PBRpredict-Suite: A Suite of Models to Predict Peptide Recognition Domain Residues from Protein Sequence (Supplementary Document)

Sumaiya Iqbal and Md Tamjidul Hoque

March 18, 2018

1 Peptide Recognition Domains in the Dataset

A wide range of PRDs were included in the rcp644 dataset of receptor chains that mediate peptide-protein interactions, for example, the Major Histocompatibility Complex (MHC I and II) domain can recognize peptide fragments derived from the pathogen. The PDZ domain generally binds to C-terminal peptide-motifs. The Src Homology 2 (SH2) and Phospho-Tyrosine Binding (PTB) domains recognize phosphorylation of tyrosine (pTyr or pY). The PTB domain can bind to the N-P-x-Y motif as well. The Src Homology 3 (SH3) domain binds to Pro-rich motifs and peptide-motifs, such as R-x-x-K. The 14-3-3, WW, Polo-box, BRCA1 C Terminus (BRCT), Forkhead-associated (FHA) domains recognize different type phosphorylation or post-translational modifications (PTMs) of threonine (pThr or pT) and serine (pSer or pS). The chromatin organization modifier (Chromo), Bromodomain and Tudor domain bind to methylated or acetylated peptides, such as Tudor domain can recognize PTMs on lysine (meLys or meK) and arginine (meArg or meR) by methylation. Chromo domain can also recognize meLys and Bromo domain recognize PTMs on lysine by acetylation (acLys or acK). The Enzyme/inhibitor complexes with hydrolase, kinase, isomerase, phosphatase, protease and so on. Further, we included antibody-antigen, amyloid fibrils, membrane or transmembrane protein and nuclear receptor complexes in the dataset. The count of sequences with different domains and the distribution of positive (peptide-binding) and negative (non-binding) class type residues of those sequences are reported in **Table 1**. The distributions reported in **Table 1** show the percentage of different class type residues before smoothing (see main article, **Section 2.2.**)

Table 1: Peptide recognition domains in our dataset.

		Count of Sequences and Distribution of Residue Types	
Peptide Recognition Domains (PRDs)	Full Set (%b/%n)*	Training Set (Fold 1 + Fold 2)	Test Set
MHC I/II	81 (11.9/88.1)	60 (30 + 30)	21
PDZ	37 (16.4/83.6)	27 (14 + 13)	10
SH2, PTB	59 (17.2/82.8)	43 (22 +21)	16
SH3	54 (16.8/83.2)	40 (20 + 20)	14
14-3-3, WW, Polo-Box BRCT, FHA	72 (11.5/88.5)	52 (28 + 24)	20
Tudor, Chromo, Bromo	54 (13.0/87.0)	40 (21 + 19)	14
Enzyme/Inhibitor	109 (12.1/87.9)	81 (41 + 40)	28
Antibody/Antigen	88	65 (33 + 32)	23
(Trans)Membrane Amyloid fibrils	(10.7/89.3)		
Nuclear, others	90 (10.4/89.6)	67 (34 + 33)	23
Total	644 (12.5/87.5)	475 (243 + 232)	169

∗ %b/%n indicates the proportional distribution of peptide-binding ('b') and non-binding ('n') residues beside the count of different domain protein sequences.)

2 Evaluation Criteria

The binary classification output of the PBRpredict is evaluated and compared using the measures listed below. Here, recall is a measure to identify a predictors completeness in classifying the positive class and precision measures a predictors exactness. Therefore, the harmonic mean of recall and precision called F1 score measure a classifiers overall correctness. The miss rate and fall-out rate measure two complementary types of incorrect predictions, respectively the misclassification of binding residue as non-binding and non-binding residue as binding. MCC is considered as another balanced measure to evaluate binary classification.

- True positive, TP = Correctly predicted peptide-binding residues
- True negative, TN = Correctly predicted non-binding residues
- False positive, FP = Incorrectly predicted peptide-binding residues
- False negative, FN = Incorrectly predicted non-binding residues

- Recall/Sensitivity/True Positive Rate, $TPR = \frac{TP}{TP+FN}$
- Specificity/True Negative Rate, $TNR = \frac{TN}{FP+TN}$
- Fall-out (or over prediction) Rate/False Positive Rate, $FPR = \frac{FP}{FP+TN}$
- Miss Rate/False Negative Rate, $FNR = \frac{FN}{FN+TP}$
- Balanced accuracy (Mean of Specificity and Recall), $ACC = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$
- Precision, $PPV = \frac{TP}{TP+FP}$
- F1 Score (Harmonic mean of precision and Recall) $= \frac{2TP}{2TP+FP+FN}$
- Mathews correlation coefficient, $MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Moreover, Area under ROC curve (AUC) is considered as the measure for probability assignment. We further plotted the ROC curves and Precision-Recall curves. The AUC values and the curves are generated using pROC [Robin *et al.*, 2011] and ROCR packages [Sing *et al.*, 2009] in R.

3 Learning Algorithms

We applied stacked generalization [Wolpert, 1992] to develop the peptide-binding residue predictor (PBRpredict). Stacking is an ensemble technique to minimize the generalization error and has been successfully applied in several machine learning tasks [Frank *et al.*, 2004, Nagi and Bhattacharyya, 2013]. To the best of our knowledge, this study has first explored stacking for identifying the pattern of protein sequence that induces binding with peptides.

Stacking framework involves two-tier learning. The classifiers of the first tier and the second tier are called base-learner and meta-learner respectively. Multiple base-learners are employed in the first tier. In the second tier, the outputs of the base-learners are combined using another meta-learner. Here, the underlying idea is: different base-learners can incorrectly learn different regions of the feature space. A meta-learner is then applied, usually non-linearly, to correct the improper training of the first tier, thus the meat-learner is trained to learn the error of the base-learners. Therefore, it is desirable to use classifiers as base-learners that can generate uncorrelated prediction outputs.

