DeepDNAbP: a deep learning-based hybrid approach to improve the identification of deoxyribonucleic acid-binding proteins

Md. Faruk Hosen[1], S. M. Hasan Mahmud[2], Kawsar Ahmed[1], Wenyu Chen[3], Mohammad Ali Moni[4], Hong-Wen Deng[6], Watshara Shoombuatong[5]*, Md Mehedi Hasan[6]*

[1]Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail, 1902, Bangladesh

[2]Department of Computer Science, American International University-Bangladesh (AIUB), Kuratoli, Dhaka 1229, Bangladesh

[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China

[4]School of Health and Rehabilitation Sciences, Faculty of Health and Behavioural Sciences, The University of Queensland, St Lucia, QLD 4072, Australia

[5]Center of Data Mining and Biomedical Informatics, Faculty of Medical Technology, Mahidol University, Bangkok, Thailand, 10700

[6]Tulane Center for Biomedical Informatics and Genomics, Division of Biomedical Informatics and Genomics, John W. Deming Department of Medicine, School of Medicine, Tulane University, New Orleans, LA, 70112 USA

*Corresponding author

watshara.sho@mahidol.ac.th
mhasan1@tulane.edu

Abstract:

Accurate identification of DNA-binding proteins (DBPs) is critical for both understanding protein function and drug design. DBPs also play essential roles in different kinds of biological activities such as DNA replication, repair, transcription, and splicing. As experimental identification of DBPs is time-consuming and sometimes biased toward prediction, constructing an effective DBP model represents an urgent need, and computational methods that can accurately predict potential DBPs based on sequence information are highly desirable. In this paper, a novel predictor called DeepDNAbP has been developed to accurately predict DBPs from sequences using a convolutional neural network (CNN) model. First, we perform three feature extraction methods, namely position-specific scoring matrix (PSSM), pseudo-amino acid composition (PseAAC) and tripeptide composition (TPC), to represent protein sequence patterns. Secondly, SHapley Additive exPlanations (SHAP) are employed to remove the redundant and irrelevant features for predicting DBPs. Finally, the best features are provided to the CNN classifier to construct the DeepDNAbP model for identifying DBPs. The final DeepDNAbP predictor achieves superior prediction performance in K-fold cross-validation tests and outperforms other existing predictors of DNA–protein binding methods. DeepDNAbP is poised to be a powerful computational resource for the prediction of DBPs. The web application and curated datasets in this study are freely available at: http://deepdbp.sblog360.blog/.

Keywords: DNA-binding protein, Feature encoding, SHapley Additive exPlanations and CNN

1. Introduction

DNA (Deoxyribonucleic Acid)-binding proteins are proteins that have specific binding affinity. They interact with DNA and are involved in various biological activities [1], [2]. For example, transcription factors (TFs) help with DNA transcription, nucleases cut DNA molecules, and histones are engaged in the packing of chromatin into the nucleus [3]. DNA-binding proteins (DBPs) are vital components of antibiotics, steroids, and other drugs treating cancer and genetic illnesses. DBPs are involved in regulating gene expression and the process of DNA synthesis. The most crucial intercellular and intracellular function of DBPs are DNA replication, repair, transcription, modification, recombination, and other biological activities associated with DNA [4]. Traditionally, researchers have identified DBPs through wet-lab experimental techniques such as chromatin immunoprecipitation on microarray (ChIP-chip), genetic analysis, and X-ray crystallography. However, wet-lab experimental techniques are costly and time-consuming. Hence, it is highly desirable to develop accurate and cost-effective prediction methods for accurately and rapidly identifying DBPs.

Over the last few decades, many computational methods have been developed for predicting DBPs [5], [6]. Most of these methods define DBP identification as a binary classification problem: Their goal is to predict whether the inputted protein is a DBP or not. These methods can largely be categorized into two groups: structure-based methods and sequence-based methods. Structure-based methods rely on structural information of the proteins including spatial distribution, net charge, electrostatic potential, and dipole and quadrupole moment tensors [7], [8]. Although structure-based methods show great predictive performance, their application is limited, since the structural information of proteins is not always adequate [9]. Some structure-based methods, including DBD-Hunter [6] and iDBPs [8], use both structural and sequential information of proteins. Sequence-based methods can overcome the limitations of the structural approach since sequence features are normally easier to extract and acceptable to use. There are three types of sequence-based features: (1) composition-based features, such as amino acid composition (AAC) [10], dipeptide composition (DPC) [11], and pseudo AAC (PseAAC) [12]–[14]; (2) autocorrelation-based features like auto crosscovariance [15], [16], normalized Moreau-Broto Autocorrelation [9], and physicochemical distance transformation [17]; and (3) profile-based features, including position-specific score matrix (PSSM) [18]–[20] and hidden Markov model (HMM) [21]. Among these, profile-based features generally perform best, so the development of sequence-based DBP predictors has become an urgent demand in bioinformatics research.

Previous studies have highlighted the significance of PSSM-based features for boosting DBP prediction. For example, Kumar et al. successfully obtained evolutionary information attached to the PSSM profile to identify DBPs [18]. Waris et al. developed an innovative method by combining the PSSM profile with dipeptide composition and split AAC [19]. In recent years, scientists have developed a series of sequence-based methods to identify DBPs, including iDNA-Prot [22], PseDNA-Pro [17], iDNAPro-PseAAC [20], iDNA-Prot|dis [23], Local-DPP [24], HMMBinder [21], IKP-DBPPred [5], iDNAProt-ES [25] and DPP-PseAAC [13]. These methods use only sequence information and identify DBPs using machine-learning algorithms, such as support vector machine (SVM) [25] or random forest (RF) [22], [24]. In iDNA-Prot [22], the RF algorithm is adopted to train the DBP identification model with Chou’s pseudo-amino-acid composition (PseAAC) [26], [27] extracted by a grey system theory [28]. In PseDNA-Pro [17], three kinds of sequence-based information (amino acid composition, physicochemical distance transformation, and PseAAC) extracted from protein sequences are used to generate the feature vector of protein data, then the SVM algorithm is employed to generate the identification model. In HMMBinder [21], the HMM profiles are first generated by HHblits [29], then the monogram and bigram features are generated from HMM profiles, merged with each other as the final feature vector, and finally, the SVM algorithm is used to generate a DBP identification model. In DPP-PseAAC [13], Chou’s general PseAAC [26], [27] is used to generate a protein sequence, the RF and SVM-RFE (support vector machine recursive feature elimination) algorithms are used to rank features, and finally, the algorithm of SVM is employed to obtain the final prediction model. Despite these methods provided by researchers, there remains room for further improvement in accurately identifying DBPs from protein sequences.

In this paper, we introduce a sophisticated method for identifying DBPs to further improve DBP prediction. Firstly, 1,052 DBPs and 1,052 non-DBPs were collected from the Protein Data Bank (PDB) [30] and used to train and test the prediction model. Secondly, we generated a feature representation of each protein using the following steps: (1) three sequence-based single-view features (TPC, PseAAC, and PSSM composition) were extracted to represent different sequence information; (2) single and combined features [31] were generated to learn the classifiers; and (3) a suitable feature selection algorithm, SHAP, was employed to select an outstanding subset of the super feature to further quantify the difference between DBPs and non-DBPs. Finally, the prediction model was learned using CNN on the selected feature subset for identifying DBPs and named *DeepDNAbP*. Experimental results demonstrated that DeepDNAbP outperforms existing methods and achieves the best performance (e.g., the highest accuracy 89.68%) for identifying DBPs, indicating that our proposed methods are potentially useful for practically identifying DBPs and is freely available on the internet. Taken together, the present study provides a useful tool for predicting DBPs as well as valuable insights into the important sequencing patterns surrounding DBPs.

2. Materials and methods

The construction process of the DeepDNAbP is shown in Figure 1. It consists of multiple steps, including data preparation, feature extraction and selection, best classifier selection, and final prediction. Here, three different feature extraction methods were employed (PSSM, PseAAC and TPC) and then the optimal subset of features was obtained using three feature selection techniques. The optimum features from each extraction technique were fed to four classifiers to build the prediction models using a 5-fold Cross Validation (CV). Finally, the classifier that achieved the best prediction performance was selected to construct the final predictor in this study. Reasons to consider feature extraction techniques in predicting DNA binding proteins include (1) the PSSM feature extraction approach has the advantage of integrating the sequence evolutionary information from the profile generated by the PSI-BLAST search, and (2) PseAAC and TPC are sequence information-based feature extraction approaches that represent amino acid (AA) order information of protein sequences.

2.1 Data preparation

The problem of identifying DBPs is defined as a binary classification problem. To develop an effective prediction model, it is important to construct a valid training dataset. The curated dataset S was presented as:

$$S = S_{posi} \cup S_{nega}$$
(1)

Where S_{posi} is the set of positive samples that only contains DBPs and S_{nega} is the set of negative samples that only includes non-DBPs. The symbol ∪ represents the "union" of these two sets.

The preserved dataset was collected and primarily used by Jun Hu et al. [32], who selected the DBP chain and non-DBP chain from the PDB database [30]. The dataset contains 2,104 protein sequences with 1,052 positive samples (DNA-binding protein sequences) and 1,052 negative samples (non-DNA-binding protein sequences). We utilized a random sampling technique to produce negative samples the same size as the DBP positive samples to overcome the issue of data imbalance. The above sequence chains were reported [30], which had $\geq 25\%$ sequence identity [33], and chains less than 50 residues long were eliminated. We additionally eliminated the proteins containing unknown residue 'X'. To further test the robustness of our model, we used an additional independent test dataset [32] that contained 148 DNA-binding proteins and 148 non-DNA-binding proteins. All curated sequences in this independent dataset are no smaller than 50 residues long and do not contain any character 'X' sequences. The datasets employed in this study can be downloaded at: http://deepdbp.sblog360.blog/. Users can collect DNA-binding protein sequences from the PDB database (https://www.rcsb.org/) and search for a protein sequence in the PDB databases using their specific ID in FASTA format.

2.2 Feature extraction

2.2.1 Position specific scoring matrix (PSSM)

PSSM is an evolutionary information scheme that is generally used for motif (pattern) representation in biological chains [34]. PSSM is used to identify hidden and evolutionary information of homologous protein sequences [35]–[37]. In PSSM, all amino acid residue is counted against 20 values. It counts the frequency of substitution in which a particular position in protein families is identified. A negative score shows that the desired amino acids are less frequent in the arrangement. In contrast, a positive score indicates that the substitutions appear more frequently. Suppose a protein sequence P having length L residues can be represented as:

$$P_{PSSM} = \begin{bmatrix} E_{1\to1} & E_{1\to2} & \cdots & E_{1\to j} & \cdots & E_{1\to20} \\ E_{2\to1} & E_{2\to2} & \cdots & E_{2\to j} & \cdots & E_{2\to20} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ E_{i\to1} & E_{i\to2} & \cdots & E_{i\to j} & \cdots & E_{i\to20} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ E_{L\to1} & E_{L\to2} & \cdots & E_{L\to j} & \cdots & E_{L\to20} \end{bmatrix}$$
(2)

Where $E_{i\to j}$ indicates the i-th position residue score within the protein arrangement, which is replaced by the amino acid type j within the evolution computing processes. The sequential order of 20 amino acids is addressed by j = 1….20. The value of P_{PSSM} is achieved through PSI-BLAST [38], [39], which investigated the SWISS-PROT database through three iterations having 0.001 as the cut-off E-value against sequence protein P within multiple sequence alignment. Consequently, L x 20 scoring matrix is obtained. Then, the P_{PSSM} is normalized using utilizing standard deviation. SWISS-PROT is a protein sequence database that contains annotations such as protein function, post-translational modifications, the structure of protein domains, variants, and other features. It also provides highly integrated information from other protein-related databases.

