPred	0.288
Meta-DPI	0.364



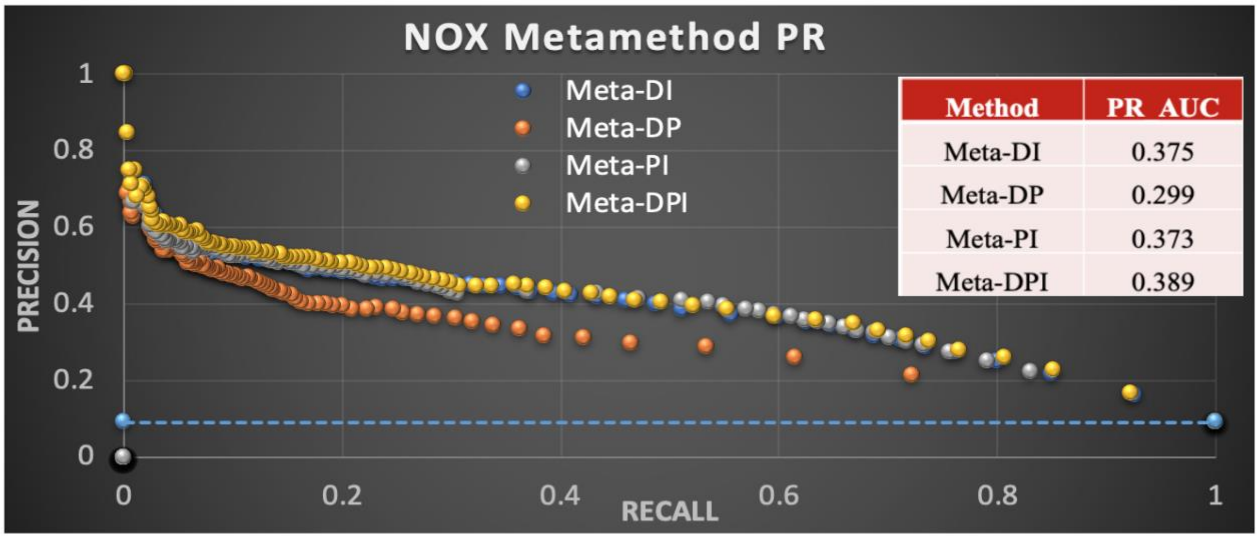
Method	PR AUC
PredUs 2.0	0.260
ISPRED4	0.359
DockPred	0.264
Meta-DPI	0.389

Fig 9. Precision-Recall Graphs for DB & NOX Datasets. Each classifier's AUC value is included in the enclosed table. The dashed line represents the PR plot of a random unskilled classifier ($y = \frac{n^+}{(n^+ + n^-)}$), which is $y = 0.052$ for the DB dataset and $y = 0.091$ for the NOX dataset.

For the DB dataset, the Precision-Recall (PR) AUC of meta-DPI (0.364) was 26% greater than the PR AUC of DockPred (0.288), which was the highest among individual classifiers. As can be seen in the top graph in Figure 9, meta-DPI could correctly classify 0-40% of the n^+ residues in the DB dataset, while maintaining a significantly higher degree of precision than the individual classifiers. The results displayed in Figures 9 and 10 further validate meta-DPI's superior classification ability even by a metric like precision, which is sensitive to class imbalance.



Method	PR AUC
Meta-DI	0.347
Meta-DP	0.327
Meta-PI	0.306
Meta-DPI	0.364



Method	PR AUC
Meta-DI	0.375
Meta-DP	0.299
Meta-PI	0.373
Meta-DPI	0.389

Fig 10. Precision-Recall Graphs for Metamethods for DB & NOX Datasets. Each metamethod's AUC value is included in the enclosed table. The dashed line represents the PR plot of a random unskilled classifier ($y = \frac{n^+}{(n^+ + n^-)}$), which is $y = 0.052$ for the DB dataset and $y = 0.091$ for the NOX dataset.

Among the individual classifiers, PredUs 2.0 consistently performed the worst out of the individual classifiers in terms of global evaluation metrics. For query proteins that had a close structural neighbor with a known complex structure, PredUs 2.0 performed well. However, for query proteins that did not have a close structural neighbor with a known complex structure, PredUs 2.0 performed poorly. The DB and NOX datasets both contain some query proteins with close structural neighbors with known complex structures, as well as some query proteins without close structural neighbors with known complex structures. This results in PredUs 2.0's classification performance on each data set as a whole being diminished by its inferior classification of the query proteins without close structural neighbors with known complex structures.

V. Conclusion

Through the development of the novel metamethod meta-DPI, I have shown that orthologous protein interface prediction methods can be combined to create a metamethod that is a superior interface classifier relative to its constituent methods. In contrast to previous metamethods that used linear regression, meta-DPI employed logistic regression as the model by which to combine prediction methods. Meta-DPI's enhanced classification performance was verified by single-threshold metrics, F_1 score and MCC, and by threshold-free metrics, ROC AUC and PR AUC. The creation of increasingly robust metamethods is a viable approach for continuing to enhance the performance of interface predictors, despite the limitations of intrinsic-based and template-based methods.

VI. Future Work

The next step in the development of meta-DPI will be to quantify the relationship between the classification performance of PredUs 2.0 and how similar the structural neighbors are to a given query protein. Once this relationship is ascertained, it can be incorporated into meta-DPI's classification model by making structural neighbor similarity one of the features analyzed during the training process. Another issue to investigate is how the superfold family of query proteins correlates with each classifiers' performance, which will allow superfold family to be an additional feature integrated into meta-DPI's classification model. The final step in perfecting the metamethod will be to determine the optimal machine learning algorithm for integrating the array of features in the classification model. Random forest and decision tree algorithms can be trained and tested in the same manner as the logistic regression model in order to determine which approach yields the best classification performance.

After meta-DPI has been optimized, I will investigate how it can be used to predict the antigen epitopes of antigen-antibody binding interactions specifically. Epitopes can be difficult to identify because they appear to be barely distinguishable from the rest of the protein surface [2]. In order to prime meta-DPI for epitope prediction, the training set will be adjusted to only include antibody-antigen interactions. Interface predictors designed specifically for epitope prediction, like EpiPred [33], will be incorporated as an input feature in meta-DPI's classification model. Identifying an antigen's epitope is a key step in the process of the development of a vaccine for a disease associated with the antigen. Additionally, understanding a monoclonal antibody's binding mechanism is critical when studying its therapeutic potential.