We explored six different machine leaning algorithms as base-learners and logistic regression as meta-learner to combine probability distributions generated at the base-level. The algorithms including the overall setup are briefly discussed below:

1) Support Vector Machine (SVM): We used radial basis function (RBF) kernel SVM [Cortes and Vapnik] as one of the base-learners. SVM is an effective algorithm for binary prediction that minimizes both the empirical classification error in the training phase and generalized error in the test phase. SVM classifies by maximizing the separating hyperplane between two classes and penalizes the instances on the wrong side of the decision boundary using a cost parameter, C. The RBF kernel parameter, λ and the cost, C were optimized to achieve best accuracy using expensive grid search. We conducted this parameter optimization using a smaller subset of random samples, specifically 386 receptor chains out of 644 chains. The best values of the parameters found are, $C = 2^3$ and $\lambda = 2^{-7}$, and used as representative parameter values for the full dataset. The optimal setup of C and makes the SVM model effective for a classification problem with imbalanced dataset and high-dimensional feature space, such as the one attempted here.

2) Random Decision Forest (RDF): The RDF [Cortes and Vapnik] operates by constructing a multitude of decision trees on sub-samples of the dataset and outputs the mean prediction of the decision trees. We used bootstrap samples to construct $1,000$ tress in the forest to develop the RDF ensemble learner.

3) Extra Tree (ET) Classifier: The extremely randomized tree or ET [Geurts *et al.*, 2006] is another ensemble method and is explored as a base-learner here. ET works by constructing randomized decision trees from the original learning sample. The best split is determined randomly from the range of values at each split. We constructed the ET model with $1,000$ tress and the quality of a split was measure by gini impurity index.

4) Gradient Boosting Classifier (GBC): Another learning technique that we used to develop a base-learner is the gradient boosting [Friedman, 2002]. GBC combines weak learners in an iterative fashion into a single learner. We used $1,000$ boosting stages where a regression tree was fit on the negative gradient of the deviance loss function. The learning rate was set to 0.1 and the maximum depth of each regression tree was set to 3. GBC gives robust performance to over-fitting with higher number of boosting stages,

and we observed that $1,000$ stages provided competitive performance for this application.

5) K Nearest Neighbors (KNN): The k nearest neighbors (KNN) classifier [Altman, 1992] operates by learning from the k closest training samples in the feature space around a target point. The classification decision is produced based on the majority votes coming from the neighbors. In this work, the value of k was set to 9 and all the neighbors were weighted uniformly.

6) Bagging (BAG): The bootstrap aggregation or bagging [Breiman, 1996] is another ensemble method, useful for reducing variance in the prediction. Here, bagging classifier was fit on multiple subsets of data with repetitions using $1,000$ decision trees, and the outputs were combined by weighted averaging.

7) Logistic Regression (LogReg): To develop the meta-learner that combines the output probabilities generated by the base-learners, we used logistic regression (LogReg) [Freedman, 2009] with L2 regularization. The LogReg classifier estimates the probability of binding versus non-binding residues based on the confidence of multiple independent base-learners.

We tuned the parameters of SVM and developed the model using libSVM package [Chang and Lin, 2011], while the rest of classifier models were built and tuned using scikit-learn [Pedregosa *et al*, 2011]. We evaluated the performance and analyzed the correlation among prospective base-learners.

4 Feature Importance Estimation

Here, we report the results of the feature importance estimation using extra tree (ET) classifier. ET estimates the feature importance using a method described by Breiman [Breiman *et al*, 1984] which works by maintaining impurity reduction for each feature [Geurts *et al.*, 2006, Louppe *et al*, 2013]. The information gain is attributed to each feature to measure the total decrease of impurity. Finally, the classifier provides an importance value for each feature, known as Gini importance, which was used to rank the features.

Figure 1 presents the ranked features according to the importance values. The importance values can be interpreted as the fraction of the test samples that were correctly classified by that feature. The training and test were done using rcp_tr475 and rcp_ts169 datasets with 60 features and win-

dow size 1. **Figure 1** shows that all the features have greater than zero importance, thus we used all 60 features to develop our predictor.

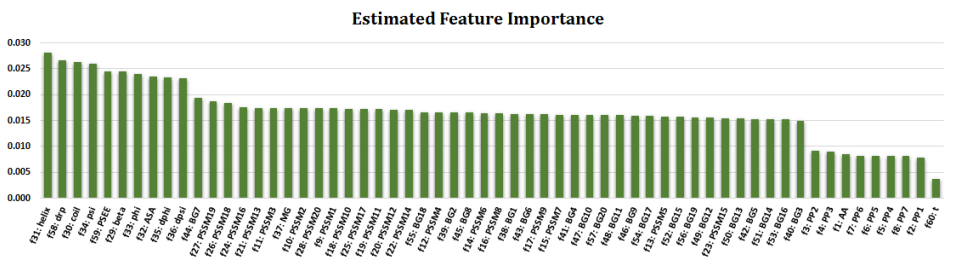


Figure 1: Estimated feature importance by ET classifier. The importance values are shown by green bars. The x-axis shows the features in their abbreviated form according to Section 2.3. Multiple features of same category are indexed by their count, i.e., 20 PSSMs are indexed from 1 to 20.

Figure 1 shows that the structural profile (composed of features: f_{29} (beta), f_{30} (coil), f_{31} (helix), f_{32} (ASA), f_{33} (phi) and f_{34} (psi)), the flexibility profile (composed of features: f_{35} (dphi), f_{36} (dpsi) and f_{58} (drp)) and the energy profile (composed of feature: f_{59} (PSEE)), are the three most dominant feature categories.

To further understand the contributions of the dominant features, we developed separate ET models by subsequently removing the six structural properties, three flexibility-related properties and one energy-based property from the feature set. Therefore, these ET models were developed based on $54(60-\text{structural profile})$, $51(60-\text{structural profile}-\text{flexibility profile})$ and $50(60-\text{structural profile}-\text{flexibility profile}-\text{energy profile})$ features. The performance of these models and the one developed using all 60 features are reported in **Table 2**. The training and test were done using rcp_tr475 and rcp_ts169 datasets, and the window size was set to 1.

The results show that all MCC, F1 score, precision, and recall continues to decrease with the removal of the dominant feature categories. Specifically, we observed no less than 5% decrease in MCC as we removed the structural, flexibility and energy profile. In addition, the F1 score is decreased by 6.2%, 4.4% and 4.8% after removal of the 6 structural properties, 3 flexibility-related properties including disorder probability and backbone angle fluctuations, and 1 position specific estimated energy (PSEE), respectively. These results validate the importance of the top features used to develop our predictor. Moreover, we want to highlight that the PSSM based evolutionary properties, secondary structure and accessible surface area based structural

Table 2: Comparison of different feature sets (training set: rcp_tr475 and test set: rcp_ts169).