Because the lengths of individual protein sequences vary in the PSSM matrix, it is difficult to make a predictor. We used the bigram probabilities descriptor, initially introduced by Sharma et al. [40] for the purpose of protein fold detection, to convert the PSSM matrix into sequences of the same length. The bigram probabilities descriptor is capable of properly handling M_{PSSM} matrices of various lengths and creating feature vectors of a fixed size. The relative probabilities of finding the j-th amino acid at i-th region in a specified protein sequence is determined with $\sum_{j=1}^{20} P_{i,j} = 1$ where i = 1,2, . . . , L. Therefore, the bigram frequency of occurrence is computed using the following formula:

$$BG_{m,n} = \sum_{i=1}^{L-1} P_{i,m} P_{i+1,n}, where\ (1 \leq \mathrm{m} \leq 20, 1 \leq \mathrm{n} \leq 20) \qquad (3)$$

Equation (1) provides frequencies of occurrences $BG_{m,n}$ in response to 400 bigram transitions where m = 1, . . . , 20; n = 1, . . . , 20. The matrix BG is a bigram matrix with 400 distinct elements described by the bigram feature vector F.

$$F = \{BG_{1,1}, BG_{1,2}, \ldots, BG_{1,20}, \ldots, BG_{2,1}, \ldots, BG_{2,20}, \ldots, BG_{20,1}, \ldots, BG_{20,20}\}^T$$
(4)

Where T denotes the transposition of the vector. The main benefit of the PSSM-bigram approach is that it does not use zeros in the bigram feature vector, making the prediction more accurate.

2.2.2 Tripeptide Compositions

In TPC, three adjacent native amino acids establish an effective and minimal biological recognition signal. It can form a valuable paradigm for finding peptides and small biological molecule copies that are suitable modulators of organic function [41]. Anishetty et al. [42]

have shown that tripeptides can be used for predicting plausible structures for oligopeptides and de novo protein design. Generally, TPC has been applied for identification of mycobacterial membrane proteins [43], submitochondrial location prediction [44], identification of voltage-gated potassium channels [45], and predicting subcellular localization of mycobacterial proteins [46]. The features obtained using TPC are 8000-Dimension vectors. A protein sequence can be defined as:

$$P = [f_1, f_2, \cdots, f_i, \cdots, f_{8000}]^T$$
(5)

Where T denotes the transposition of the vector and f_i represents the frequency of the i-th $(i = 1,2,3,\cdots,8000)$ tripeptide and can be expressed as:

$$f_i = n_i/(L-2)$$
(6)

Where n_i and L represent the frequency of the i-th tripeptide and the length of the protein chain, respectively.

2.2.3 Pseudo Amino Acid Composition (PseAAC)

The sequence of an amino acid can be represented by a set of discrete numbers mapping the arrangements of its physicochemical properties into a fixed number of features. The traditional amino acid composition method has been broadly used to predict the structural class of proteins [47], [48] and records the frequency of amino acids in the order of only one protein. To study the order information of protein chain, Chou [49] proposed a method called PseAAC to extract the features. It can represent both compositional and sequential order information. This method is widely utilized for protein function prediction [49]–[51]. The following equation can express the features of PseAAC:

$$X = [x_1, x_2, \cdots, x_{19}, x_{20}, x_{20+1}, \cdots, x_{20+\lambda}]^T \quad (\lambda < L)$$
(7)

Where L represents the length of the given protein sequence and each of the elements is displayed as follows:

$$X_\delta = \begin{cases} \frac{F_\delta}{\sum_{\delta=1}^{20} F_\delta + W\sum_{k=1}^{\lambda}\psi_k}, & 1 \le \delta \le 20 \\ \frac{W\psi_\delta}{\sum_{\delta=1}^{20} F_\delta + W\sum_{k=1}^{\lambda}\psi_k}, & 20+1 \le \delta \le 20+\lambda \end{cases} \quad (8)$$

Where X denotes a feature vector, F_δ indicates the frequency at δ-th amino acid (AA) in the protein sequence, and W represents the weight factor with a value of 0.05. From equation (8), we can see that the first 20 components represent the frequency of occurrence in the protein chain, and λ is the sequential element that expresses different steps in AA sequence information. It is achieved through the physicochemical properties of AA. In this work, the range of the parameter δ is 0–50. Here ψ_k represents *j*-tier correlation factor for the protein. Due to the model accuracy of prediction results, the ideal δ parameters can be resolved from various parameter settings. Finally, an 80-D feature vector is generated from each sequence.

2.3 SHAP (SHapley Additive exPlanations)

SHAP (Shapley Additive exPlanations), introduced by Lundberg and Lee [52], is a model additive explanation method where each prediction is interpreted by the contribution of the features to the model's output. This SHAP feature selection approach calculates Shapley values from coalitional game theory. These SHAP scores encode the value of a feature for a model in order to utilize the contribution information of each feature and then order the features based on their importance. In SHAP, it replaces each feature X_i with the binary variable Z_i which informs whether the feature value is present or not. SHAP specifies this explanation as:

$$g(z') = \Phi_0 + \sum_{i=1}^{M} \Phi_i z_i' \qquad (9)$$

Where g represents the explanation model, $z' \in \{0,1\}^M$ indicates the coalition vector, M is the number of input features included in the model and $\Phi_i \in R$. More specifically, if the coalition vector z' equals to 1, the corresponding feature is observed, and 0 means the corresponding feature is absent. Here the symbol Φ_i is the feature attribution values for a feature i which tells how much the presence of feature i will contribute to the final output. Based on the concept of game theory, SHAP values can be calculated using the following equation:

$$\phi_i = \sum_{S \subseteq M \setminus \{i\}} \frac{|S|!(M-|S|-1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)] \qquad (10)$$

Where M represents the set of all features included in the model, S indicates all feature subsets achieved from M after excluding feature i, and the function $f_x(S)$ computes the total contribution of a given set of features S. In SHAP, the contribution of each feature is estimated by assessing the difference between the prediction when the value of corresponding feature i is known, versus when corresponding feature value i is unknown for all subsets $S \subseteq M \setminus \{i\}$.

2.4 Convolutional Neural Network (CNN)

CNN, a class of artificial neural networks (ANN) in deep learning, is commonly applied for natural language processing, recommender systems, and image/video recognition [53]. Typically, CNNs consist of three layers: convolutional layers (CLs), pooling layers (PLs), and fully connected layers (FCLs) [53]. A convolutional layer performs feature extraction with a rectified linear unit (ReLU) activation function [54]. Next, a max-pooling layer is applied to reduce the size of the features. Finally, the fully connected layer and the output sigmoid layer are utilized to classify the tasks.

As a core component of CNNs, CLs can help the model to learn global and local structures from input vectors [54]. We use two CLs in our model; multiple CLs are stacked along the depth of the network, allowing the network to extract high-level features.

This technique showed significant improvement in terms of computational complexity as well as program runtime after adding one more CL and max-pooling layer. Each convolutional layer output can be calculated using the following formula:

$$y_k^l = f\left(\sum_m W_{m,k}^l y_m^{l-1} + b_k^l\right), \tag{11}$$

Where l denotes the layer index, and m and k represent the index of the input and output feature maps, respectively. Input y_k^l indicates the k feature map of the l layer and output y_m^{l-1} indicates the m-th feature map of the $l-1$ layer. W and b are the convolutional weight tensor and the bias term, respectively. The output level of our model is basically a logistic regression algorithm, where y_k^l is the input and is computed as follows:

$$\breve{y} = f(W^l y^l + b^l) \tag{12}$$

Where output $\breve{y}$ indicates the final predicted score. b and W are the bias vector and weight matrix, respectively. Since DBP identification is a binary classification problem, its output value is 2, addressed by a positive and negative class. To discover the appropriate parameters, we want to reduce cross-entropy loss by determining the adaptive moment [55] and back-propagation strategy:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \breve{y}_l + (1-y_i)\log(1-\breve{y}_l)] \tag{13}$$

To further improve the efficiency of our model, batch normalization [56] and dropout [57] tricks were used. During the training process, the dropout trick drops a little unit in FCLs, while batch normalization primarily helps to normalize the input layers to zero mean and unit standard deviation. Indeed, dropout was able to manage the overfitting problem, and batch normalization allowed us to use high learning rates.

2.5. Evaluating matrices

In classification algorithms, success rates can be evaluated with the help of several performance measures. In this work, we have used the following eight performance evaluation parameters to assess the prediction of our method.

$$Accuracy(Acc) = \frac{TP + TN}{TN + TP + FN + FP} \tag{14}$$

$$Sensitivity\ (Sen) = \frac{TP}{TP + FN} \tag{15}$$

$$Specificity\ (Spe) = \frac{TN}{FP + TN} \tag{16}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TN + FP)(TN + FN)(TP + FN)}} \tag{17}$$

$$Precision\ (Pre) = \frac{TP}{TP + FP} \tag{18}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

Where TP, FP, TN and FN indicate the number of correctly classified positive samples, the number of negative samples classified as positive samples, the number of correctly classified negative samples, and the number of positive samples classified as negative samples, respectively.

The Receiver Operating Characteristic (ROC) curve (represented as AUC) is also computed because it is a powerful evaluation metric for assessing the performance of a binary predictor. The ROC-curve is a graphical plot of the true positive rate (TPR) against the false positive rate (FPR) under different threshold settings. TPR is sensitivity, whereas FPR is 1-specificity.

2.6. Experimental Setup and Packages

In this study, all experiments were completed using Python version 3.7.7 or above on three separate machines with the following configurations:

- A desktop computer with Intel Core i5 CPU @ 2.71GHz x 4, Windows 10, 64-bit OS and 8 GB RAM.
- A desktop computer with Intel Core i5 CPU @ 2.11GHz x 4, Windows 10, 64-bit OS and 8 GB RAM.
- A server machine with Intel Core i5-3320M CPU @ 2.60GHz x 4, Ubuntu 18.04.2 LTS, 64-bit OS, 13 MB L3 cache and 64 GB RAM.

CNNs and SHAP were used for model learning and feature selection. These are accessible from the Python packages TensorFlow 2.0 [58] and SHAP 0.39.0. In the CNN architecture, we used optimized parameter settings. Specifically, we used batch size 16, kernel size 4, 2 hidden layers, and a dropout rate of 0.5. In addition to installed packages in Scikit-learn [59], we used Matplotlib [60], Seaborn [61] and Plotly packages [62] to plot our graphs.

3. Results and Discussion

We evaluated the performance of three feature extraction techniques using four ML classifiers, namely, Support Vector Machines (SVM) [63], K-Nearest Neighbor (KNN) [64], XGBoost [65], Logistic Regression Ensembles (LRE) [66], and CNN [54], based on 5-fold CV test. We observed evolutionary-based PSSM features that attained the highest performance with ACC in the range of 76.56%–85.78% regardless of different classifiers. We also noted that two classifiers, XGBoost and CNN, achieved similar performance on three encodings (PSSM, PseAAC, and TPC). In the case of PSSM, CNN achieved a superior performance, better than XGBoost classifiers and significantly better than LRE, SVM and KNN. After applying the feature selection SHAP technique, the PSSM features achieved 89.68% and 0.70 for AUC and MCC, respectively, 2.54% and 0.06% higher than without feature selection. Overall, the comparative analysis indicates that evolutionary-based features encodings reflect significant evidence around DBPs; hence, we considered this extraction method for further analysis.

3.1 Performance comparison of four single-view features

Initially, we explored the discriminative performance of the three single-view features: PSSM, PseAAC and TPC. We then applied the CNN classifier to assess the performance of the three sequence representation models used in this work. Table 1 illustrates the comparison of the discriminative performance of these single-view features without applying the feature selection technique. From Table 1, we observe that the PSSM outperforms the other two single-view features on most of the evaluation indices. Only PseAAC shows higher specificity and precision values than PSSM and DPC. Moreover, the accuracy of PSSM is 81.25% on the 2104 dataset, which is 7.81% and 9.37% higher than PseAAC and TPC models, respectively. The MCC and F1 score of PSSM are 0.64 and 81.82%, which are 0.17 and 5.76% higher than PseAAC and 0.20 and 10.85% higher than TPC, respectively. The best sensitivity is 90.00%, which is also achieved by the PSSM model.

A bar graph of PSSM, PseAAC and TPC is shown in Figure 2, which confirms the distinct and high quality of the target features. In Figure 2, it is evident that the AUC value of PSSM is 87.14%, which is larger than the other single-view features. The fundamental reason PSSM outperforms the other two single-view features is that the PSSM incorporates direct relevant information for identifying DNA-binding proteins. PSSM is an effective evolutionary information-based feature extraction technique that can generate a highly discriminatory nature for pattern (motif) representation in the protein sequence. Moreover, in most cases, we can see that the highest values were achieved with the PSSM-Bigram. These experiments confirm that the PSSM descriptor is more informative, depicting the evolutionary-based features' efficiency and playing a more significant role in predicting DNA binding protein than the other descriptors.