VII. References

1. Vagenende, V., et al., *Quantifying the molecular origins of opposite solvent effects on protein-protein interactions.* PLoS Comput Biol, 2013. **9**(5): p. e1003072.
2. Esmaielbeiki, R., et al., *Progress and challenges in predicting protein interfaces.* Brief Bioinform, 2016. **17**(1): p. 117-31.
3. Hwang, H., D. Petrey, and B. Honig, *A hybrid method for protein-protein interface prediction.* Protein Sci, 2016. **25**(1): p. 159-65.
4. Diaz-Eufracio, B.I., J.J. Naveja, and J.L. Medina-Franco, *Protein-Protein Interaction Modulators for Epigenetic Therapies.* Adv Protein Chem Struct Biol, 2018. **110**: p. 65-84.
5. Ramagopal, U.A., et al., *Structural basis for cancer immunotherapy by the first-in-class checkpoint inhibitor ipilimumab.* Proc Natl Acad Sci U S A, 2017. **114**(21): p. E4223-E4232.
6. Cheng, Y., *Single-particle cryo-EM-How did it get here and where will it go.* Science, 2018. **361**(6405): p. 876-880.
7. Chen, H. and H.X. Zhou, *Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against NMR data.* Proteins, 2005. **61**(1): p. 21-35.
8. Neuvirth, H., R. Raz, and G. Schreiber, *ProMate: a structure based prediction program to identify the location of protein-protein binding sites.* J Mol Biol, 2004. **338**(1): p. 181-99.
9. Liang, S., et al., *Protein binding site prediction using an empirical scoring function.* Nucleic Acids Res, 2006. **34**(13): p. 3698-707.
10. Qin, S. and H.X. Zhou, *meta-PPISP: a meta web server for protein-protein interaction site prediction.* Bioinformatics, 2007. **23**(24): p. 3386-7.
11. Hajduk, P.J., J.R. Huth, and S.W. Fesik, *Druggability indices for protein targets derived from NMR-based screening data.* J Med Chem, 2005. **48**(7): p. 2518-25.
12. Viswanathan, R., et al., *Protein-protein binding supersites.* PLoS Comput Biol, 2019. **15**(1): p. e1006704.
13. Zhu, H., et al., *NOXclass: prediction of protein-protein interaction types.* BMC Bioinformatics, 2006. **7**: p. 27.
14. Vreven, T., et al., *Updates to the Integrated Protein-Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2.* J Mol Biol, 2015. **427**(19): p. 3031-41.
15. Sobolev, V., et al., *Automated analysis of interatomic contacts in proteins.* Bioinformatics, 1999. **15**(4): p. 327-32.
16. Petrey, D. and B. Honig, *GRASP2: visualization, surface properties, and electrostatics of macromolecular structures and sequences.* Methods Enzymol, 2003. **374**: p. 492-509.
17. Yang, A.S. and B. Honig, *An integrated approach to the analysis and modeling of protein sequences and structures. I. Protein structural alignment and a quantitative measure for protein structural distance.* J Mol Biol, 2000. **301**(3): p. 665-78.
18. Li, W. and A. Godzik, *Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences.* Bioinformatics, 2006. **22**(13): p. 1658-9.
19. Zhang, Q.C., et al., *PredUs: a web server for predicting protein interfaces using structural neighbors.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W283-7.

20. Kabsch, W. and C. Sander, *Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features*. Biopolymers, 1983. **22**(12): p. 2577-637.
21. Savojardo, C., et al., *ISPRED4: interaction sites PREDiction in protein structures with a refining grammar model.* Bioinformatics, 2017. **33**(11): p. 1656-1663.
22. Altschul, S.F., et al., *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*. Nucleic Acids Res, 1997. **25**(17): p. 3389-402.
23. Suzek, B.E., et al., *UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches.* Bioinformatics, 2015. **31**(6): p. 926-32.
24. Sander, C. and R. Schneider, *Database of homology-derived protein structures and the structural meaning of sequence alignment.* Proteins, 1991. **9**(1): p. 56-68.
25. Kidera, A., *Statistical analysis of the physical properties of the 20 naturally occurring amino acids* J Protein Chem. , 1985. **4**: p. 23-55.
26. Remmert, M., et al., *HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment.* Nat Methods, 2011. **9**(2): p. 173-5.
27. Jones, D.T., et al., *PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments.* Bioinformatics, 2012. **28**(2): p. 184-90.
28. Mihel, J., et al., *PSAIA - protein structure and interaction analyzer.* BMC Struct Biol, 2008. **8**: p. 21.
29. Adamczak, R., A. Porollo, and J. Meller, *Accurate prediction of solvent accessibility using neural networks-based regression.* Proteins, 2004. **56**(4): p. 753-67.
30. Klein, D.G.K.M., *Logistic Regression: A Self-Learning Text*. 2nd ed. Statistics for Biology and Health, ed. K. Dietz. 2002, New York: Springer-Verlag.
31. Chicco, D. and G. Jurman, *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC Genomics, 2020. **21**(1): p. 6.
32. Saito, T. and M. Rehmsmeier, *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets.* PLoS One, 2015. **10**(3): p. e0118432.
33. Krawczyk, K., et al., *Improving B-cell epitope prediction and its application to global antibody-antigen docking.* Bioinformatics, 2014. **30**(16): p. 2288-94.

Appendix A: List of Proteins in Docking Benchmark and NOX Datasets

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
2SIC.E	Docking Benchmark	173	29	17
2A5T.A	Docking Benchmark	192	30	10
1DFJ.I	Docking Benchmark	325	35	18
2BTF.A	Docking Benchmark	267	33	18
1KLU.D	Docking Benchmark	165	28	11
4JCV.E	Docking Benchmark	177	29	11
4G6M.H	Docking Benchmark	320	34	14
3R9A.B	Docking Benchmark	217	31	17
2B42.B	Docking Benchmark	141	27	21
1KAC.A	Docking Benchmark	143	27	9
1GXD.C	Docking Benchmark	167	28	18
2FD6.H	Docking Benchmark	327	35	14
1DE4.E	Docking Benchmark	87	23	21
3G6D.L	Docking Benchmark	329	35	15
3SZK.F	Docking Benchmark	116	25	11
1JIW.I	Docking Benchmark	84	23	15
1JIW.P	Docking Benchmark	332	35	20
1ACB.I	Docking Benchmark	55	20	11
1GHQ.A	Docking Benchmark	197	30	5
1VFB.A	Docking Benchmark	166	28	15
3SZK.E	Docking Benchmark	107	25	12
2B42.A	Docking Benchmark	269	33	19
1KAC.B	Docking Benchmark	101	24	12
1ZHH.A	Docking Benchmark	240	32	13
3EO1.A	Docking Benchmark	330	35	12
2W9E.H	Docking Benchmark	328	35	17
1DQJ.A	Docking Benchmark	327	35	18
3EOA.L	Docking Benchmark	322	34	12
1FC2.D	Docking Benchmark	174	29	10
2A5T.B	Docking Benchmark	214	31	14
2C0L.A	Docking Benchmark	214	31	19
1TMQ.A	Docking Benchmark	306	34	18
1US7.A	Docking Benchmark	151	27	7
4M76.B	Docking Benchmark	130	26	12