Metric	60 Features	54 Features	51 Features	50 Features
MCC	**0.478**	0.454	0.431	0.407
F1 Score	**0.505**	0.474	0.453	0.431
Precision	**0.788**	0.787	0.765	0.739
Recall	**0.372**	0.339	0.322	0.304

Best values are marked in bold.

60 features: all

54 features: all - structural profile

51 features: all - structural profile - flexibility profile

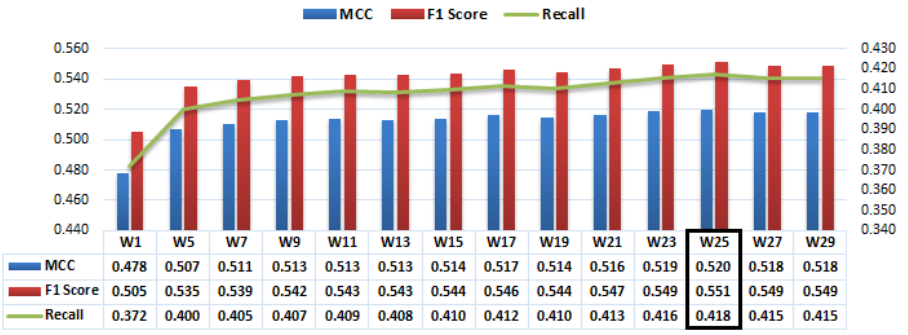
50 features: all - structural profile - flexibility profile - energy profile

properties have been used in the literature [Taherzadeh *et al.*, 2016] for predicting peptide-binidng residues, however, the flexibility and energy based properties have been used for the first time in this work.

5 Window Selection

We searched for a suitable size of the sliding window (W) that determines the number of residues around a target residue, which can mediate the interaction between the target residue and a peptide residue. We developed 15 different models with extra-tree (ET) classifier with 15 different window sizes (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27 and 29). We chose ET classifier for this set of runs as this technique is computationally less costly as well as gives comparable performance (*see* main article, Table 2). The models were trained using rcp_tr475 dataset, and was independently tested using rcp_ts169 dataset.

The result is shown in **Figure 2** in terms of recall, F1 score, MCC and AUC score. We observed that all the scores were improved with the increase of window size, which highlights that inclusion of neighborhood residue information better guides the predictor to learn about a target residue. We have also observed irregular changes in MCC, which were not very significant. However, the scores get flat from window size 19 to higher. Finally, we picked 25 as an optimum value of window as it gave better MCC, F1 score and recall than the adjacent competitors, size 23 and 27. Therefore, we took the features of 12 residues on either side of a target residue while determining if it is interacting or not.



	W1	W5	W7	W9	W11	W13	W15	W17	W19	W21	W23	W25	W27	W29
MCC	0.478	0.507	0.511	0.513	0.513	0.513	0.514	0.517	0.514	0.516	0.519	0.520	0.518	0.518
F1 Score	0.505	0.535	0.539	0.542	0.543	0.543	0.544	0.546	0.544	0.547	0.549	0.551	0.549	0.549
Recall	0.372	0.400	0.405	0.407	0.409	0.408	0.410	0.412	0.410	0.413	0.416	0.418	0.415	0.415

Figure 2: Performance comparison of different sliding window sizes using extra-tree classifier. The MCC, F1 score and recall values are reported. The optimum size of window and the corresponding performance scores are marked by a black rectangle.

6 Finalizing the PBRpredict-Suite Models

In the proposed PBRpredict-Suite, we included three models to predict protein's peptide-binding residues: PBRpredict-strict, PBRpredict-moderate and PBRpredict-flexible, that use different thresholds for classification. In this section, we discuss the development of these 3 different predictor models.

We named the stacked model sM4 with 63 features in the meta-level (see **main article, Section 4.2**) as PBRpredict-strict. This model provided a well-balanced performance when compared with the state-of-the-art predictor which is supported by both statistics (**main article, Section 4.4**) and case-studies (**main article, Section 4.5**). However, we call this model '*strict*' in predicting the positive class (peptide-binding residues) as it resulted in fine false positive rate (fall-out rate/FPR) even at the cost of compromised recall score (TPR). Moreover, we observed that the PBRpredict-strict model provides conservative performance in identifying the binding residues in full-length sequence, relatively longer than the structure-specific shorter sequence, to avoid the false positive predictions or over-prediction (see **Figure 4**). Note that, we included only the structure-specific sequences from PDB in our training dataset, as we needed the experimental structures to extract the interaction information and annotate the protein sequence. However, we intend to design models that can identify peptide-binding sites in sequences with domains that are not known to the training set as well as within the full-length protein sequence with no experimentally solved struc-

ture. Therefore, we tuned our model further to improve the true positive rate (recall/TPR) or positive-class prediction accuracy of our model.

We attempted to relax the classification threshold to recover the positive-class type (peptide-binding) residues that are falsely predicted as negative-class (non-binding). A classification threshold, which is traditionally kept as 0.5, is used to binarize the real-value probabilities generated by a classifier algorithm such as the samples with a probability output $\geq$ threshold is predicted as of positive-class, otherwise labeled as of negative-class. To understand the probabilistic behavior of the learners, we visualized the distributions of the probabilities generated by SVM, GBC, KNN for four different prediction types: true positives (TP), false positive (FP), true negative (TN) and false negative (FN) using the threshold value 0.5. **Figure 3** shows the distribution plots. The plots for SVM, GBC and KNN, shown in **Figure 3(a)–(c)**, were generated from the independent prediction outputs on one fold of the rcp_tr475 set while trained using the other fold (**main article, Section 3.2**). The plot for LogReg in **Figure 3(d)** was generated from the prediction outputs on rcp_ts169 dataset while trained on the full rcp_tr475 set.

Note that, by tuning the threshold, our purpose is to correct the false negative (FN) prediction outputs, represented by the blue curve in **Figure 3**. However, care must be taken in lowering the threshold from 0.5, which may convert the corresponding true negatives (TN) under the green curve into false positives (FP), represented by the red curve. Therefore, we can only increase the accuracy of positive class (peptide-binding residue) prediction or decrease the miss rate at a cost of increased over-prediction rate (false positive rate).