3.2 Performance comparison on hybrid feature space

To yield the best performance, a hybrid feature space was created by adding different combinations of the single feature extraction methods. This hybrid feature space is constructed by combining PSSM, PseAAC and TPC feature spaces. We manually combined single-view features to produce a hybrid feature space. To assess the performance of the hybrid feature space, we applied the CNN architecture on the experimental datasets. Table 2 shows the compared results of these hybrid features. Corresponding bar graphs are shown in Figure 3. Among the four hybrid feature datasets listed in Table 2, PSSM+PseAAC achieved the highest AUC score (83.78%) and PseAAC+TPC had the lowest (77.90%). Upon closer examination of the predictive results, we find that the MCC score of the individual feature TPC is slightly higher than the hybrid feature space (PseAAC+TPC). In addition, the prediction performance of the highest hybrid feature space (PSSM+PseAAC) is considerably lower than the single-view feature space (PSSM) regarding the evaluation indices. However, some hybrid feature spaces exhibit performances similar to the single-view feature space in a few cases. We can conclude that the hybrid feature space has high dimensionality and long execution time compared to the single-view feature space. Thus, the computational cost of the single-view feature is less than that of the hybrid feature space. In this regard, it is more effective to consider single-view feature space rather than hybrid feature space.

3.3 Performance comparison of different classifiers

We performed a series of comparative analyses using three individual feature descriptors (PSSM, PseAAC and TPC) to investigate the impacts of various feature spaces applying different classification algorithms. Each specific feature descriptor was assessed using four classification algorithms: SVM, KNN, XGBoost, LRE and CNN. The performance of these three feature extraction techniques for different classification algorithms is shown in Table 3. Using the KNN classifier, the PSSM feature space achieved AUC values of 81.89%, PseAAC achieved AUC of 74.93%, and TPC achieved AUC of 75.45%. Likewise, the AUC values of PSSM, PseAAC and TPC-based features were 87.70%, 83.48% and 77.82%, respectively, for the SVM classifier. For the XGBoost classifier, PSSM, PseAAC and TPC feature spaces obtained AUC values of 88.32%, 82.62% and 81.39%, respectively. Using the LRE classifier, the PSSM feature space achieved AUC values of 80.75%, PseAAC achieved AUC of 80.92%, and TPC achieved AUC of 74.26%. Similarly, the AUC values of PSSM, PseAAC and TPC using CNN classifier were 89.68%, 83.48% and 82.19%, respectively. Performance metrics including accuracy (Acc), sensitivity (Sen), specificity (Spe), MCC, and F1-measure for PSSM, PseAAC and TPC feature spaces are shown in Table 3.

For the CNN classifier, the prediction accuracies are 85.78%, 78.13% and 74.88% for PSSM, PseAAC and TPC feature spaces, respectively. The produced sensitivity is 90.63% for PSSM, 74.36% for PseAAC and 75.00% for TPC. Likewise, specificity values of 78.13% for PSSM, 84.00% for PseAAC, 74.79% for TPC were achieved. In addition, the MCC and F1-measure values are respectively 0.70 and 85.30% for PSSM feature space, 0.58 and 80.56% for PseAAC feature space, and 0.50 and 72.26% for TPC feature space. In this research, we used 1D convolution layers. A 1D convolution is used for time series, vector data, and NLP in one direction, right to left (vector/feature values), while a 2D kernel moves in two directions, height x width, which is mostly used in image classification. We have provided detailed implementation information about 1D convolution for three feature extraction methods (PSSM, PseAAC and TPC) in Table 4.

In the case of KNN, SVM, LRE and XGBoost classifiers for all types of feature spaces, the accuracy (Acc), sensitivity (Sen), specificity (Spe), Precision (Pre), MCC, and F1-measure scores are also shown in Table 3. From the above discussion, we see that among the experimental datasets, PSSM feature space achieved better predictive performance compared to PseAAC and TPC. We can observe from Table 3 that all three feature spaces obtained promising results for the CNN classifier, followed by XGBoost, LRE, SVM and KNN classifier. However, TPC feature space yielded more unrhymed results than the other two feature extraction techniques, while PseAAC obtained slightly better results than TPC. In addition, CNN consistently achieved better prediction performance compared to XGBoost, LRE, SVM, and KNN for all three feature spaces. Bar graphs for the three-feature group for different classifiers are illustrated in Figure 4 (A)–(C). We found that the bar graphs generated by CNN are significantly higher than the other classifiers.

3.4 Performance comparison using SHAP feature selection

Feature selection is a fundamental strategy for selecting the best set of features in the field of pattern recognition and biological data processing [67]–[69]. It is a combinational optimization technique that can extend the prediction capacity of a model. Several feature selection methods have been broadly employed with DNA binding protein datasets in recent works. Typically, the feature selection method evaluates feature subsets of DBP using a classification algorithm. It gives individual assessment indicators as well as fitness levels based on the accuracy provided for the effective elimination of redundant data on DNA binding protein features and for the extraction of data for each specific protein. The experimental results of Principal Component Analysis (PCA) [70], Univariate [71], SHAP [52], Sparse Group Lasso (SGL) [72] and Regularized Random Forests (RRF) [73] feature selection techniques on the DNA binding protein dataset with different feature dimensions are listed in Table 4. The SHAP technique obtained a significantly higher prediction performance than PCA and Univariate on DNA binding protein features. To compare the performance of the proposed model, we also evaluated the CNN classifier without feature selection technique, as shown in Table 1. We can compare Table 1 and Table 4 and see that the model performance improved and obtained the best results when the feature selection SHAP technique (see for 200 features) is applied on the dataset.

From Table 4, we found the optimal prediction effect with our dataset when taking full DNA binding protein features. Thus, it is better to eliminate a few features from our experimental datasets. However, for the PCA technique, the AUC, Acc, Sen, Spe, MCC and F1 scores for 200, 300 and 350 features were slightly better than 150 features. The best results were obtained for 300 features: Acc, Sen, Spe, MCC, Pre and F1 scores for 300 features were 82.72%, 90.00%, 78.26%, 0.68, 78.26% and 83.72%, respectively. In contrast, the Acc, Sen, Spe, MCC, Pre and F1 scores for 150 features were 74.42%, 77.27%, 71.43%, 0.49, 73.91% and 75.56%, which are 8.3%, 12.73%, 6.83%, 0.19, 4.35% and 8.16% lower than 300 features, respectively. For the Univariate technique, the Acc, Sen, Spe, MCC, Pre and F1 scores for 200 features were 84.38%, 90.63%, 78.13%, 0.69, 80.56% and 85.29%, respectively. These results are higher than the other feature dimensions. Similarly, for the SHAP technique, the Acc, Sen, Spe, MCC, Pre and F1 scores for 200 features reach 85.78%, 90.63%, 78.13%, 0.70, 80.56% and 85.30%, respectively, higher than the other feature dimensions. The Acc, Sen, Spe, MCC, Pre and F1 scores for 150 features utilizing SHAP are 81.25%, 87.50%, 75.00%, 0.63, 77.78% and 82.35%, which are 4.53%, 3.13%, 3.13%, 0.07, 2.78% and 2.95% lower than using 200 features, respectively. In the case of the SGL technique, the AUC, Acc, Sen, Spe, MCC and F1 scores for 150, 200, 300 and 350 features were slightly lower than for 250 features. The best results were obtained for 250 features: Acc, Sen, Spe, MCC, Pre, and F1 scores for 250 features were 78.20%, 78.18%, 78.22%, 0.56%, 79.63%, and 78.90%, respectively. In contrast, the Acc, Sen, Spe, MCC, Pre and F1 scores for 150 features were 73.93%, 73.87%, 74.00%, 0.48%, 75.93%, and 74.89%, which are 4.27%, 4.31%, 4.22%, 0.06%, 3.70% and 4.01% lower than using 250 (lower feature dimensions) features, respectively. For the RRF, the Acc, Sen, Spe, MCC, Pre, and F1 scores for 250 features were 78.01%, 78.78%, 78.01%, 0.55%, 78.87% and 78.12%, respectively. These results for 250 features are higher than the other feature dimensions of RRF.

Moreover, the AUC score of 200 features for PCA was 87.83%, which is 5.00% higher than 250 features, 2.4% higher than 300 features, 4.3% higher than 350 features, and 6.53% higher than 150 features. For the Univariate technique, the AUC of 200 features was 89.36%, which is 1.68%, 4.59%, 4.25%, and 4.03% higher than 250, 300, 350, and 150 features, respectively. For the SHAP technique, the AUC of 200 features was 89.68%, which is 0.39%, 3.87%, 0.7%, and 3.77% higher than 250, 300, 350 and 150 features, respectively. For the SGL technique, the AUC of 250 features was 84.87%, which is 1.45%, 2.52%, 3.47%, and

3.18% higher than 200, 300, 350, and 150 features, respectively. Moreover, the AUC score of 250 features for RRF was 84.10%, which is 3.4% higher than 150 features, 1.2% higher than 200 features, 2.1% higher than 300 features, and 3.08% higher than 350 features. Clearly, 200, 250, and 300 features displayed great significance in most cases, and prediction results decreased from 150 features for three feature selection techniques. We did not see improved results on feature selection algorithms for DNA binding protein datasets, and the SHAP is an appropriate algorithm here. To make a clear comparison of prediction effects, the results SHAP importance-bar graph of the PSSM dataset for different feature dimensions is shown in Figures 5AB and Supplementary Figure S1. The SHAP bar plot shows the important features in the form of rectangular horizontal bars, where the lengths of the bars are equivalent to the importance of that feature. (A feature is considered important when its Shapley value is high.) As we needed global significance, we summed the contribution of each feature, or absolute Shapley values, then plotted each of the features by sorting them in decreasing order. Figure 5(A)–(B) shows the important features based on SHAP contributions for the XGBoost trained before predicting DBPs.

The prediction performances show that SHAP is the most significant of the three feature selection methods for most feature dimensions. The SHAP summary plot for the different feature dimensions on the SHAP method are outlined in Figures 6AB and Supplementary Figure S2). The SHAP summary plot gives a high-level composite view that displays the importance of features with feature effects. Each point in the plot represents a SHAP value for a specific feature of an instance. The values that pull the prediction power of the model downwards are on the left, and the values that push the prediction further up are on the right. On the y-axis, the features are placed in descending order, and the position of the x-axis is determined by the Shapley value. The colors separate the relative size of the features between instances: Low values are colored blue and high values are colored red. Overlapping data points in the y-axis direction show the distribution of SHAP values for each individual feature. Moreover, in the summary plot we clearly observe the relationship between the value of a feature and the effect on prediction.

3.5 CNN Hyperparameter Adjustment

CNN consistently achieved better prediction performance because we performed hyperparameter tuning in order to obtain the expected results. Hyperparameters affect classification performance as well as learning time. Hyperparameters of a CNN model set parameters such as kernel size, number of kernels, hidden layers, activation functions, learning rate, batch size, and dropout percentage.

We used two convolutional layers as the hidden layers. In the first convolutional layer, we used 48 filters and the kernel size was set to 4. In the second convolutional layer, we used 64 filters and kernel size was set to 4. In the fully connected layer, we used 64 filters. We used ReLU as the activation function in the hidden layer. Before the fully connected layer, we added a dropout layer, and the dropout rate was 0.5. We ran 50 epochs and set up a batch size equal to 32. We utilized the Adam algorithm and set a learning rate of 0.00001 for optimizing the binary cross-entropy loss. Hence, the hyperparameter setting was sampled as shown in

Table 5. Detailed parameter settings of the other three classifiers for different feature extraction are listed in Table 6.

3.6 Comparing DeepDNAbP with existing DBP methods

In this section, the proposed DeepDNAbP method was compared with other existing DBP prediction methods, including DPP-PseAAC [13], iDNA-Prot [22], iDNA-Prot|dis [23], PseDNA-Pro [17], PSFM-DBT [74], IKP-DBPPred [5], Local-DPP [24], iDNAProt-ES [25] and TargetDBP [32], on the independent validation datasets. Here we considered PSSM features as experimental datasets to train the CNN model. The prediction outcomes of DPP-PseAAC [13], iDNA-Prot [22], iDNA-Prot|dis [23], PSFM-DBT [74], PseDNA-Pro [17], IKP-DBPPred [5], Local-DPP [24] and TargetDBP [32] were achieved by providing the 296 protein sequences to their respective web servers. Although the iDNAProt-ES [25] web server was not working, the outcomes of iDNAProt-ES were calculated using the standalone variant of iDNAProt-ES. Table 7 shows the performance comparisons between DeepDNAbP with existing DBP predictors.