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
2HLE.B	Docking Benchmark	107	25	13
2B4J.C	Docking Benchmark	69	22	8
1AVX.A	Docking Benchmark	158	28	19
2CFH.C	Docking Benchmark	118	26	10
1XQS.A	Docking Benchmark	186	29	23
3L5W.L	Docking Benchmark	326	35	10
1T6B.Y	Docking Benchmark	122	26	14
1ML0.A	Docking Benchmark	278	33	9
1OFU.A	Docking Benchmark	214	31	12
1ZHI.B	Docking Benchmark	108	25	8
1F34.A	Docking Benchmark	235	31	22
2AJF.E	Docking Benchmark	147	27	11
1ZHI.A	Docking Benchmark	162	28	9
3S9D.B	Docking Benchmark	152	28	12
1US7.B	Docking Benchmark	156	28	8
1E6J.H	Docking Benchmark	330	35	7
1TMQ.B	Docking Benchmark	93	24	15
2BTF.P	Docking Benchmark	106	25	18
3BX7.C	Docking Benchmark	101	24	16
1E4K.C	Docking Benchmark	141	27	12
2VDB.A	Docking Benchmark	499	39	11
1KXP.A	Docking Benchmark	250	32	20
1FLE.E	Docking Benchmark	167	28	11
1AK4.A	Docking Benchmark	119	26	9
3MXW.L	Docking Benchmark	327	35	14
2VXT.H	Docking Benchmark	324	35	13
4DN4.L	Docking Benchmark	310	34	14
2VIS.A	Docking Benchmark	332	35	12
1QA9.A	Docking Benchmark	83	23	13
1MQ8.B	Docking Benchmark	129	26	12
3DAW.B	Docking Benchmark	108	25	16
3HMX.L	Docking Benchmark	329	35	16
3F1P.B	Docking Benchmark	90	24	18
3AAA.C	Docking Benchmark	93	24	11
3F1P.A	Docking Benchmark	94	24	13

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
3BX7.A	Docking Benchmark	138	27	16
3DAW.A	Docking Benchmark	278	33	20
1QA9.B	Docking Benchmark	82	23	15
1MLC.A	Docking Benchmark	327	35	13
4G6J.H	Docking Benchmark	320	34	16
1T6B.X	Docking Benchmark	491	39	12
1JPS.H	Docking Benchmark	324	35	16
2J0T.A	Docking Benchmark	128	26	18
4FQI.H	Docking Benchmark	332	35	11
1AHW.A	Docking Benchmark	325	35	20
2HQS.A	Docking Benchmark	308	34	17
1FFW.A	Docking Benchmark	98	24	8
3HI6.X	Docking Benchmark	324	35	15
7CEI.B	Docking Benchmark	107	25	10
1GXD.A	Docking Benchmark	504	39	21
3V6Z.A	Docking Benchmark	329	35	21
1ACB.E	Docking Benchmark	172	29	13
1AY7.A	Docking Benchmark	85	23	9
1BGX.H	Docking Benchmark	332	35	31
1KXP.D	Docking Benchmark	361	36	26
3VLB.A	Docking Benchmark	286	33	15
1KTZ.A	Docking Benchmark	80	23	7
3BIW.A	Docking Benchmark	352	35	9
1EXB.E	Docking Benchmark	76	22	3
2O3B.A	Docking Benchmark	170	28	15
2SIC.I	Docking Benchmark	94	24	11
3VLB.B	Docking Benchmark	157	28	17
1DFJ.E	Docking Benchmark	103	25	18
1IRA.X	Docking Benchmark	113	25	23
1AY7.B	Docking Benchmark	67	22	9
2HRK.B	Docking Benchmark	102	24	8
2J0T.D	Docking Benchmark	105	25	11
7CEI.A	Docking Benchmark	73	22	13
1FFW.B	Docking Benchmark	59	21	10
1PVH.B	Docking Benchmark	139	27	8

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
3RVW.C	Docking Benchmark	321	34	13
1WEJ.H	Docking Benchmark	327	35	11
1S1Q.A	Docking Benchmark	115	25	11

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
1DOR.A	NOX	213	30	25
1YVE.I	NOX	331	35	25
1DOW.A	NOX	179	29	14
1NSE.A	NOX	313	34	39
1EMV.A	NOX	73	22	13
1CP2.A	NOX	179	29	13
1RRP.A	NOX	162	28	32
1XIK.A	NOX	239	32	37
1BKD.R	NOX	127	26	25
1TCO.A	NOX	232	31	21
1JKM.A	NOX	244	32	16
1I8L.A	NOX	164	28	8
1QBI.A	NOX	298	34	17
1QFH.A	NOX	180	29	37
1D09.A	NOX	206	30	15
1B6C.A	NOX	88	23	15
1CMX.A	NOX	164	28	22
1ETH.A	NOX	302	34	14
3C98.A	NOX	400	37	27
1BVN.T	NOX	61	21	15
1REQ.A	NOX	495	39	64
1STF.E	NOX	148	27	12
4SGB.I	NOX	48	19	6
1ONE.A	NOX	266	33	34
2NAC.A	NOX	275	33	56
1BJN.A	NOX	253	32	32
1CNZ.A	NOX	253	32	43
1SMP.I	NOX	81	23	12

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
1B3A.A	NOX	58	21	12
1B34.A	NOX	70	22	17
1AT3.A	NOX	159	28	15
1ISA.A	NOX	144	27	9
1MSP.A	NOX	107	25	12
1CMB.A	NOX	94	24	15
1I2M.A	NOX	126	26	24
1TRK.A	NOX	429	38	67
3HHR.A	NOX	139	27	21
1SPU.A	NOX	514	40	112
1LFD.A	NOX	74	22	11
1VOK.A	NOX	149	27	19
1AVW.A	NOX	159	28	19
1BRM.A	NOX	261	32	54
1UEA.A	NOX	133	26	22
1CSE.I	NOX	55	20	10
1ITB.A	NOX	119	26	25
1SMT.A	NOX	90	24	22
1EG9.A	NOX	293	34	31
1CVS.A	NOX	100	24	17
1FRV.A	NOX	198	30	59
1BI7.A	NOX	202	30	25
1TX4.A	NOX	145	27	17
1B5E.A	NOX	187	29	36
1F6Y.A	NOX	183	29	19
4MDH.A	NOX	239	32	26
1HGX.A	NOX	125	26	18
2AE2.A	NOX	196	30	18
1C0F.S	NOX	101	24	22
1HLU.A	NOX	277	33	11
1DHK.A	NOX	321	34	26
1WGJ.A	NOX	207	30	11
4XXH.A	NOX	187	29	19
2AAI.A	NOX	199	30	25
2PFL.A	NOX	444	38	23

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
1QAE.A	NOX	164	28	13
1WQ1.R	NOX	120	26	14
1BML.A	NOX	171	29	26
1GUX.A	NOX	137	27	13
1EUV.A	NOX	158	28	27
1GPE.A	NOX	351	35	28
1CC0.A	NOX	130	26	9
1PDK.A	NOX	170	28	28
1DCE.A	NOX	433	38	39
2PTC.I	NOX	53	20	8
1TGS.I	NOX	49	20	13
1QFE.A	NOX	170	28	7
1JTD.A	NOX	177	29	12
1B9M.A	NOX	217	31	41
2HHM.A	NOX	193	30	21
1VLT.A	NOX	120	26	14
1CLI.A	NOX	246	32	42
1YCS.A	NOX	153	28	11
1SOX.A	NOX	333	35	24
1GLA.F	NOX	121	26	7
1EFV.A	NOX	228	31	59
1FIN.A	NOX	223	31	29
1PP2.L	NOX	105	25	17
1BYF.A	NOX	94	24	16
1B8J.A	NOX	289	33	60
1BUH.A	NOX	221	31	12
1COZ.A	NOX	102	24	10
1HJR.A	NOX	126	26	15
1PNK.A	NOX	172	29	80
1QAX.A	NOX	327	35	78
1QOR.A	NOX	235	31	21
1FSS.A	NOX	341	35	15
1AVZ.B	NOX	89	23	11
1LUC.A	NOX	239	32	30
1AVA.A	NOX	259	32	17