The plots of **Figure 3** again highlight the strength of SVM for this application. The SVM model correctly predicts the highest mass of binding (orange curve) and non-binding residues (green curve) with a high confidence, higher (0.85 1.0) and lower (0.0 0.15) probability values respectively. Moreover, **Figure 3(a)** shows that the SVM model provided the lowest overlap between TNs (correctly predicted non-binding residues) and FNs (incorrectly predicted binding residues) near the threshold margin compared to the other base-learners, GBC and KNN. Therefore, we can lower the threshold of SVM to gain an increase in recall score (TPR) at a cost of lower increase in the false positive rate (FPR). On the other hand, we noticed an opposite scenario from the outputs of KNN in **Figure 3(c)** with almost overlapped density curves for TNs and FNs. Therefore, we can only achieve an increase in TPR at a cost of high FPR. To mention, the curves for GBC in **Figure 3(b)** were better than those of KNN, however worse

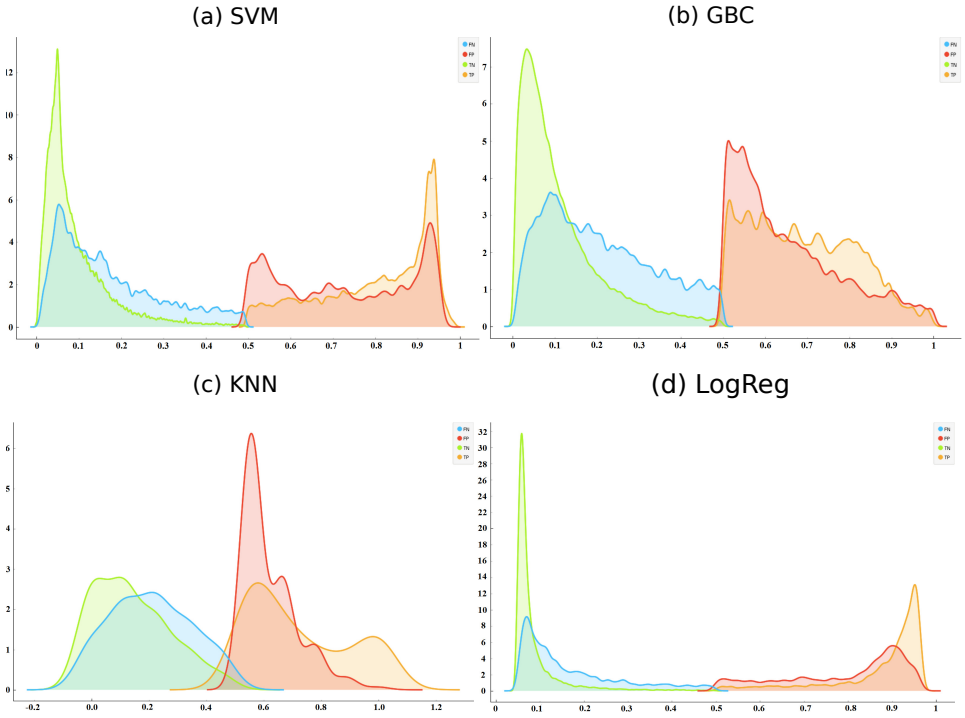


Figure 3: Probability distributions given by (a) SVM, (b) GBC, (c) KNN and (d) LogReg for different prediction type: true positives (TP, orange curve), false positives (FP, red curve), true negatives (TN, green curve) and false negatives (FN, blue curve) using the threshold value 0.5. The x-axis and y-axis show the probabilities generated by the corresponding classifier and the relative density, respectively.

than those given by SVM. **Figure 3(d)** shows that the density curves given by LogReg are even better than SVM in terms of overlap between TNs and FNs near the margin (0.5). It suggests that the application of the meta-learner improved the performance over the base-learners and we can tune the threshold of the meta-learner as well to correct the FNs.

To search for appropriate thresholds, we checked 7 different values, which are 0.45, 0.4, 0.35, 0.3, 0.25, 0.2 and 0.15 for SVM, GBC and KNN. Each classifier was independently evaluated on one fold of the training set while trained by the other fold using these 7 different threshold values (results not shown). This experiment did not result in any certain value of the threshold. For classifiers, i.e., KNN, the recall and balanced accuracy continue to increase with the lower threshold value at a cost of very high over-prediction

which is not desirable. Thus, we finally chose the thresholds according to certain statistics on the probabilities of false negatives (FNs) given by the classifiers as our aim is to correct FNs by assigning a different threshold to segregate the positive and negative class.

We quantified the mean probabilities of FNs ($mean(FN_{pr})$) along with the standard deviations ($std(FN_{pr})$) which are 0.172 ± 0.122 for SVM, 0.209 ± 0.130 for GBC, 0.208 ± 0.138 for KNN and 0.199 ± 0.105 for the LogReg. We checked the median values ($median(FN_{pr})$) as well which are 0.139 for SVM, 0.187 for GBC, 0.222 for KNN and 0.191 for the LogReg. Then, we considered the $mean(FN_{pr}) + std(FN_{pr})$, $mean(FN_{pr})$ and $median(FN_{pr})$ values as different sets of thresholds.

Table 3: Performnace of SVM, GBC and KNN using different thresholds on rcp_ts169 dataset.