According to the value F1 and MCC, two overall performance evaluation metrics recorded in Table 7, we can clearly see that DeepDNAbP demonstrated superior performance over other methods. Comparing our proposed DeepDNAbP method with the second most accurate predictor, TargetDBP, we see that DeepDNAbP showed improvements of 3.72%, 5.47%, 2.24%, 0.073, and 2.2% on Acc, Sen, Spe, MCC and F1 scores, respectively. It has not escaped our notice that iDNAProt-ES [25] obtained the highest Sen value (91.89%) and the lowest Spe value (51.35%). The reason for the high Sen is that iDNAProt-ES predicts fewer false negative results. In contrast, Local-DPP [24] has the highest specificity (93.92%) and shows a much smaller Rec value, denoting too many false negatives during prediction, explaining why Acc values of iDNAProt-ES [25] and Local-DPP [24] are lower than those of DeepDNAbP.

From the above observations and comparisons, we concluded that our proposed DeepDNAbP method outperforms the existing models in the literature so far. This indicates that our technique is a competitive tool for predicting DBPs and could be a significant improvement to existing methods.

4. Data and software availability

To examine the importance of DBPs, we have developed an online software tool DeepDNAbP, which is freely accessible for researchers and academic use at http://deepdbp.sblog360.blog/. The main goal of this web tool is to provide a simple process and a friendly user interface. To use the tool, users first need to input the protein sequence into the input box. After submission, the server will evaluate the protein sequence and check the format for processing. After completing the submitted predication task, the result will be displayed on a separate page, and users will be able to see the prediction probability of the protein.

The DeepDNAbP prediction will take longer when users submit a large number of protein sequences as the input, since the server will generate the PSSM matrix by performing PSI-BLAST (Position-Specific Iterative Basic Local Alignment Search Tool) and calculate the

PSSM encoding. To minimize computational resource limitations, we strongly recommended inputting fewer than 10 protein sequences at a time. The datasets employed in this study are available at http://deepdbp.sblog360.blog/data.

5. Conclusions

Accurate identification of DNA-binding proteins (DBPs) is essential in uncovering the internal mechanisms of protein–DNA interactions and understanding many biological processes. In this study, we have developed a new DBP predictor called DeepDNAbP. In DeepDNAbP, the input feature is first extracted by using three feature extraction methods: PSSM, PseAAC, and TPC. Next, we serially combine the input features to make a hybrid feature: PSSM+PseAAC, PSSM+TPC, PseAAC+TPC, PSSM+PseAAC+TPC. The algorithm of CNNs is then employed to build the DeepDNAbP model. Experimental results have shown that DeepDNAbP outperforms all other state-of-the-art methods. The principal reason for the superior performance of DeepDNAbP is our evolutionary-based feature representation, which provides more useful information to the CNN-based DBP identification model.

DeepDNAbP has certain limitations. To develop a more robust model for DBP prediction, future studies should focus on assembling a primary database of the experimentally identified DBPs, which is useful for computational biologists. Although the different feature learning advances the prediction performance, it is necessary to explore other recently reported computational approaches [75-79] to examine whether they improve the prediction performance of the proposed algorithm.

Conflict of Interest Statement

The authors declare that they have no known competing financial interests.

Acknowledgments

Md Mehedi Hasan is benefited by the support of NIH projects of U19AG05537301 and R01AR069055.

6. References

[1] Y. Dai *et al.*, “Application of bioconjugation chemistry on biosensor fabrication for detection of TAR-DNA binding protein 43,” *Biosensors and Bioelectronics*, vol. 117, pp. 60–67, Oct. 2018, doi: 10.1016/j.bios.2018.05.060.

[2] Q. Zhang, P. Liu, X. Wang, Y. Zhang, Y. Han, and B. Yu, “StackPDB: Predicting DNA-binding proteins based on XGB-RFE feature optimization and stacked ensemble classifier,” *Applied Soft Computing*, vol. 99, Feb. 2021, doi: 10.1016/j.asoc.2020.106921.

[3] B. Ren *et al.*, “Genome-Wide Location and Function of DNA Binding Proteins.” [Online]. Available: www.sciencemag.org

[4] R. E. Langlois and H. Lu, “Boosting the prediction and understanding of DNA-binding domains from sequence,” *Nucleic Acids Research*, vol. 38, no. 10, pp. 3149–3158, Feb. 2010, doi: 10.1093/nar/gkq061.

[5] K. Qu, K. Han, S. Wu, G. Wang, and L. Wei, “Identification of DNA-binding proteins using mixed feature representation methods,” *Molecules*, vol. 22, no. 10, Oct. 2017, doi: 10.3390/molecules22101602.

[6] M. Gao and J. Skolnick, “DBD-Hunter: A knowledge-based method for the prediction of DNA-protein interactions,” *Nucleic Acids Research*, vol. 36, no. 12, pp. 3978–3992, Jul. 2008, doi: 10.1093/nar/gkn332.

[7] G. Nimrod, A. Szilágyi, C. Leslie, and N. Ben-Tal, “Identification of DNA-binding Proteins Using Structural, Electrostatic and Evolutionary Features,” *Journal of Molecular Biology*, vol. 387, no. 4, pp. 1040–1053, Apr. 2009, doi: 10.1016/j.jmb.2009.02.023.

[8] G. Nimrod, M. Schushan, A. Szilágyi, C. Leslie, and N. Ben-Tal, “iDBPs: A web server for the identification of DNA binding proteins,” *Bioinformatics*, vol. 26, no. 5, pp. 692–693, Jan. 2010, doi: 10.1093/bioinformatics/btq019.

[9] Y. Wang, Y. Ding, F. Guo, L. Wei, and J. Tang, “Improved detection of DNA-binding proteins via compression technology on PSSM information,” *PLoS ONE*, vol. 12, no. 9, Sep. 2017, doi: 10.1371/journal.pone.0185587.

[10] G. B. Motion, A. J. M. Howden, E. Huitema, and S. Jones, “DNA-binding protein prediction using plant specific support vector machines: Validation and application of a new genome annotation tool,” *Nucleic Acids Research*, vol. 43, no. 22, Dec. 2015, doi: 10.1093/nar/gkv805.

[11] L. Nanni and A. Lumini, “Combing ontologies and dipeptide composition for predicting DNA-binding proteins,” *Amino Acids*, vol. 34, no. 4, pp. 635–641, May 2008, doi: 10.1007/s00726-007-0016-3.

[12] S. Adilina, D. M. Farid, and S. Shatabda, “Effective DNA binding protein prediction by using key features via Chou’s general PseAAC,” *Journal of Theoretical Biology*, vol. 460, pp. 64–78, Jan. 2019, doi: 10.1016/j.jtbi.2018.10.027.

[13] M. S. Rahman, S. Shatabda, S. Saha, M. Kaykobad, and M. S. Rahman, “DPP-PseAAC: A DNA-binding protein prediction model using Chou’s general PseAAC,” *Journal of Theoretical Biology*, vol. 452, pp. 22–34, Sep. 2018, doi: 10.1016/j.jtbi.2018.05.006.

[14] X. Fu, W. Zhu, B. Liao, L. Cai, L. Peng, and J. Yang, “Improved DNA-Binding protein identification by incorporating evolutionary information into the Chou’s PseAAC,” *IEEE Access*, vol. 6, pp. 66545–66556, 2018, doi: 10.1109/ACCESS.2018.2876656.

[15] D. IEEE International Conference on Bioinformatics and Biomedicine 2015 Washington *et al.*, *2015 IEEE International Conference on Bioinformatics and Biomedicine Nov. 9-12, 2015, Washington DC, USA : proceedings*.

[16] B. Liu, S. Wang, Q. Dong, S. Li, and X. Liu, “Identification of DNA-Binding Proteins by Combining Auto-Cross Covariance Transformation and Ensemble Learning,” *IEEE Transactions on Nanobioscience*, vol. 15, no. 4, pp. 328–334, Jun. 2016, doi: 10.1109/TNB.2016.2555951.

[17] B. Liu, J. Xu, S. Fan, R. Xu, J. Zhou, and X. Wang, “PseDNA-Pro: DNA-binding protein identification by combining chou’s PseAAC and Physicochemical distance transformation,” *Molecular Informatics*, vol. 34, no. 1, pp. 8–17, 2015, doi: 10.1002/minf.201400025.

[18] M. Kumar, M. M. Gromiha, and G. P. S. Raghava, “Identification of DNA-binding proteins using support vector machines and evolutionary profiles,” *BMC Bioinformatics*, vol. 8, Nov. 2007, doi: 10.1186/1471-2105-8-463.

[19] M. Waris, K. Ahmad, M. Kabir, and M. Hayat, “Identification of DNA binding proteins using evolutionary profiles position specific scoring matrix,” *Neurocomputing*, vol. 199, pp. 154–162, Jul. 2016, doi: 10.1016/j.neucom.2016.03.025.

[20] B. Liu, S. Wang, and X. Wang, “DNA binding protein identification by combining pseudo amino acid composition and profile-based protein representation,” *Scientific Reports*, vol. 5, Oct. 2015, doi: 10.1038/srep15479.

[21] R. Zaman, S. Y. Chowdhury, M. A. Rashid, A. Sharma, A. Dehzangi, and S. Shatabda, “HMMBinder: DNA-Binding Protein Prediction Using HMM Profile Based Features,” *BioMed Research International*, vol. 2017, 2017, doi: 10.1155/2017/4590609.

[22] W. Z. Lin, J. A. Fang, X. Xiao, and K. C. Chou, “iDNA-prot: Identification of DNA binding proteins using random forest with grey model,” *PLoS ONE*, vol. 6, no. 9, Sep. 2011, doi: 10.1371/journal.pone.0024756.

[23] B. Liu *et al.*, “IDNA-Prot|dis: Identifying DNA-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition,” *PLoS ONE*, vol. 9, no. 9, Sep. 2014, doi: 10.1371/journal.pone.0106691.

[24] L. Wei, J. Tang, and Q. Zou, “Local-DPP: An improved DNA-binding protein prediction method by exploring local evolutionary information,” *Information Sciences*, vol. 384, pp. 135–144, Apr. 2017, doi: 10.1016/j.ins.2016.06.026.

[25] S. Y. Chowdhury, S. Shatabda, and A. Dehzangi, “IDNAProt-ES: Identification of DNA-binding Proteins Using Evolutionary and Structural Features,” *Scientific Reports*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/s41598-017-14945-1.

[26] K. C. Chou, “Some remarks on protein attribute prediction and pseudo amino acid composition,” *Journal of Theoretical Biology*, vol. 273, no. 1. pp. 236–247, Mar. 21, 2011. doi: 10.1016/j.jtbi.2010.12.024.

[27] K.-C. Chou, “Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition,” 2001.

[28] D. Julong Deynrt, “Introduction to Grey System Theory.”

[29] M. Remmert, A. Biegert, A. Hauser, and J. Söding, “HHblits: Lightning-fast iterative protein sequence searching by HMM-HMM alignment,” *Nature Methods*, vol. 9, no. 2, pp. 173–175, Feb. 2012, doi: 10.1038/nmeth.1818.

[30] P. W. Rose *et al.*, “The RCSB Protein Data Bank: Views of structural biology for basic and applied research and education,” *Nucleic Acids Research*, vol. 43, no. D1, pp. D345–D356, Jan. 2015, doi: 10.1093/nar/gku1214.

[31] R. Storn and K. Price, “Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” Kluwer Academic Publishers, 1997.

[32] J. Hu, X. G. Zhou, Y. H. Zhu, D. J. Yu, and G. J. Zhang, “TargetDBP: Accurate DNA-Binding Protein Prediction Via Sequence-Based Multi-View Feature Learning,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 4, pp. 1419–1429, Jul. 2020, doi: 10.1109/TCBB.2019.2893634.

[33] K. C. Chou and H. bin Shen, “Recent progress in protein subcellular location prediction,” *Analytical Biochemistry*, vol. 370, no. 1. Academic Press Inc., pp. 1–16, Nov. 01, 2007. doi: 10.1016/j.ab.2007.07.006.

[34] X. He *et al.*, “TargetFreeze: Identifying Antifreeze Proteins via a Combination of Weights using Sequence Evolutionary Information and Pseudo Amino Acid Composition,” *Journal of Membrane Biology*, vol. 248, no. 6, pp. 1005–1014, Dec. 2015, doi: 10.1007/s00232-015-9811-z.