PDB ID	Dataset	Surface Residues	Dynamic Cutoff Residues	Experimental Annotated Residues
1H2A.L	NOX	349	35	55
1CQI.A	NOX	201	30	18
2HDH.A	NOX	226	31	16
1YPI.A	NOX	179	29	23
2UTG.A	NOX	66	21	14
3TMK.A	NOX	167	28	11
1HSS.A	NOX	92	24	15
1ZBD.A	NOX	131	26	19
1BO1.A	NOX	252	32	18
1XSO.A	NOX	107	25	7
1B8A.A	NOX	333	35	56
1B7B.A	NOX	230	31	28
1KPE.A	NOX	93	24	33
1QAV.A	NOX	76	22	13
1EAI.C	NOX	60	21	10
1F60.A	NOX	319	34	28
2PCB.A	NOX	224	31	6
1ATN.A	NOX	270	33	16

Appendix B: Prediction Files for 1ACB.I

ISPRED4

Residue Number	Residue Type	Prediction Score
8	LYS	0.31
9	SER	0.06
10	PHE	0.00
11	PRO	0.02
12	GLU	0.02
13	VAL	0.00
14	VAL	0.06
15	GLY	0.01
16	LYS	0.01
17	THR	0.01
18	VAL	0.00
19	ASP	0.02
20	GLN	0.01
21	ALA	0.00
22	ARG	0.01
23	GLU	0.01
24	TYR	0.01
25	PHE	0.00
26	THR	0.02
27	LEU	0.31
28	HIS	0.17
29	TYR	0.03
30	PRO	0.02
31	GLN	0.02
32	TYR	0.00
33	ASP	0.05
34	VAL	0.00
35	TYR	0.02
36	PHE	0.00
37	LEU	0.04
38	PRO	0.03
39	GLU	0.02

DockPred

Residue Number	Residue Type	Prediction Score
8	LYS	0.89
9	SER	0.49
10	PHE	0.22
11	PRO	0.24
12	GLU	0.27
13	VAL	0.02
14	VAL	0.11
15	GLY	0.07
16	LYS	0.09
17	THR	0.06
18	VAL	0.33
19	ASP	0.40
20	GLN	0.07
21	ALA	0.00
22	ARG	0.80
23	GLU	0.14
24	TYR	0.20
25	PHE	0.09
26	THR	0.27
27	LEU	0.21
28	HIS	0.42
29	TYR	0.46
30	PRO	0.40
31	GLN	0.61
32	TYR	0.70
33	ASP	0.65
34	VAL	0.48
35	TYR	0.90
36	PHE	0.75
37	LEU	0.65
38	PRO	0.61
39	GLU	0.34

PredUs 2.0

Residue Number	Residue Type	Prediction Score
8	LYS	0.03
9	SER	0.05
10	PHE	0.00
11	PRO	0.00
12	GLU	0.00
13	VAL	0.00
14	VAL	0.00
15	GLY	0.00
16	LYS	0.00
17	THR	0.00
18	VAL	0.00
19	ASP	0.00
20	GLN	0.05
21	ALA	0.00
22	ARG	0.07
23	GLU	0.08
24	TYR	0.05
25	PHE	0.00
26	THR	0.08
27	LEU	0.05
28	HIS	0.06
29	TYR	0.03
30	PRO	0.03
31	GLN	0.09
32	TYR	0.09
33	ASP	0.17
34	VAL	0.03
35	TYR	0.08
36	PHE	0.01
37	LEU	0.00
38	PRO	0.00
39	GLU	0.00

40	GLY	0.05
41	SER	0.35
42	PRO	0.96
43	VAL	0.25
44	THR	0.96
45	LEU	0.67
46	ASP	0.98
47	LEU	0.97
48	ARG	0.17
49	TYR	0.24
50	ASN	0.12
51	ARG	0.00
52	VAL	0.00
53	ARG	0.1
54	VAL	0.00
55	PHE	0.26
56	TYR	0.05
57	ASN	0.04
58	PRO	0.01
59	GLY	0.01
60	THR	0.01
61	ASN	0.01
62	VAL	0.02
63	VAL	0.00
64	ASN	0.01
65	HIS	0.01
66	VAL	0.02
67	PRO	0.00
68	HIS	0.01
69	VAL	0.00
70	GLY	0.01

40	GLY	0.44
41	SER	0.71
42	PRO	0.70
43	VAL	0.68
44	THR	0.55
45	LEU	0.58
46	ASP	0.50
47	LEU	0.78
48	ARG	0.59
49	TYR	1.00
50	ASN	0.53
51	ARG	0.32
52	VAL	0.04
53	ARG	0.61
54	VAL	0.05
55	PHE	0.46
56	TYR	0.63
57	ASN	0.11
58	PRO	0.36
59	GLY	0.20
60	THR	0.11
61	ASN	0.38
62	VAL	0.04
63	VAL	0.03
64	ASN	0.17
65	HIS	0.37
66	VAL	0.37
67	PRO	0.14
68	HIS	0.62
69	VAL	0.54
70	GLY	0.37

40	GLY	0.00
41	SER	0.00
42	PRO	0.17
43	VAL	0.23
44	THR	0.19
45	LEU	0.28
46	ASP	0.25
47	LEU	0.65
48	ARG	1.00
49	TYR	0.43
50	ASN	0.31
51	ARG	0.31
52	VAL	0.00
53	ARG	0.22
54	VAL	0.00
55	PHE	0.05
56	TYR	0.00
57	ASN	0.00
58	PRO	0.00
59	GLY	0.00
60	THR	0.00
61	ASN	0.00
62	VAL	0.00
63	VAL	0.00
64	ASN	0.00
65	HIS	0.00
66	VAL	0.00
67	PRO	0.00
68	HIS	0.18
69	VAL	0.08
70	GLY	0.26