Thresholds	TPR	TNR	FPR	FNR	ACC	PPV	F1 Score	MCC
				SVM				
Traditional: 0.5	0.547	**0.959**	**0.041**	0.453	0.753	**0.762**	0.637	**0.579**
$mean(FN_{pr}) + std(FN_{pr})$: 0.3	0.639	0.926	0.074	0.361	0.782	0.672	**0.655**	0.576
$mean(FN_{pr})$: 0.17	0.747	0.854	0.146	0.253	0.800	0.547	0.632	0.538
$median(FN_{pr})$: 0.14	**0.785**	0.821	0.179	**0.215**	**0.803**	0.509	0.618	0.523
				GBC				
Traditional: 0.5	0.373	**0.977**	**0.023**	0.627	0.675	**0.791**	0.507	0.480
$mean(FN_{pr}) + std(FN_{pr})$: 0.34	0.526	0.934	0.066	0.474	0.730	0.652	**0.582**	**0.500**
$mean(FN_{pr})$: 0.21	0.692	0.827	0.173	0.308	0.759	0.486	0.571	0.459
$median(FN_{pr})$: 0.19	**0.722**	0.800	0.200	**0.278**	**0.761**	0.460	0.562	0.448
				KNN				
Traditional: 0.5	0.348	**0.965**	**0.035**	0.652	0.657	**0.701**	0.465	**0.420**
$mean(FN_{pr}) + std(FN_{pr})$: 0.35	0.440	0.926	0.074	0.560	0.683	0.586	**0.502**	0.411
$mean(FN_{pr})$: 0.21	0.744	0.687	0.313	**0.256**	**0.761**	0.360	0.458	0.347
$median(FN_{pr})$: 0.22	**0.744**	0.687	0.313	**0.256**	**0.716**	0.360	0.485	0.347

Best values for each classifier are bold faced.

We report the performances of SVM, GBC and KNN on rcp_ts169 dataset using these modified thresholds in the **Table 3**. The results showed that for all the classifiers, the recall, miss-rate and accuracy (ACC) scores improved with lower threshold values. The models with the traditional threshold (0.5) produced the most balanced performance for SVM and KNN with the highest MCC scores. On the other hand, the models with thresholds equal to $mean(FN_{pr}) + std(FN_{pr})$ provided the best F1 scores for all the classifiers and the best MCC for GBC. Moreover, the fall-out or over-prediction rates with these threshold values were reasonable, specifically no greater than 7.5%. On the other, the $median(FN_{pr})$ values were

lower than the $mean(FN_{pr})$ values for the SVM and GBC. Therefore, the use of $median(FN_{pr})$ values as thresholds resulted in outstanding recall scores, however at a cost of very high fall-out rate which was not desirable. In addition, the performances of KNN models with $mean(FN_{pr})$ and $median(FN_{pr})$ as thresholds were similar. Therefore, we did not consider the $median(FN_{pr})$ value as the threshold in the meta-level.

In the main article (**Table 5**), we report the results of the stacked models with modified threshold values on the rcp_ts169 dataset. The stacked model for which the $mean(FN_{pr}) + std(FN_{pr})$ and the $mean(FN_{pr})$ are used as thresholds for all the base-level and meta-level learners are named as PBRpredict-moderate and PBRpredict-flexible, respectively. The actual threshold values are reported in the footnote of **Table 5**.

The output shows that the PBRpredict-strict with the threshold value of 0.5 resulted in the lowest fall-out rate with the highest MCC score (a balanced measure to assess a binary classifier), however, the recall score was lower as well as the miss rate was higher than those of other models in the suite. In PBRpredict-moderate, the thresholds were relaxed and set to a relatively lower values, defined by the $mean(FN_{pr})+std(FN_{pr})$. Subsequently, the true positive rate (TPR) was increased by 19.4% at a cost of 4.54% decrease in the true negative rate (TNR). In addition, the F1 score and ACC were also improved by 2.19% and 4.27% for the PBRpredict-moderate than those of PBRpredict-strict model. In the PBRpredict-flexible model, the thresholds were even further lowered and set to $mean(FN_{pr})$. Therefore, all the false negative predictions (miss rate) of PBRpredict-strict with probability values greater than or equal to the $mean(FN_{pr})$ were corrected by the PBRpredict-flexible at a cost of high fall-out rate of around 16%.

In **Figure 4**, we illustrate the usefulness of these 3-different prediction using an example. PBD ID: 2CIA50 stores the structure of a sequence (chain A) with SH2 domain bound to a phosphopeptide. In Figure 9(a), we present the structure-specific sequence of chain A (length: 102) and the predicted annotation produced by PBRpredict-strict. The peptide-binding residues are marked in blue on the amino acid sequence. The true and false predictions are marked respectively in green and red on the predicted annotations (b for peptide-binding and n for non-binding). We observed that PBRpredict-strict could recognize most of the binding residues in the structure-specific sequence. However, the same model failed to recognize those residues when the input was the full-length sequence (UniProtKB: O43639, length: 380) containing the shorter structure-specific sequence (**Figure 4(b)**). On the other hand, the PBRpredict-moderate and flexible models could identify the binding residues on the full-length sequence, however with an increased num-

ber of false predictions of non-binding residues as binding residues. Therefore, the PBRpredict-Suite contains all the models that serve the purpose of recognizing peptide-binding residues under different scenarios.