[35] S. F. Altschul *et al.*, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” Oxford University Press, 1997. [Online]. Available: http://nar.oxfordjournals.org/

[36] S. F. Altschul and E. v. Koonin, “Iterated profile searches with PSI-BLAST - A tool for discovery in protein databases,” *Trends in Biochemical Sciences*, vol. 23, no. 11. pp. 444–447, Nov. 01, 1998. doi: 10.1016/S0968-0004(98)01298-5.

[37] M. Hayat and A. Khan, “MemHyb: Predicting membrane protein types by hybridizing SAAC and PSSM,” *Journal of Theoretical Biology*, vol. 292, pp. 93–102, Jan. 2012, doi: 10.1016/j.jtbi.2011.09.026.

[38] T. Liu, X. Zheng, and J. Wang, “Prediction of protein structural class for low-similarity sequences using support vector machine and PSI-BLAST profile,” *Biochimie*, vol. 92, no. 10, pp. 1330–1334, Oct. 2010, doi: 10.1016/j.biochi.2010.06.013.

[39] A. A. Schäffer *et al.*, “Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements,” 2001.

[40] A. Sharma, J. Lyons, A. Dehzangi, and K. K. Paliwal, “A feature extraction technique using bi-gram probabilities of position specific scoring matrix for protein fold recognition,” *Journal of Theoretical Biology*, vol. 320, pp. 41–46, Mar. 2013, doi: 10.1016/j.jtbi.2012.12.008.

[41] P. Ung and D. A. Winkler, “Tripeptide motifs in biology: Targets for peptidomimetic design,” *Journal of Medicinal Chemistry*, vol. 54, no. 5, pp. 1111–1125, Mar. 2011, doi: 10.1021/jm1012984.

[42] S. Anishetty, G. Pennathur, and R. Anishetty, “Tripeptide analysis of protein structures,” 2002. [Online]. Available: http://www.au-

[43] C. Ding, L. F. Yuan, S. H. Guo, H. Lin, and W. Chen, “Identification of mycobacterial membrane proteins and their types using over-represented tripeptide compositions,” *Journal of Proteomics*, vol. 77, pp. 321–328, 2012, doi: 10.1016/j.jprot.2012.09.006.

[44] H. Lin, W. Chen, L. F. Yuan, Z. Q. Li, and H. Ding, “Using Over-Represented Tetrapeptides to Predict Protein Submitochondria Locations,” *Acta Biotheoretica*, vol. 61, no. 2, pp. 259–268, Jun. 2013, doi: 10.1007/s10441-013-9181-9.

[45] W. X. Liu, E. Z. Deng, W. Chen, and H. Lin, “Identifying the subfamilies of voltage-gated potassium channels using feature selection technique,” *International Journal of Molecular Sciences*, vol. 15, no. 7, pp. 12940–12951, Jul. 2014, doi: 10.3390/ijms150712940.

[46] P. P. Zhu *et al.*, “Predicting the subcellular localization of mycobacterial proteins by incorporating the optimal tripeptides into the general form of pseudo amino acid composition,” *Molecular BioSystems*, vol. 11, no. 2, pp. 558–563, Feb. 2015, doi: 10.1039/c4mb00645c.

[47] I. Bahar, A. R. Atilgan, R. L. Jernigan, and B. Erman, “Understanding the Recognition of Protein Structural Classes by Amino Acid Composition,” Wiley-Liss, Inc. †, 1997.

[48] G.-P. Zhou and K. Doctor, “Subcellular Location Prediction of Apoptosis Proteins,” 2003.

[49] K. C. Chou, “Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes,” *Bioinformatics*, vol. 21, no. 1, pp. 10–19, Jan. 2005, doi: 10.1093/bioinformatics/bth466.

[50] C. Chen, X. Zhou, Y. Tian, X. Zou, and P. Cai, “Predicting protein structural class with pseudo-amino acid composition and support vector machine fusion network,” *Analytical Biochemistry*, vol. 357, no. 1, pp. 116–121, Oct. 2006, doi: 10.1016/j.ab.2006.07.022.

[51] M. Esmaeili, H. Mohabatkar, and S. Mohsenzadeh, “Using the concept of Chou’s pseudo amino acid composition for risk type prediction of human papillomaviruses,” *Journal of Theoretical Biology*, vol. 263, no. 2, pp. 203–209, Mar. 2010, doi: 10.1016/j.jtbi.2009.11.016.

[52] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” May 2017, [Online]. Available: http://arxiv.org/abs/1705.07874

[53] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, “Deep learning for plant identification using vein morphological patterns,” *Computers and Electronics in Agriculture*, vol. 127, pp. 418–424, Sep. 2016, doi: 10.1016/j.compag.2016.07.003.

[54] P. Luo, Y. Ding, X. Lei, and F. X. Wu, “DeepDriver: Predicting cancer driver genes based on somatic mutations using deep convolutional neural networks,” *Frontiers in Genetics*, vol. 10, no. JAN, 2019, doi: 10.3389/fgene.2019.00013.

[55] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[56] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.”

[57] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” 2014.

[58] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” Mar. 2016, [Online]. Available: http://arxiv.org/abs/1603.04467

[59] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” 2011. [Online]. Available: http://scikit-learn.sourceforge.net.

[60] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007, doi: 10.1109/MCSE.2007.55

[61] M. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/joss.03021.

[62] Plotly Technologies Inc. Collaborative data science. Montréal, QC, 2015. https://plot.ly.

[63] C. Cortes and V. Vapnik, ‘‘Support-vector networks,’’ Mach. Learn., vol. 297, no. 20, pp. 273–297, 1995.

[64] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *American Statistician*, vol. 46, no. 3, pp. 175–185, 1992, doi: 10.1080/00031305.1992.10475879.

[65] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, vol. 13-17-August-2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[66] A.-A. Christidis, S. van Aelst, and R. Zamar, "Data-Driven Diverse Logistic Regression Ensembles," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.08591

[67] S. M. H. Mahmud *et al.*, "PreDTIs: prediction of drug–target interactions based on multiple feature information using gradient boosting framework with data balancing and feature selection techniques," *Briefings in Bioinformatics*, Mar. 2021, doi: 10.1093/bib/bbab046.

[68] S. M. Hasan Mahmud, W. Chen, H. Jahan, B. Dai, S. U. Din, and A. M. Dzisoo, "DeepACTION: A deep learning-based method for predicting novel drug-target interactions," *Analytical Biochemistry*, vol. 610, Dec. 2020, doi: 10.1016/j.ab.2020.113978.

[69] S. M. H. Mahmud, W. Chen, H. Meng, H. Jahan, Y. Liu, and S. M. M. Hasan, "Prediction of drug-target interaction based on protein features using undersampling and feature selection techniques with boosting," *Analytical Biochemistry*, vol. 589, Jan. 2020, doi: 10.1016/j.ab.2019.113507.

[70] K. Pearson, " LIII. On lines and planes of closest fit to systems of points in space ," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, Nov. 1901, doi: 10.1080/14786440109462720.

[71] L. M. Leemis and J. T. McQueston, "Univariate distribution relationships," *American Statistician*, vol. 62, no. 1, pp. 45–53, Feb. 2008, doi: 10.1198/000313008X270448.

[72] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 231–245, 2013, doi: 10.1080/10618600.2012.681250.

[73] H. Deng and G. Runger, "Gene selection with guided regularized random forest," *Pattern Recognition*, vol. 46, no. 12, pp. 3483–3489, Dec. 2013, doi: 10.1016/j.patcog.2013.05.018.

[74] S. Basith, M. M. Hasan, G. Lee, L. Wei, B. Manavalan, Integrative machine learning framework for the identification of cell-specific enhancers from the human genome, Briefings in Bioinformatics, Volume 22, Issue 6, November 2021, bbab252, https://doi.org/10.1093/bib/bbab252

[75] M. M. Hasan, M. A. Alam, W. Shoombuatong, H.-W. Deng, B. Manavalan, and H. Kurata, "NeuroPred-FRL: an interpretable prediction model for identifying neuropeptide using feature representation learning," *Briefings in Bioinformatics*, May 2021, doi: 10.1093/bib/bbab167.

[76] M. M. Hasan, N. Schaduangrat, S. Basith, G. Lee, W. Shoombuatong, B.Manavalan HLPpred-Fuse: improved and robust prediction of hemolytic peptide and its activity by fusing multiple feature representation. Bioinformatics. 2020 Jun 1;36(11):3350-3356.

[77] M. M. Hasan, Hasan, S. Basith, M.S. Khatun, G Lee, B. Manavalan, H. Kurata. Meta-i6mA: an interspecies predictor for identifying DNA N6-methyladenine sites of plant genomes by exploiting informative features in an integrative machine-learning framework. Brief Bioinform. 2021 May 20;22(3):bbaa202.

[78] P. Charoenkwan, W. Chiangjong, C. Nantasenamat, M. M. Hasan, B. Manavalan, W. Shoombuatong, StackIL6: a stacking ensemble model for improving the prediction of IL-6 inducing peptides, *Briefings in Bioinformatics*, Volume 22, Issue 6, November 2021, bbab172, https://doi.org/10.1093/bib/bbab172.

[79] B. Manavalan, M. M. Hasan, S. Basith, V.Gosu, T. Shin, G. Lee, Empirical Comparison and Analysis of Web-Based DNA N4-Methylcytosine Site Prediction Tools, Molecular Therapy - Nucleic Acids,22: 406-420 (2020)

Figure Legends

Figure 1. Overall framework of DeepDNAbP

Figure 2. Performances of CNN classifier on PSSM, PseAAC, and TPC features

Figure 3. Performances of different combined feature groups using CNN

Figure 4. Performances of three feature groups on the different classifiers on various features (A) PSSM (B) PseAAC (C) TPC

Figure 5. SHAP feature importance bar plot showing the most important features for different feature dimensions: (A) 200 features and (B) 250 features

Figure 6. SHAP summary plot showing a high-level composite view that displays the importance of features with feature effects: (A) 200 features (B) 250 features

Table. 1 Cross-Validation results of CNN with three Single-View Features

Features	**AUC**	**Acc**	**Sen**	**Spe**	**MCC**	**Pre**	**F1**
PSSM	87.14%	81.25%	90.00%	73.53%	0.64	75.00%	81.82%
PseAAC	80.65%	73.44%	71.05%	76.92%	0.47	81.82%	76.06%
TPC	77.61%	71.88%	75.86%	68.57%	0.44	66.67%	70.97%

Table 2 Performance of the CNN classifier of different combined feature groups

Feature	AUC	Acc	Sen	Spe	MCC	Pre	F1
PSSM+ PseAAC	83.78%	78.20%	77.32%	78.95%	0.56	75.76%	76.53%
PSSM+TPC	80.41%	72.52%	75.00%	70.27%	0.45	69.44%	72.12%
PseAAC+TPC	77.90%	72.04%	72.07%	72.00%	0.44	74.07%	73.06%
PSSM +PseAAC+ TPC	80.59%	74.88%	74.77%	75.00%	0.50	76.85%	75.80%

Table 3 Performance of different classifiers and feature groups on the dataset

Classifiers	Features	AUC	Acc (%)	Sen (%)	Spe (%)	MCC	Precision	F1 measure
KNN	PSSM	81.89%	76.56%	83.87%	69.70%	0.54	72.22%	77.61%
	PseAAC	74.93%	73.44%	70.00%	79.17%	0.48	84.85%	76.71%
	TPC	75.45%	70.62%	71.76%	69.84%	0.41	61.62%	66.30%
SVM	PSSM	87.70%	81.25%	90.00%	73.53%	0.64	75.00%	81.82%
	PseAAC	83.48%	75.00%	77.42%	72.73%	0.50	72.73%	75.00%
	TPC	77.82%	72.04%	74.75%	69.64%	0.44	68.52%	71.50%
XGBoost	PSSM	88.32%	82.81%	85.71%	79.31%	0.65	83.33%	84.51%
	PseAAC	82.62%	76.56%	73.68%	80.77%	0.54	84.85%	78.87%
	TPC	81.39%	74.41%	70.64%	78.43%	0.49	77.78%	74.04%
CNN	PSSM	89.68%	85.78%	90.63%	78.13%	0.70	80.56%	85.30%
	PseAAC	83.48%	78.13%	74.36%	84.00%	0.58	87.88%	80.56%
	TPC	82.19%	74.88%	75.00%	74.79%	0.50	69.70%	72.26%
LRE	PSSM	80.75%	76.38%	74.29%	78.95%	0.53	81.25%	77.61%
	PseAAC	80.92%	75.47%	74.55%	76.47%	0.51	77.36%	75.93%
	TPC	74.26%	72.44%	72.31%	72.58%	0.45	73.44%	72.87%