Appendix C: Python Script for Logistic Regression

```
#!/usr/bin/env python

# imports
import pandas as pd
import numpy as np
import statsmodels.api as sm

# set table of data
aucframe= pd.DataFrame({})

#  create logistic regression function
def log_reg_nox():
   col_names = ['residue', 'predus', 'ispred', 'dockpred', 'annotated']
   # load dataset
   df =
pd.read_csv("/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files
/Logistic_regresion_corrected/noxdata.csv", header=None, names=col_names)
   df.isnull().any()
   data = df.fillna(method='ffill')
   feature_cols = ['predus','ispred','dockpred']
   protein = data.residue
   X = data[feature_cols] # Features
   y = data.annotated # Target variable
   x = sm.add_constant(X)
   logit_model=sm.Logit(y,x)
   result=logit_model.fit()
   print(result.summary2())
   coefficients = result.params
   benchmark=
pd.read_csv('/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files/
Logistic_regresion_corrected/benchmarkdata.csv', header=None, names=col_names)
   protein= benchmark.residue
   predusval = benchmark.predus
   ispredval = benchmark.ispred
   dockpred = benchmark.dockpred
   predcoef = coefficients[1]
   ispredcoef = coefficients[2]
   dockpredcoef= coefficients[3]
   val = (coefficients[0] + predcoef * predusval + ispredval* ispredcoef+dockpred *
dockpredcoef)*(-1)
   exponent = np.exp(val)
   pval = (1/(1+exponent))
   results = pd.DataFrame({"residue": protein, "prediction value": pval})
```

```
path="/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files/Logist
ic_regresion_corrected/predictionvalues/predus_ispred_dockpred/benchmarkpredictionvalues.cs
v"
    results.to_csv(path,sep=",", index=False, header=True)
log_reg_nox()
def log_reg_bnch():
    col_names = ['residue', 'predus', 'ispred', 'dockpred', 'annotated']

# load dataset
    df =
pd.read_csv("/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files
/Logistic_regresion_corrected/benchmarkdata.csv", header=None, names=col_names)
    df.isnull().any()
    data = df.fillna(method='ffill')
    feature_cols = ['predus','ispred','dockpred']
    protein = data.residue
    X = data[feature_cols] # Features
    y = data.annotated # Target variable

    # fit the model with data
    x = sm.add_constant(X)
    logit_model=sm.Logit(y,x)
    result=logit_model.fit()
    print(result.summary2())
    coefficients = result.params
    print(coefficients)

# prediction value
    nox=
pd.read_csv('/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files/
Logistic_regresion_corrected/noxdata.csv', header=None, names=col_names)
    protein= nox.residue
    protein= nox.residue
    predusval = nox.predus
    ispredval = nox.ispred
    dockpred = nox.dockpred
    predcoef = coefficients[1]
    ispredcoef = coefficients[2]
    dockpredcoef= coefficients[3]
    val = (coefficients[0] + predcoef * predusval +  ispredval* ispredcoef+dockpred *
dockpredcoef )*(-1)
    exponent = np.exp(val)
    pval = (1/(1+exponent))
    results = pd.DataFrame({"residue": protein, "prediction value": pval})
```

```
path="/Users/evanedelstein/Desktop/Research_Evan/Raji_Summer2019_atom/Data_Files/Logist
ic_regresion_corrected/predictionvalues/predus_ispred_dockpred/noxpredictionvalues.csv"
    results.to_csv(path,sep=",", index=False, header=True)
log_reg_bnch()
def log_reg_nox_ispred_dockpred():
    col_names = ['residue', 'predus', 'ispred', 'dockpred', 'annotated']
```

Appendix D: Perl6 Script for Calculating F_1 Score

```
#!/usr/bin/perl

# declaration of file names as variables
my $DBMark_surface_file =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/Dynamic_Cutoff/DBMark_
surfaceres.csv>;
my $NOX_surface_file =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/Dynamic_Cutoff/NOX_surf
aceres.csv>;
my $Dbmark_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_DBMark
_data_sorted/>;
my $NOX_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_NOX_da
ta_sorted/>;
my $Dbmark_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/Dbmark_Annotate
d_Residues>;
my $NOX_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/NOX_Annotated_
Residues>;
#creating data table file
my $F_Score_File =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/Fscores_Results/Ispred.cutoff.F
scores.csv>;

# initiating values for global dataset calculations
my $TP_Dbmark_sum = 0;
my $TP_NOX_sum = 0;
my $Pred_Dbmark_sum = 0;
my $Pred_NOX_sum = 0;
my $Ann_Dbmark_sum = 0;
my $Ann_NOX_sum = 0;

# creating array of query proteins' dynamic cutoffs
if (my $F_Score_Data = open $F_Score_File, :w) {
  $F_Score_Data.print("Database", ",", "Protein_ID", ",", "TP", ",", "Interface_residues", ",",
"Dynamic_Cutoff", ",", "Precision", ",", "Recall", ",","F_Score", "\n");
}
my %Dbmark_cutoff;
my @Dbmark_cutoff;
for $DBMark_surface_file.IO.lines -> $line {
  my $protein = split(',', $line)[0];
  my $cutoff = split(',', $line)[2];
  @Dbmark_cutoff.push: $protein;