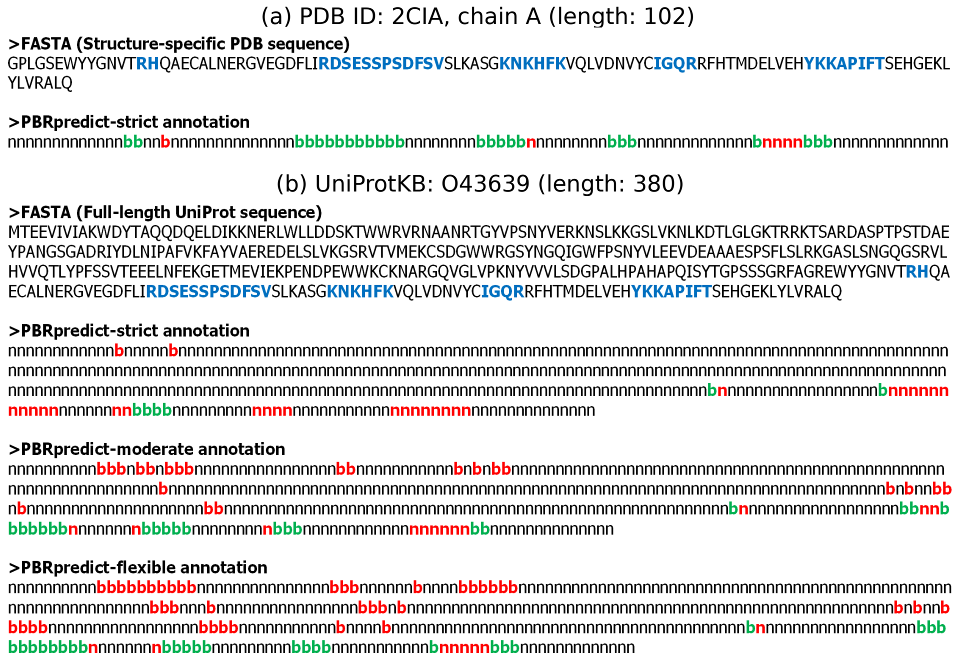


Figure 4: The outputs of PBRpredict-Suite models on (a) the structure-specific and (b) the full-length sequence of a same protein. Figure in (a) shows the protein sequence and predicted annotations given by PBRpredict-strict on PDB sequence (ID: 2CIA, chain A). Figure in (b) shows the protein sequence and predicted annotations given by all PBRpredict-Suite models on the full-length UniProt sequence (ID: O43696) of the same protein. The peptide-binding residues are marked in blue on the amino acid sequence. The true and false predictions are marked respectively in green and red on the predicted annotations ('b' for peptide-binding and 'n' for non-binding).

7 Performance of PBRpredict-Suite Modles on Structure-specific Sequences with Unknown Domain

In the main article, we statistically evaluated the performance of three different peptide-binding residue predictor models in PBRpredict-Suite separately

on two different datasets. In **Table 5** of **Section 4.3**, we reported the performance of the models on 169 protein chains, namely rcp_ts169 dataset, that share less than 40% sequence similarity with the chains in the training set. Moreover, in **Table 8** of **Section 4.5.2**, we outlined the performances of the PBRpredict-Suite models on 17 protein chains with 3 different peptide-recognition domains that are not known to the training set. In this section, we want to further discuss the performance of the models on the protein-peptide complexes that were deposited in the Protein Data Bank (PDB) after we accesed the PDB to collect the training dataset, specifically on September 2016.

Table 4: Comparison of PBRpredict-Suite models on 53 protein chains, collected in between June 2017 and March 2018.

Metric	PBRpredict-strict	PBRpredict-moderate	PBRpredict-flexible
Recall/TPR	0.453	0.666	**0.793**
Specificity/TNR	**0.992**	0.960	0.891
Fall-out rate/FPR	**0.008**	0.040	0.109
Miss rate/FNR	0.547	0.334	**0.207**
Accuracy/ACC	0.723	0.813	**0.842**
Precision	**0.895**	0.714	0.520
F1 Score	0.604	**0.645**	0.577
MCC	0.602	**0.689**	0.628

Best values are bold faced.

PBRpredict-strict thresholds: SVM(0.5), GBC(0.5), KNN(0.5), LogReg(0.5).

PBRpredict-moderate thresholds: SVM(0.3), GBC(0.34), KNN(0.35), LogReg(0.3).

PBRpredict-flexible thresholds: SVM(0.17), GBC(0.21), KNN(0.21), LogReg(0.2).

For this study, we collected 53 structures that were deposited in PDB in between June 2017 and March 2018 following the similar steps described in **Section 2.1**. The performnace of the proposed models while evaluated against the synthetic annotation (with smoothing) is reported in Table 4. The result shows that the PBRpredict-strict model with the threshold value of 0.5 resulted in the lowest false positive rate (FPR) with the highest precision score (correctness in predicting the positive class/peptide-binding residues). On the other hand, the PBRpredict-flexible model with the most relaxed threshold values resulted in the highest true positive rate (TPR), however, with the lowest specificity which quantifies a classifier's ability to predict the negative class/non-bining residues. However, the PBRpredict-

moderate provided the most balanced performance with the highest Mathews correlation coefficient (MCC) and F1 score (harmonic mean of the precisiona nd sensitivity).

For further visualization of the prediction quality and comparison with the existing predictor in the literature, SPRINT [Taherzadeh *et al.*, 2016], we mapped the prediction outputs on three sample structures from this dataset. **Figure 5** shows the actual peptide-binding residues (*red*) of human Mst1 kinase with SARAH domain, and the prediction outputs of PBRpredict-moderate and SPRINT are highlighted in *yellow* and *magenta*, respectively. The PBRpredict-moderate performed quite well with 90.2% accuracy and 80.5% MCC scores here, whereas SPRINT clearly suffered form overprediction.

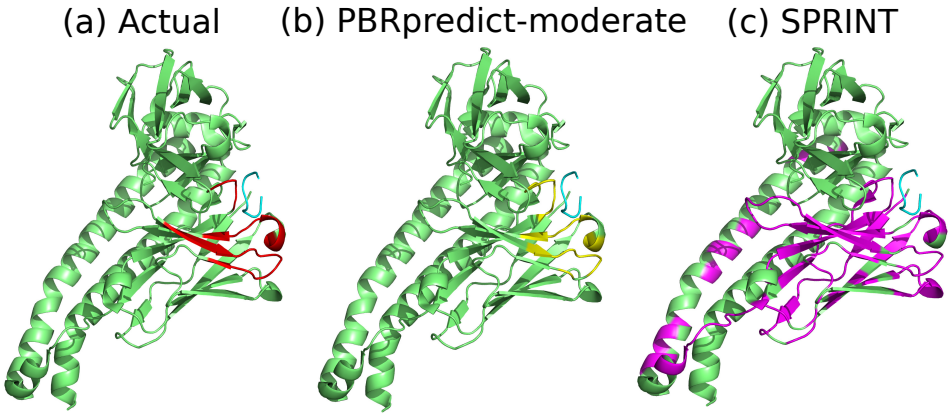


Figure 5: (a) Peptide-binding residues (*red*) of the SARAH domain (*green*), bound to a C8 peptide (*cyan*) in PDB 5XCR. The prediction outputs of PBRpredict-moderate (*yellow*) and SPRINT (*magenta*) are shown in (b) and (c), respectively.