Table 4 Prediction results of the feature selection methods on different setting for PSSM dataset

Classifier	**Feature Groups**	**No. of features**	**AUC**	**Acc%**	**Sen%**	**Spe%**	**MCC%**	**Precision**	**F1%**
CNN	**PCA**	150	81.30%	74.42%	77.27%	71.43%	**0.49**	73.91%	75.56%
		200	87.83%	79.07%	88.88%	72.00%	**0.60**	69.57%	78.05%
		250	82.83%	76.74%	80.95%	72.73%	**0.54**	73.91%	77.27%
		300	85.43%	82.72%	90.00%	78.26%	**0.68**	78.26%	83.72%
		350	83.53%	75.00%	83.33%	67.64%	**0.51**	69.44%	75.75%
	Univarite FST	150	85.33%	81.17%	82.98%	78.95%	**0.62**	82.98%	82.98%
		200	89.36%	84.38%	90.63%	78.13%	**0.69**	80.56%	85.29%
		250	87.68%	82,35%	84.78%	79.49%	**0.64**	82.98%	83.87%
		300	84.77%	80.00%	78.85%	81.82%	**0.59**	87.23%	82.83%
		350	85.11%	78.82%	79.59%	77.78%	**0.57**	82.98%	81.25%
	SHAP	150	85.91%	81.25%	87.50%	75.00%	**0.63**	77.78%	82.35%
		200	89.68%	85.78%	90.63%	78.13%	**0.70**	80.56%	85.30%
		250	89.29%	84.38%	88.24%	80.00%	**0.69**	83.33%	85.71%
		300	85.81%	82.81%	87.88%	77.42%	**0.66**	80.56%	84.06%
		350	88.98%	79.69%	89.66%	71.43%	**0.61**	72.22%	79.99%
	SGL	150	81.60%	73.93%	73.87%	74.00%	**0.48**	75.93%	74.89%
		200	83.33%	75.36%	76.42%	74.29%	**0.51**	75.00%	75.70%
		250	84.87%	78.20%	78.18%	78.22%	**0.56**	79.63%	78.90%
		300	82.26%	77.25%	76.79%	77.78%	**0.54**	79.63%	78.18%
		350	81.31%	74.41%	74.11%	74.75%	**0.49**	76.85%	75.45%
	RRF	150	80.70%	73.45%	72.98%	73.75%	**0.47**	74.33%	73.89%
		200	82.90%	74.87%	75.98%	74.01%	**0.50**	74.88%	74.78%
		250	84.10%	78.01%	78.78%	78.01%	**0.55**	78.87%	78.12%
		300	82.00%	78.95%	75.87%	76.98%	**0.55**	78.60%	77.98%
		350	81.01%	74.11%	74.01%	73.03%	**0.50**	75.88%	74.60%

Table 5 CNN model Hyperparameters

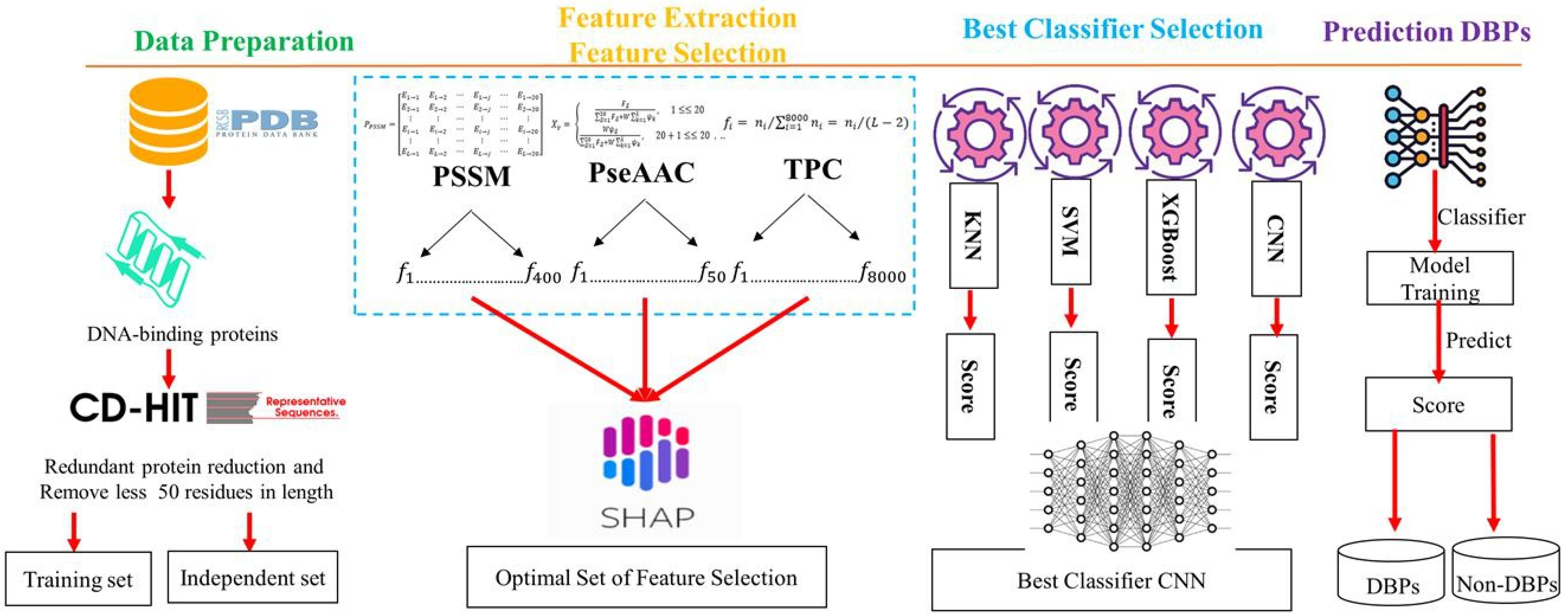
Hyperparameters	**Range**
Learning rate	[**0.00001**,0.01,0.001,0.0001]
Batch Size	[16,**32**,64,128]
Number of Filters	[32,**48**,**64,**96]
Kernel Size	[3,**4,**5]
Number of Hidden layers	[**2**,3]
Optimizer	[**'Adam'**]
Dropout rate	[0.2, **0.5**]
Activation	[**'relu'**,**'sigmoid'**]

Table 6 Parameters setting of KNN, SVM, and XGBoost

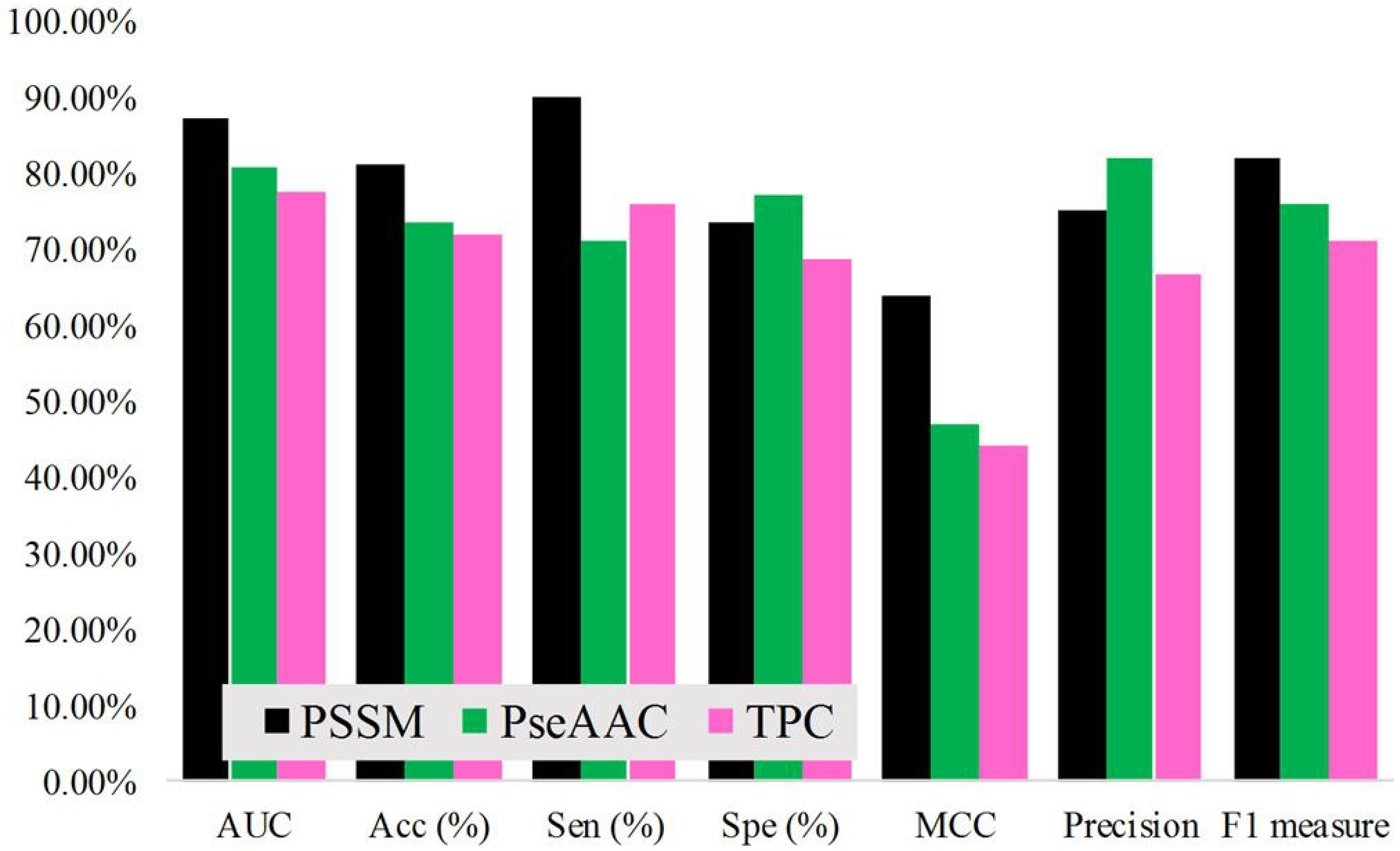
Classifiers	Feature Extraction	Parameters
KNN	PSSM	KNeighborsClassifier (n_neighbors = 8, leaf_size=50, metric = 'minkowski', p = 2)
	PseAAC	KNeighborsClassifier (n_neighbors = 10, leaf_size=50, metric = 'minkowski', p = 2)
	TPC	KNeighborsClassifier (n_neighbors = 5, leaf_size=50, metric = 'minkowski', p = 2)
SVM	PSSM	SVC (kernel='rbf', C=15, gamma=0.0001, probability=True)
	PseAAC	SVC (kernel='rbf', C=50, gamma=0.0001, probability=True)
	TPC	SVC (kernel='rbf', C=10, gamma=0.0001, probability=True)
XGBoost	PSSM	PSSM:XGBClassifier (learning_rate=0.001, subsample = 1, n_estimators=1000, max_depth=10, gamma=10)
	PseAAC	XGBClassifier (learning_rate=0.01, subsample = 0.8, n_estimators=1000, max_depth=4, gamma=10)
	TPC	XGBClassifier (learning_rate=0.1, subsample = 1, n_estimators=1000, max_depth=4, gamma=10)