  @Dbmark_cutoff.push: $cutoff;
}
  %Dbmark_cutoff = @Dbmark_cutoff;

# looping through annotated residue files
for dir($Dbmark_annotateddir) -> $file {
    my @annotatedres;
    my $Dbmark_filename = split('/', $file.IO.path)[7];
    my $Dbmark_protein = split('_', $Dbmark_filename)[0];
    say $Dbmark_protein;
    my $cutoff_res = %Dbmark_cutoff{$Dbmark_protein};
    my int $cutoff_res_int = +$cutoff_res;
    say $cutoff_res_int;
    my $Dbmark_protein_ispred = "$Dbmark_preddir$Dbmark_protein.ispred_sorted.csv";
    for $file.IO.lines -> $line {
      my ($annres_num, $annres) = $line.split('_');
      @annotatedres.push: $annres_num;
    }
    my $N = @annotatedres.elems;

# assigning predicted residues to an array
    my @predres;
    my $predfile = open $Dbmark_protein_ispred, :r;
    my $preddata = $predfile.slurp;
    for $preddata.lines($cutoff_res_int) -> $prediction {
      my ($predres_num, $predres) = $prediction.split(', ');
      @predres.push: $predres_num;
    }
    $predfile.close;

# comparing annotated and predicted residues to determine true positives
    my @TPres;
    my %lookup = map { $_ => 1 }, @annotatedres;
    for (@predres) -> $res {
       if (%lookup{ $res }) {
       @TPres.push: $res;
       }
     }
    say "My tpres = ", @TPres;
    my $TP = @TPres.elems;
    my $Recall = $TP/$N;
    my $Precision = $TP/$cutoff_res_int;

# calculating precision, recall, and F1 scores
    my $F_Score;
    if ($TP == 0) {
      $F_Score = 0;
    } else {
      $F_Score = (2 * $Recall * $Precision)/($Recall + $Precision);
    }
    say $F_Score;
    if (my $F_Score_Data = open $F_Score_File, :a) {
      $F_Score_Data.print("DBMark", ",", $Dbmark_protein, ",", $TP, ",", $N, ",",
$cutoff_res_int, ",", $Precision, ",", $Recall, ",", $F_Score, "\n");
    }
    $TP_Dbmark_sum += $TP;
    $Pred_Dbmark_sum += $cutoff_res_int;
    $Ann_Dbmark_sum += $N;
  }

# repeating process for proteins in NOX database
  my %NOX_cutoff;
  my @NOX_cutoff;
  for $NOX_surface_file.IO.lines -> $line {
    my $protein = split(',', $line)[0];
    my $cutoff = split(',', $line)[2];
    @NOX_cutoff.push: $protein;
    @NOX_cutoff.push: $cutoff;
  }
    %NOX_cutoff = @NOX_cutoff;
    for dir($NOX_annotateddir) -> $file {
        my @annotatedres;
        my $NOX_filename = split('/', $file.IO.path)[7];
        my $NOX_protein = split('_', $NOX_filename)[0];
        say $NOX_protein;
        my $cutoff_res = %NOX_cutoff{$NOX_protein};
        my int $cutoff_res_int = +$cutoff_res;
        say $cutoff_res_int;
        my $NOX_protein_ispred = "$NOX_preddir$NOX_protein.ispred_sorted.csv";
        for $file.IO.lines -> $line {
          my ($annres_num, $annres) = $line.split('_');
          @annotatedres.push: $annres_num;
        }
        my $N = @annotatedres.elems;
        my @predres;
        my $predfile = open $NOX_protein_ispred, :r;
        my $preddata = $predfile.slurp;
        for $preddata.lines($cutoff_res_int) -> $prediction {
          my ($predres_num, $predres) = $prediction.split(', ');
            @predres.push: $predres_num;
        }
        $predfile.close;
        my @TPres;
       my %lookup = map { $_ => 1 }, @annotatedres;
       for (@predres) -> $res {
          if (%lookup{ $res }) {
          @TPres.push: $res;
          }
        }
        my $TP = @TPres.elems;
        my $Recall = $TP/$N;
        my $Precision = $TP/$cutoff_res_int;
        my $F_Score;
        if ($TP == 0) {
          $F_Score = 0;
        } else {
          $F_Score = (2 * $Recall * $Precision)/($Recall + $Precision);
        }
        say $F_Score;
        if (my $F_Score_Data = open $F_Score_File, :a) {
          $F_Score_Data.print("NOX", ",", $NOX_protein, ",", $TP, ",", $N, ",", $cutoff_res_int,
",", $Precision, ",", $Recall, ",", $F_Score, "\n");
        }
        $TP_NOX_sum += $TP;
        $Pred_NOX_sum += $cutoff_res_int;
        $Ann_NOX_sum += $N;
      }

# calculating global precision, recall, and F1 scores for Docking Benchmark NOX databases
      my $Global_Dbmark_Precision = $TP_Dbmark_sum/$Pred_Dbmark_sum;
      my $Global_Dbmark_Recall = $TP_Dbmark_sum/$Ann_Dbmark_sum;
      my $Global_Dbmark_F_Score = (2 * $Global_Dbmark_Recall *
$Global_Dbmark_Precision)/($Global_Dbmark_Recall + $Global_Dbmark_Precision);
      my $Global_NOX_Precision = $TP_NOX_sum/$Pred_NOX_sum;
```

```
    my $Global_NOX_Recall = $TP_NOX_sum/$Ann_NOX_sum;
    my $Global_NOX_F_Score = (2 * $Global_NOX_Recall *
$Global_NOX_Precision)/($Global_NOX_Recall + $Global_NOX_Precision);
    my $F_Score_Global =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/Fscores_Results/Ispred.cutoff.F
scores.Totals.csv>;
    if (my $F_Score_Data_Totals = open $F_Score_Global, :w) {
      $F_Score_Data_Totals.print("DBmark_Global_F_Score", ",", "NOX_Global_F_Score",
"\n");
      $F_Score_Data_Totals.print($Global_Dbmark_F_Score, ",", $Global_NOX_F_Score);
    }
```