Figure 6 shows the structure of a complex containing ROR gamma ligand-binding domain (LBD) bound with a repressor peptide. In **Figure 6 (a)**, the actual peptide-binding residues are shown in *red*, and the prediction outputs of PBRpredict-moderate and SPRINT are highlighted in *yellow* and *magenta* in **Figure 6(b)** and **(c)**, respectively. In this case, PBRpredict-moderate resulted in 84.9% accuracy and 75.3% MCC scores, whereas the comparative visualization shows that SPRINT overpredicted the residues as peptide-binding which are far apart from the peptide (*cyan*). We observed similar outputs from PBRpredict-moderate and SPRINT in case of predicting peptide-binding residues in Lysine-specific histone demethylase

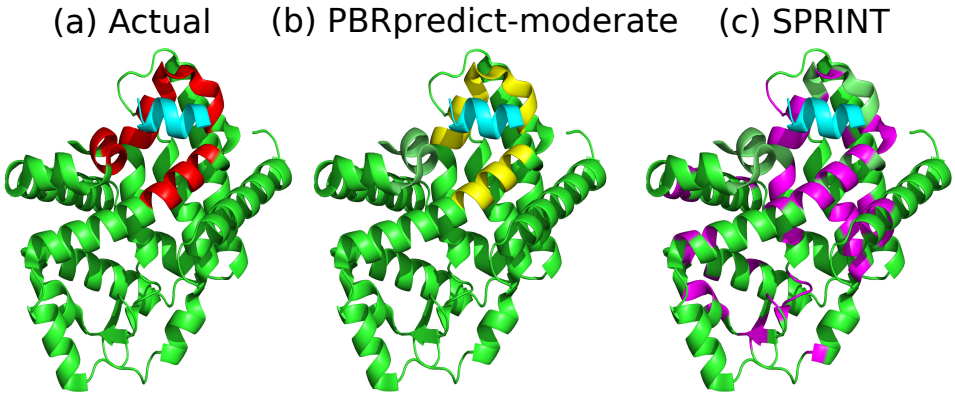


Figure 6: (a) Peptide-binding residues (*red*) of protein with ROR gamma ligand-binding domain (LBD) (*green*), bound to a repressor peptide (*cyan*) in PDB 5X8U. The prediction outputs of PBRpredict-moderate (*yellow*) and SPRINT (*magenta*) are shown in (b) and (c), respectively.

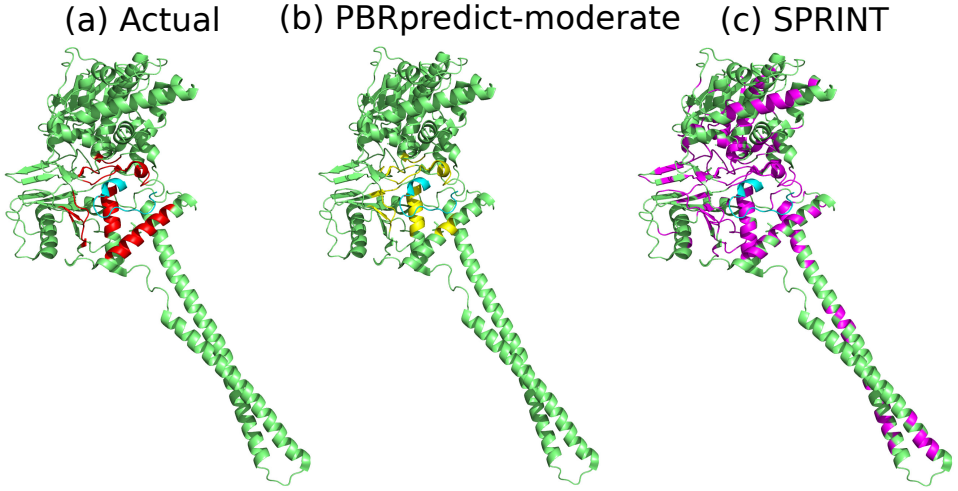


Figure 7: (a) Peptide-binding residues (*red*) of the Lysine-specific histone demethylase 1A (*green*), bound to a peptide 9 (*cyan*) in PDB 5X60. The prediction outputs of PBRpredict-moderate (*yellow*) and SPRINT (*magenta*) are shown in (b) and (c), respectively.

1A, shown in **Figure 7**.

The above results of the PBRpredict-Suite model suggest the strength of the proposed models, compared to the existing model, in locating poten-

tial peptide-binding sites within sequences for which the cognate domains are not known to the models. Therefore, the predictors can be useful in determining possible peptide-binding sites from protein sequence when no putative interaction information is known.

8 Case Study on Full-length Protein Sequences with Unknown Domain

In this section, we study the full-length protein sequences with PBRpredict-Suite models. Here, we want to evaluate the ability of the proposed models in identifying potential peptide-binding residues in proteins for which no experimental or template structure is available. For this study, we chose the Gid4 protein. Recently, [Chen *et al.*, 2017] discovered that the Gid4 subunit of the ubiquitin ligase GID in the yeast Saccharomyces cerevisiae targets the gluconeogenic enzymes, and recognizes the N-terminal proline (P) residue and the short 5-residue-long adjacent sequence motifs. The authors [Chen *et al.*, 2017] identified such interactions through in vitro experiments with two-hybrid assays.

We computationally predicted the potential residues in the Gid4 protein that may mediate such interactions with gluconeogenic enzymes to degrade them and down-regulate the gluconeogenesis. We collected 3 Swiss-Prot reviewed proteins from UniProt, GID4_YEAST (ID: P38263), GID4_HUMAN (ID: Q8IVV7) and GID4_MOUSE (ID: Q9CPY6), and ran the PBRpredict-Suite models on these sequences to identify possible peptide-binding residues. As PBRpredict-strict model produce conservative output on full-length proteins (**Figure 4**), here we show the predicted peptide-binding residues given by PBRpredict-moderate and flexible only. We report the results on GID4_YEAST in the main article (**Fig. 9**) and on GID4_HUMAN and GID4_MOUSE below.

GID4_HUMAN (UniProtKB – Q8IVV7): **Figure 8(a)** and **(b)** show the possible binding residues in blue identified by the PBRpredict-moderate and PBRpredict-flexible model in GID4_HUMAN. The moderate and flexible model found 8 and 39 binding-residue respectively with an average confidence of 0.58 and 0.55.

GID4_MOUSE (UniProtKB – Q9CPY6): The potential binding residues in GID4_MOUSE, predicted by the PBRpredict-moderate and PBRpredict-flexible model, are shown in **Figure 9(a)** and **(b)**. The moderate and flexible model found 19 and 67 binding-residues respectively with an average confidence of 0.56 and 0.55.