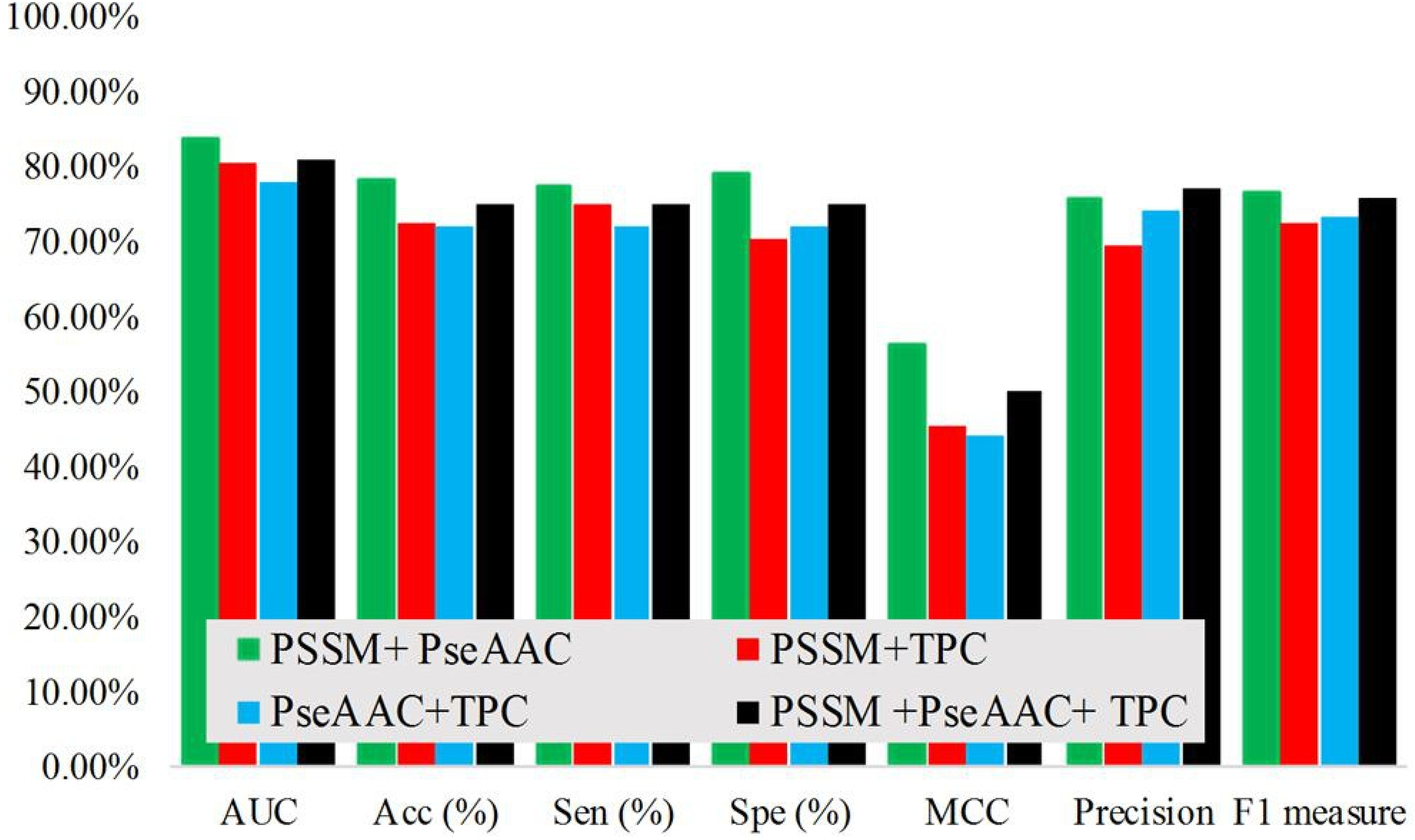
Table 7 Performance Comparisons of DeepDNAbP with existing methods on the Independent Validation Dataset

Predictor	Acc	Sen	Spe	MCC	Pre	F1
DPP-PseAAC [13]	61.15	55.41	66.89	0.225	62.60	0.588
iDNA-Prot [22]	62.16	63.51	60.81	0.243	61.84	0.627
iDNA-Prot|dis [23]	68.24	72.30	64.19	0.366	66.88	0.695
PseDNA-Pro [17]	67.23	78.38	56.08	0.354	64.09	0.705
PSFM-DBT [74]	68.58	71 .62	65.54	0.372	67.52	0.695
IKP-DBPPred [5]	58.11	52.70	63.51	0.163	59.09	0.557
Local-DPP [24]	48.65	3.38	93.92	-0.06	35.71	0.062
iDNAProt-ES(on PDB1075)[25]	71.62	91 .89	51.35	0.473	65.38	0.764
TargetDBP [32]	76.69	76.35	77.03	0.534	76.87	0.766
DeepDNAbP	**80.41**	**81.82**	**79.27**	**0.607**	**76.06**	**0.788**



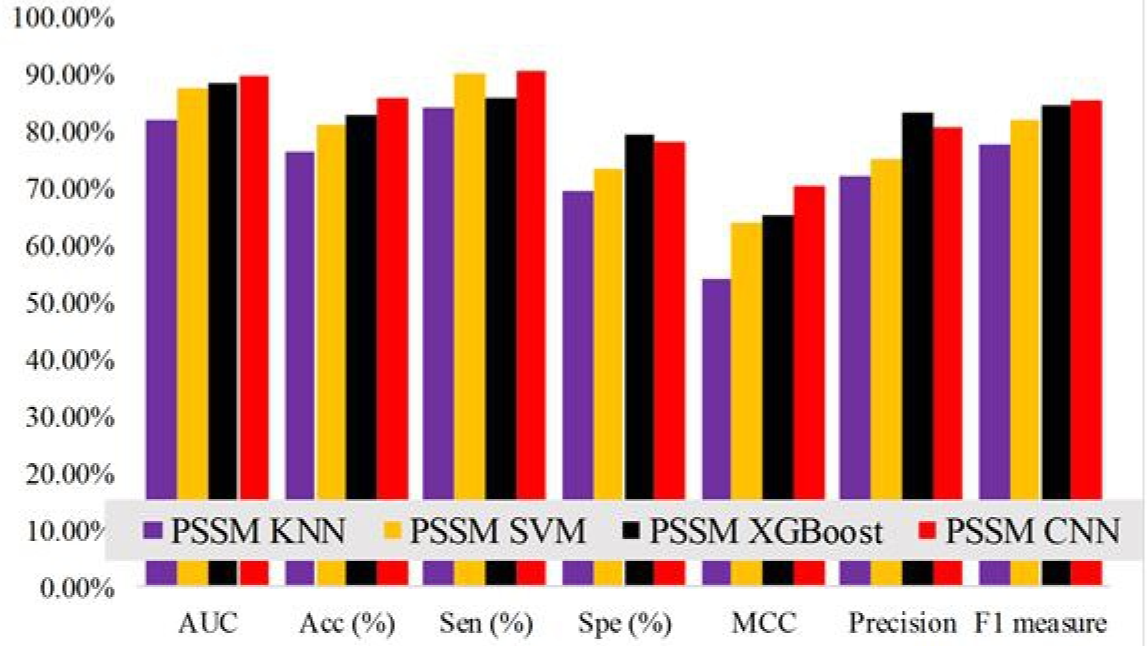
Data Preparation
Feature Extraction
Feature Selection
Best Classifier Selection
Prediction DBPs
PDB PROTEIN DATA BANK
DNA-binding proteins
CD-HIT
Representative Sequences.
Redundant protein reduction and Remove less 50 residues in length
Training set
Independent set
PSSM
PseAAC
TPC
$f_1 \ldots f_{400}$
$f_1 \ldots f_{50}$
$f_1 \ldots f_{8000}$
$f_i = n_i/\sum_{i=1}^{8000} n_i = n_i/(L-2)$
SHAP
Optimal Set of Feature Selection
KNN
SVM
XGBoost
CNN
Score
Score
Score
Score
Best Classifier CNN
Classifier
Model Training
Predict
Score
DBPs
Non-DBPs


100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
PSSM
PseAAC
TPC
AUC
Acc (%)
Sen (%)
Spe (%)
MCC
Precision
F1 measure



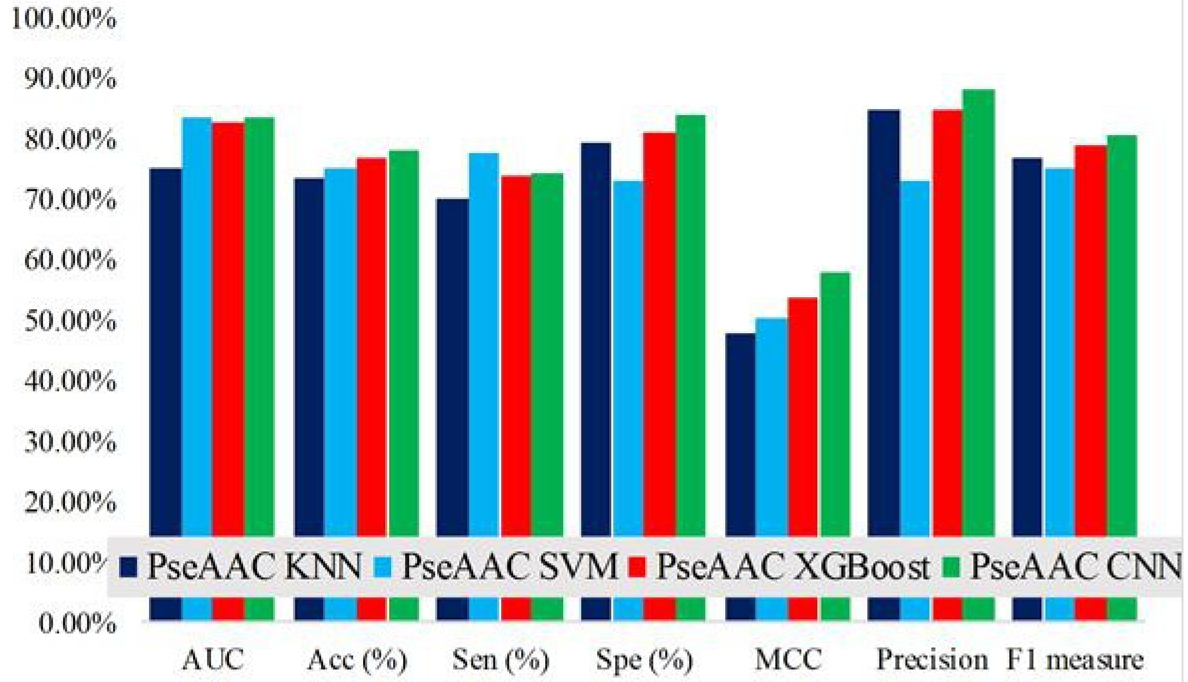
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
PSSM+ PseAAC
PSSM+TPC
PseAAC+TPC
PSSM +PseAAC+ TPC
AUC
Acc (%)
Sen (%)
Spe (%)
MCC
Precision
F1 measure

A



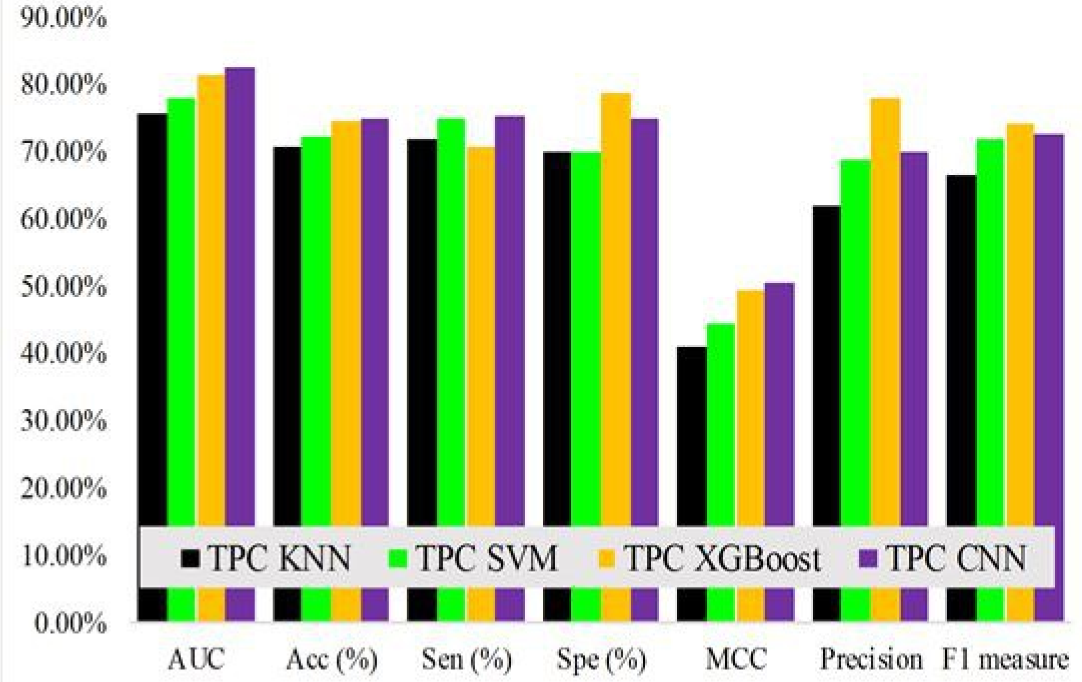
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
PSSM KNN
PSSM SVM
PSSM XGBoost
PSSM CNN
AUC
Acc (%)
Sen (%)
Spe (%)
MCC
Precision
F1 measure