Appendix E: Perl6 Script for Calculating MCC Score

```
#!/usr/bin/perl

# declaration of file names as variables
my $DBMark_surface_file =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/Dynamic_Cutoff/DBMark_
surfaceres.csv>;
my $NOX_surface_file =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/Dynamic_Cutoff/NOX_surf
aceres.csv>;
my $Dbmark_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_DBMark
_data_sorted/>;
my $NOX_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_NOX_da
ta_sorted/>;
my $Dbmark_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/Dbmark_Annotate
d_Residues>;
my $NOX_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/NOX_Annotated_
Residues>;

# creating data table file
my $MCC_File =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/MCC_Results/ISPRED/Ispred.
MCC.csv>;
```

```
# initiating values for global dataset calculations
my $TP_Dbmark_sum = 0;
my $TP_NOX_sum = 0;
my $FP_Dbmark_sum = 0;
my $FP_NOX_sum = 0;
my $TN_Dbmark_sum = 0;
my $TN_NOX_sum = 0;
my $FN_Dbmark_sum = 0;
my $FN_NOX_sum = 0;
if (my $MCC_Data = open $MCC_File, :w) {
  $MCC_Data.print("Database", ",", "Protein_ID", ",", "Interface_residues", ",",
"Dynamic_Cutoff", ",", "Sequence_residues", ",", "TP", ",", "FP", ",", "TN", ",", "FN",
",","MCC", "\n");
}

# creating array of query proteins' dynamic cutoffs
my %Dbmark_cutoff;
my @Dbmark_cutoff;
for $DBMark_surface_file.IO.lines -> $line {
  my $protein = split(',', $line)[0];
  my $cutoff = split(',', $line)[2];
  @Dbmark_cutoff.push: $protein;
  @Dbmark_cutoff.push: $cutoff;
}
%Dbmark_cutoff = @Dbmark_cutoff;

# looping through annotated residue files
for dir($Dbmark_annotateddir) -> $file {
  my @annotatedres;
  my $Dbmark_filename = split('/', $file.IO.path)[7];
  my $Dbmark_protein = split('_', $Dbmark_filename)[0];
  say $Dbmark_protein;
  my $cutoff_res = %Dbmark_cutoff{$Dbmark_protein};
  my int $cutoff_res_int = +$cutoff_res;
  say $cutoff_res_int;
  my $Dbmark_protein_ispred = "$Dbmark_preddir$Dbmark_protein.ispred_sorted.csv";
  for $file.IO.lines -> $line {
    my ($annres_num, $annres) = $line.split('_');
    @annotatedres.push: $annres_num;
  }
  my $N = @annotatedres.elems;

# assigning predicted residues to an array
  my @predres;
  my @seqres;
  my $predfile = open $Dbmark_protein_ispred, :r;
  my $preddata = $predfile.slurp;
  for $preddata.lines($cutoff_res_int) -> $prediction {
    my ($predres_num, $predval) = split ', ', $prediction;
    @predres.push: $predres_num;
  }
  for $preddata.lines -> $prediction {
    my ($predres_num, $predval) = split ', ', $prediction;
    @seqres.push: $predres_num;
  }
  $predfile.close;

# comparing annotated and predicted residues to determine true positives
  my @TPres;
  my %lookup = map { $_ => 1 }, @annotatedres;
  for (@predres) -> $res {
     if (%lookup{ $res }) {
     @TPres.push: $res;
     }
   }

# calculating confusion matrix values and MCC score
  my $Seqres = @seqres.elems;
  my $neg = $Seqres - $cutoff_res_int;
  my $TP = @TPres.elems;
  my $FP = $cutoff_res_int - $TP;
  my $FN = $N - $TP;
  my $TN = $neg - $FN;
  my $MCC_num = ($TP * $TN) - ($FP * $FN);
  my $MCC_denom = sqrt(($TP + $FN) * ($TP + $FP) * ($TN + $FP) * ($TN + $FN));
  my $MCC = $MCC_num/$MCC_denom;
  say "My MCC = ", $MCC;
  if (my $MCC_Data = open $MCC_File, :a) {
    $MCC_Data.print("DBMark", ",", $Dbmark_protein, ",", $N, ",", $cutoff_res_int, ",",
$Seqres, ",", $TP, ",", $FP, ",", $TN, ",", $FN, ",",$MCC, "\n");
  }
  $TP_Dbmark_sum += $TP;
  $FP_Dbmark_sum += $FP;
  $TN_Dbmark_sum += $TN;
  $FN_Dbmark_sum += $FN;
}

# repeating process for proteins in NOX database
my %NOX_cutoff;
my @NOX_cutoff;
for $NOX_surface_file.IO.lines -> $line {
  my $protein = split(',', $line)[0];
  my $cutoff = split(',', $line)[2];
  @NOX_cutoff.push: $protein;
  @NOX_cutoff.push: $cutoff;
}
  %NOX_cutoff = @NOX_cutoff;
  for dir($NOX_annotateddir) -> $file {
      my @annotatedres;
      my $NOX_filename = split('/', $file.IO.path)[7];
      my $NOX_protein = split('_', $NOX_filename)[0];
      say $NOX_protein;
      my $cutoff_res = %NOX_cutoff{$NOX_protein};
      my int $cutoff_res_int = +$cutoff_res;
      say $cutoff_res_int;
      my $NOX_protein_ispred = "$NOX_preddir$NOX_protein.ispred_sorted.csv";
      for $file.IO.lines -> $line {
        my ($annres_num, $annres) = $line.split('_');
        @annotatedres.push: $annres_num;
      }
      my $N = @annotatedres.elems;
      my @predres;
      my @seqres;
      my $predfile = open $NOX_protein_ispred, :r;
      my $preddata = $predfile.slurp;
      for $preddata.lines($cutoff_res_int) -> $prediction {
        my ($predres_num, $predval) = split ', ', $prediction;
            @predres.push: $predres_num;
        }
        for $preddata.lines -> $prediction {
          my ($predres_num, $predval) = split ', ', $prediction;
          @seqres.push: $predres_num;
        }
        $predfile.close;
        say "My predres = ", @predres;
        my @TPres;
       my %lookup = map { $_ => 1 }, @annotatedres;
       for (@predres) -> $res {
          if (%lookup{ $res }) {
          @TPres.push: $res;
          }
        }
        my $Seqres = @seqres.elems;
        my $neg = $Seqres - $cutoff_res_int;
        my $TP = @TPres.elems;
        my $FP = $cutoff_res_int - $TP;
        my $FN = $N - $TP;
        my $TN = $neg - $FN;
        my $MCC_num = ($TP * $TN) - ($FP * $FN);
        my $MCC_denom = sqrt(($TP + $FN) * ($TP + $FP) * ($TN + $FP) * ($TN + $FN));
        my $MCC = $MCC_num/$MCC_denom;
        say "My MCC = ", $MCC;
        if (my $MCC_Data = open $MCC_File, :a) {
          $MCC_Data.print("NOX", ",", $NOX_protein, ",", $N, ",", $cutoff_res_int, ",", $Seqres,
",", $TP, ",", $FP, ",", $TN, ",", $FN, ",",$MCC, "\n");
        }
        $TP_NOX_sum += $TP;
        $FP_NOX_sum += $FP;
        $TN_NOX_sum += $TN;
        $FN_NOX_sum += $FN;
      }

# calculating global confusion matrix values and MCC scores for Docking Benchmark NOX
databases
    my $Dbmark_MCC_num = ($TP_Dbmark_sum * $TN_Dbmark_sum) - ($FP_Dbmark_sum *
$FN_Dbmark_sum);
    my $Dbmark_MCC_denom = sqrt(($TP_Dbmark_sum + $FN_Dbmark_sum) *
($TP_Dbmark_sum + $FP_Dbmark_sum) * ($TN_Dbmark_sum + $FP_Dbmark_sum) *
($TN_Dbmark_sum + $FN_Dbmark_sum));
    my $Dbmark_MCC = $Dbmark_MCC_num/$Dbmark_MCC_denom;
    my $NOX_MCC_num = ($TP_NOX_sum * $TN_NOX_sum) - ($FP_NOX_sum *
$FN_NOX_sum);
    my $NOX_MCC_denom = sqrt(($TP_NOX_sum + $FN_NOX_sum) * ($TP_NOX_sum +
$FP_NOX_sum) * ($TN_NOX_sum + $FP_NOX_sum) * ($TN_NOX_sum +
$FN_NOX_sum));
    my $NOX_MCC = $NOX_MCC_num/$NOX_MCC_denom;
    say "My Dbmark_MCC = ", $Dbmark_MCC;
    say "My NOX_MCC = ", $NOX_MCC;
    my $MCC_Total =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/MCC_Results/ISPRED/Ispred.
MCC.Totals.csv>;
    if (my $MCC_tot = open $MCC_Total, :w) {
      $MCC_tot.print("Dbmark_MCC", ",", "NOX_MCC", "\n", $Dbmark_MCC, ",",
$NOX_MCC);
    }
```