(a) PBRpredict-moderate annotation of GID4_HUMAN

MCARGQVGRGTQLRTGRPCSQVPGSRWRPERLLRRQRAGGRPSRPHPARARPGLSLPATLLGSRAAAAVPLPLPPALAPGDPAMPVRTECPPPAGASAASAASLI
PPPPINTQQPGVATSLLYSGSKFRGHQKSKGNSYDVEVVLQHVDTGNSYLCGYLKIKGLTEEYPTLTTFFEGEIISKKHPFLTRKWDADEDVDRKHWGKFLAFYQY
AKSFNSDDFDYEELKNGDYVFMRWKEQFLVPDHTIKDISGASFAGFYYICFQKSAASIEGYYYHRSSEWYQSLNLTHVPEHSAPIYEFR

(b) PBRpredict-flexible annotation of GID4_HUMAN

MCARGQVGRGTQLRTGRPCSQVPGSRWRPERLLRRQRAGGRPSRPHPARARPGLSLPATLLGSRAAAAVPLPLPPALAPGDPAMPVRTECPPPAGASAASAASLI
PPPPINTQQPGVATSLLYSGSKFRGHQKSKGNSYDVEVVLQHVDTGNSYLCGYLKIKGLTEEYPTLTTFFEGEIISKKHPFLTRKWDADEDVDRKHWGKFLAF
YQYAKSFNSDDFDYEELKNGDYVFMRWKEQFLVPDHTIKDISGASFAGFYYICFQKSAASIEGYYYHRSSEWYQSLNLTHVPEHSAPIYEFR

Figure 8: Figure (a) and (b) show the prediction outputs of PBRpredict-moderate and flexible models that are mapped on to the sequence of GID4_HUMAN. The predicted binding residues are marked in blue.

(a) PBRpredict-moderate annotation of GID4_MOUSE

MPVRTECPPPAGASTTSAASLIPPPPINTQQPGVATSLLYSGSKFRGHQKSKGNSYDVEVVLQHVDTGNSYLCGYLKIKGLTEEYPTLTTFFEGEIISKKHPFLTRK
WDADEDVDRKHWGKFLAFYQYAKSFNSDDFDYEELKNGDYVFMRWKEQFLVPDHTIKDISGASFAGFYYICFQKSAASIEGYYYHRSSEWYQSLNLTHVPEH
SAPIYEFR

(b) PBRpredict-flexible annotation of GID4_MOUSE

MPVRTECPPPAGASTTSAASLIPPPPINTQQPGVATSLLYSGSKFRGHQKSKGNSYDVEVVLQHVDTGNSYLCGYLKIKGLTEEYPTLTTFFEGEIISKKHPFLTR
KWDADEDVDRKHWGKFLAFYQYAKSFNSDDFDYEELKNGDYVFMRWKEQFLVPDHTIKDISGASFAGFYYICFQKSAASIEGYYYHRSSEWYQSLNLT
HVPEHSAPIYEFR

Figure 9: Figure (a) and (b) show the prediction outputs of PBRpredict-moderate and flexible models that are mapped on to the sequence of GID4_MOUSE. The predicted binding residues are marked in blue.

The above case-studies show that the PBRpredict-Suite can be a useful tool in revealing the amino acid compositions that mediates crucial interactions with peptide motifs from sequence alone when no structure is available. Such residue patterns can be further utilized for their cognate peptide identification. The above outcomes can further guide the experimental determination of the complex structure of these proteins by truncating the portion of the chain with potential peptide-binding sites.

References

[Altman, 1992] Altman,N.S. (1992) An introduction to kernel and nearest-neighbor nonparametric regression, *The American Statistician*, **46**, 175 – 185.

[Breiman *et al*, 1984] Breiman, L. *et al.* (1984) Classification and regression trees. *CRC press.*

[Breiman, 1996] Breiman,L. (1996) Bagging predictors, *Machine learning*, **24**, 123 – 140.

[Chang and Lin, 2011] Chang,C.-C. and Lin,C.-J. (2011) LIBSVM : a library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1 – 27:27.

[Chen *et al.*, 2017] Chen, S.-J. *et al.* (2017) An N-end rule pathway that recognizes proline and destroys gluconeogenic enzymes. *Science*, **355**, p.eaal3655.

[Cortes and Vapnik] Cortes,C. and Vapnik,V. (1995) Support-vector networks, *Machine learning*, **20**, 273 – 297.

[Frank *et al.*, 2004] Frank,E. *et al.* (2004) Data mining in bioinformatics using Weka, *Bioinformatics*, **20**, 2479 – 2481.

[Freedman, 2009] Freedman,D.A. (2009) Statistical models: theory and practice. *Cambridge university press.*

[Friedman, 2002] Friedman,J.H. (2002) Stochastic gradient boosting, *Computational Statistics & Data Analysis*, **38**, 367 – 378.

[Geurts *et al.*, 2006] Geurts,P. *et al.* (2006) Extremely randomized trees, *Machine learning*, **63**, 3 – 42.

[Ho, 1995] Ho,T.K. (1995) Random decision forests, *In Proceedings of the Third International Conference on Document Analysis and Recognition*, 278 – 282.

[Louppe *et al*, 2013] Louppe, G. *et al* (2013) Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 431 – 439.

[Nagi and Bhattacharyya, 2013] Nagi,S. and Bhattacharyya,D.K. (2013) Classification of microarray cancer data using ensemble approach, *Network Modeling Analysis in Health Informatics and Bioinformatics*, **2**, 159 – 173.

[Pedregosa *et al*, 2011] Pedregosa,F. *et al.* (2011) Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, **12**, 2825 – 2830.

[Robin *et al.*, 2011] Robin,X. *et al.* (2011) pROC: an open-source package for R and S+ to analyze and compare ROC curves, *BMC Bioinformatics*, **12**, 77.

[Sing *et al.*, 2009] Sing,T. *et al.* (2009) Visualizing the performance of scoring classifiers, *Package ROCR Version 1.0*, **4**.

[Taherzadeh *et al.*, 2016] Taherzadeh,G. *et al.* (2016) Sequencebased prediction of protein-peptide binding sites using support vector machine, *Journal of computational chemistry*, **37**, 1223 – 1229.

[Wolpert, 1992] Wolpert,D.H. (1992) Stacked generalization, *Neural networks*, **5**, 241 – 259.