Appendix F: Per6 Scripts for Calculating Evaluation Metrics of ROC and PR Curves

```
#!/usr/bin/perl

# declaration of file names as variables
my $Dbmark_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_DBMark
_data_sorted/>;
my $NOX_preddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Data_Files/ISPRED/ISPRED_NOX_da
ta_sorted/>;
my $Dbmark_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/Dbmark_Annotate
d_Residues>;
my $NOX_annotateddir =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Annotated_Residues/NOX_Annotated_
Residues>;

# creating file w/ data table of gloabal TPR/FPR values at each threshold
my $ROC_File =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/ROC_Curves_Results/ROC_D
ata/ISPRED/Ispred.ROC.thresholds.csv>;
if (my $ROC_Data = open $ROC_File, :w) {
  $ROC_Data.print("Threshold", ",", "Global_Dbmark_TPR", ",", "Global_Dbmark_FPR", ",",
"Predicted_Total", ",", "Annotated_Total", ",", "Global_NOX_TPR", ",", "Global_NOX_FPR",
",", "Global_Total_TPR", ",", "Global_Total_FPR", "\n");
}

#creating file w/ data table of TPR/FPR values for each protein at the 0 threshold mark
my $Zero_Threshold_File =
</Users/mordechaiwalder/Desktop/Research_Mordechai/Results/ROC_Curves_Results/ROC_D
ata/ISPRED/Ispred.ROC.zero_threshold.proteins.csv>;
if (my $Zero_Data = open $Zero_Threshold_File, :w) {
  $Zero_Data.print("Protein", ",", "TP", ",", "Annotated_Residues", ",", "TPR", ",", "FP", ",",
"Non-Annotated_Residues", ",", "FPR", "\n");
}

#looping through threshold values
for 0.00, 1.01, 0.01 -> $start, $stop, $inc
 {
   my @seq = flat ($start, *+$inc ... $stop);
   for (@seq) -> $threshold {
     #initializing values
     my $TP_Dbmark_sum = 0;
     my $TP_NOX_sum = 0;
     my $FP_Dbmark_sum = 0;
     my $FP_NOX_sum = 0;
     my $Ressum_Dbmark = 0;
     my $Ressum_NOX = 0;
     my $Neg_Dbmark_sum = 0;
     my $Neg_NOX_sum = 0;
     print $threshold;
     print "\n";

     # looping through Docking Benchmark proteins
     for dir($Dbmark_annotateddir) -> $file {
         my @annotatedres;
         my $Dbmark_filename = split('/', $file.IO.path)[7];
         my $Dbmark_protein = split('_', $Dbmark_filename)[0];
         say $Dbmark_protein;
         my $Dbmark_protein_ispred = "$Dbmark_preddir$Dbmark_protein.ispred_sorted.csv";
         for $file.IO.lines -> $line {
           my ($annres_num, $annres) = $line.split('_');
           @annotatedres.push: $annres_num;
         }
         my $N = @annotatedres.elems;
         my @predres;
         my @seqres;
         my $predfile = open $Dbmark_protein_ispred, :r;
         my $preddata = $predfile.slurp;
         for $preddata.lines -> $prediction {
           my ($predres_num, $predval) = split ', ', $prediction;
           @seqres.push: $predres_num;
           if ($predval >= $threshold) {
             @predres.push: $predres_num;
          }
         }
         $predfile.close;
         my @TPres;
        my %lookup = map { $_ => 1 }, @annotatedres;
        for (@predres) -> $res {
           if (%lookup{ $res }) {
           @TPres.push: $res;
            }
         }
         my $pred = @predres.elems;
         my $TP = @TPres.elems;
         my $Seqres = @seqres.elems;
         my $TPR = $TP/$N;
         my $FP = $pred - $TP;
         my $neg = $Seqres - $N;
         my $FPR = $FP/$neg;
        if ($threshold == 0) {
          if (my $Zero_Data = open $Zero_Threshold_File, :a) {
            $Zero_Data.print($Dbmark_protein, ",", $TP, ",", $N, ",", $TPR, ",", $FP, ",", $neg, ",",
     $FPR, "\n");
           }
        }
        $TP_Dbmark_sum += $TP;
        $FP_Dbmark_sum += $FP;
        $Ressum_Dbmark += $N;
        $Neg_Dbmark_sum += $neg;
        }
        my $Seq_Dbmark_sum = $Neg_Dbmark_sum + $Ressum_Dbmark;
        my $Pred_Dbmark_sum = $TP_Dbmark_sum + $FP_Dbmark_sum;
        say "my TP_Dbmark_sum = ", $TP_Dbmark_sum;
        say "my FP_Dbmark_sum = ", $FP_Dbmark_sum;
        say "my Ressum_Dbmark = ", $Ressum_Dbmark;
        say "my Neg_Dbmark_sum = ", $Neg_Dbmark_sum;
        my $Global_Dbmark_TPR = $TP_Dbmark_sum/$Ressum_Dbmark;
        my $Global_Dbmark_FPR = $FP_Dbmark_sum/$Neg_Dbmark_sum;
        say "my Global_Dbmark_TPR = ", $Global_Dbmark_TPR;
        say "my Global_Dbmark_FPR = ", $Global_Dbmark_FPR;
        #looping through NOX
        for dir($NOX_annotateddir) -> $file {
           my @annotatedres;
           my $NOX_filename = split('/', $file.IO.path)[7];
           my $NOX_protein = split('_', $NOX_filename)[0];
           say $NOX_protein;
           my $NOX_protein_ispred = "$NOX_preddir$NOX_protein.ispred_sorted.csv";
           for $file.IO.lines -> $line {
             my ($annres_num, $annres) = $line.split('_');
             @annotatedres.push: $annres_num;
           }
           my $N = @annotatedres.elems;
           my @predres;
           my @seqres;
           my $predfile = open $NOX_protein_ispred, :r;
           my $preddata = $predfile.slurp;
           for $preddata.lines -> $prediction {
             my ($predres_num, $predval) = split ', ', $prediction;
             @seqres.push: $predres_num;
             if ($predval >= $threshold) {
               @predres.push: $predres_num;
            }
           }
           $predfile.close;
           my @TPres;
```

```
        my %lookup = map { $_ => 1 }, @annotatedres;
        for (@predres) -> $res {
          if (%lookup{ $res }) {
          @TPres.push: $res;
          }
         }
         #say "My tpres = ", @TPres;
         my $pred = @predres.elems;
         my $TP = @TPres.elems;
         my $Seqres = @seqres.elems;
         my $TPR = $TP/$N;
         my $FP = $pred - $TP;
         my $neg = $Seqres - $N;
         my $FPR = $FP/$neg;
       if ($threshold == 0) {
         if (my $Zero_Data = open $Zero_Threshold_File, :a) {
           $Zero_Data.print($NOX_protein, ",", $TP, ",", $N, ",", $TPR, ",", $FP, ",", $neg, ",",
$FPR, "\n");
         }
       }
       $TP_NOX_sum += $TP;
       $FP_NOX_sum += $FP;
       $Ressum_NOX += $N;
       $Neg_NOX_sum += $neg;
       }
       my $Seq_NOX_sum = $Neg_NOX_sum + $Ressum_NOX;
       my $Pred_NOX_sum = $TP_NOX_sum + $FP_NOX_sum;
       say "my TP_NOX_sum = ", $TP_NOX_sum;
       say "my FP_NOX_sum = ", $FP_NOX_sum;
       say "my Ressum_NOX = ", $Ressum_NOX;
       say "my Neg_NOX_sum = ", $Neg_NOX_sum;
       my $Global_NOX_TPR = $TP_NOX_sum/$Ressum_NOX;
       my $Global_NOX_FPR = $FP_NOX_sum/$Neg_NOX_sum;
       say "my Global_NOX_TPR = ", $Global_NOX_TPR;
       say "my Global_NOX_FPR = ", $Global_NOX_FPR;
       my $TP_Total_sum = $TP_NOX_sum + $TP_Dbmark_sum;
       my $FP_Total_sum = $FP_NOX_sum + $FP_Dbmark_sum;
       my $Ressum_Total = $Ressum_NOX + $Ressum_Dbmark;
       my $Neg_Total_sum = $Neg_NOX_sum + $Neg_Dbmark_sum;
       my $Global_Total_TPR = $TP_Total_sum/$Ressum_Total;
       my $Global_Total_FPR = $FP_Total_sum/$Neg_Total_sum;
       say "my Global_Total_TPR = ", $Global_Total_TPR;
       say "my Global_Total_FPR = ", $Global_Total_FPR;
      if (my $ROC_Data = open $ROC_File, :a) {
```

```
      $ROC_Data.print($threshold, ",", $Global_Dbmark_TPR, ",", $Global_Dbmark_FPR, ",",
$Pred_Dbmark_sum, ",", $Ressum_Dbmark, ",", $Global_NOX_TPR, ",", $Global_NOX_FPR,
",", $Global_Total_TPR, ",", $Global_Total_FPR, "\n");
      }
    }
}
```