B



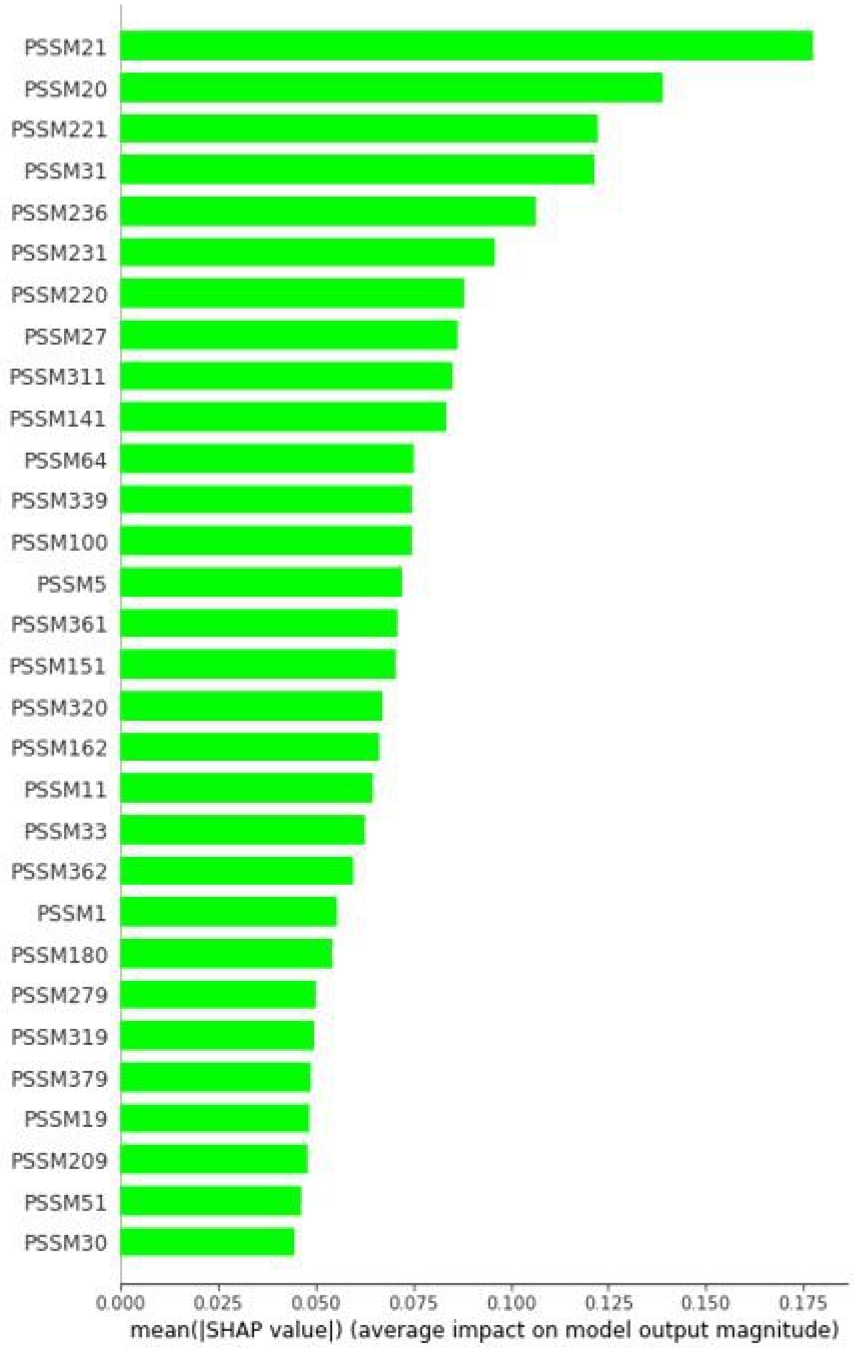
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
PseAAC KNN
PseAAC SVM
PseAAC XGBoost
PseAAC CNN
AUC
Acc (%)
Sen (%)
Spe (%)
MCC
Precision
F1 measure

C

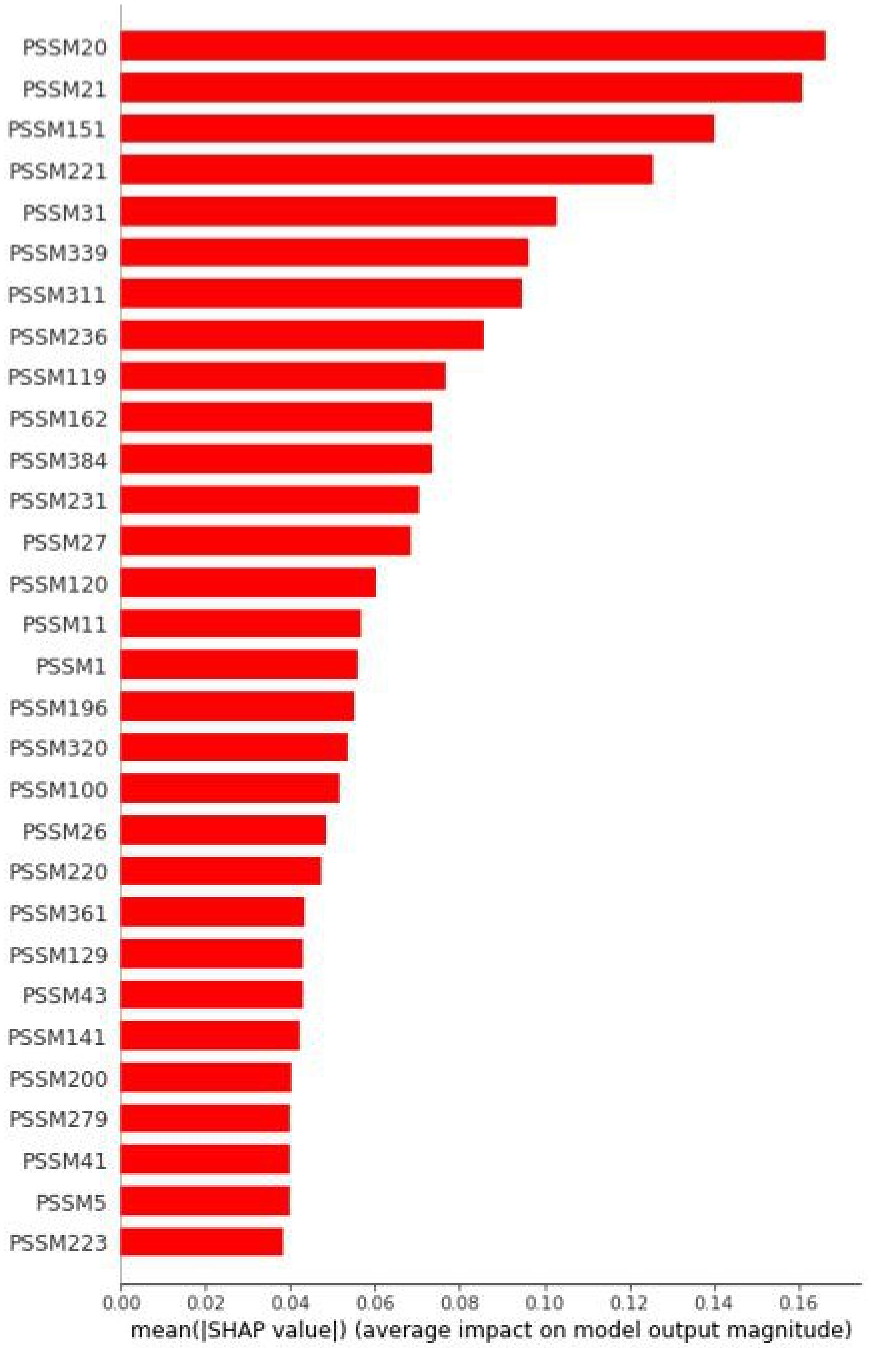


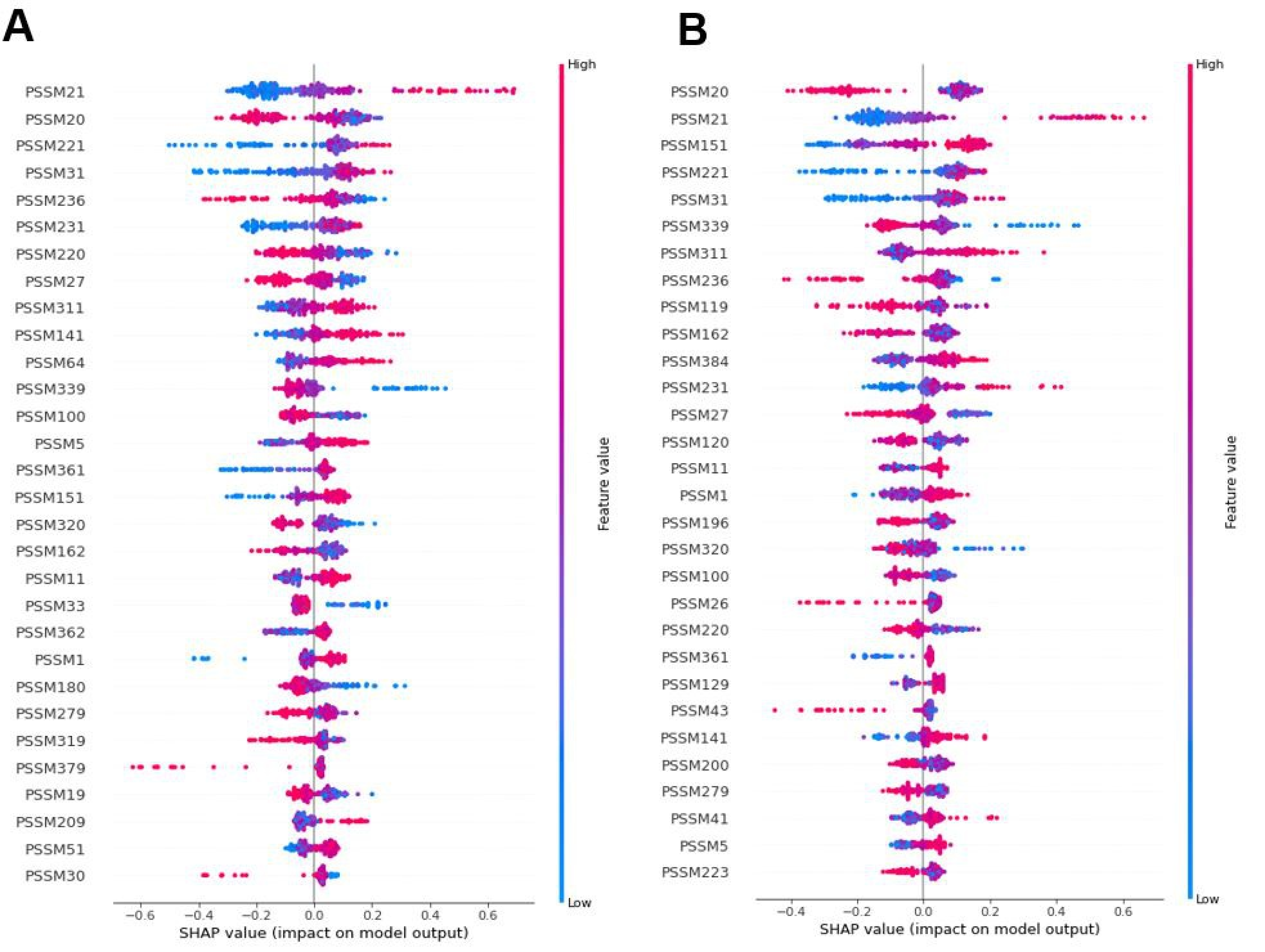
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
TPC KNN
TPC SVM
TPC XGBoost
TPC CNN
AUC
Acc (%)
Sen (%)
Spe (%)
MCC
Precision
F1 measure

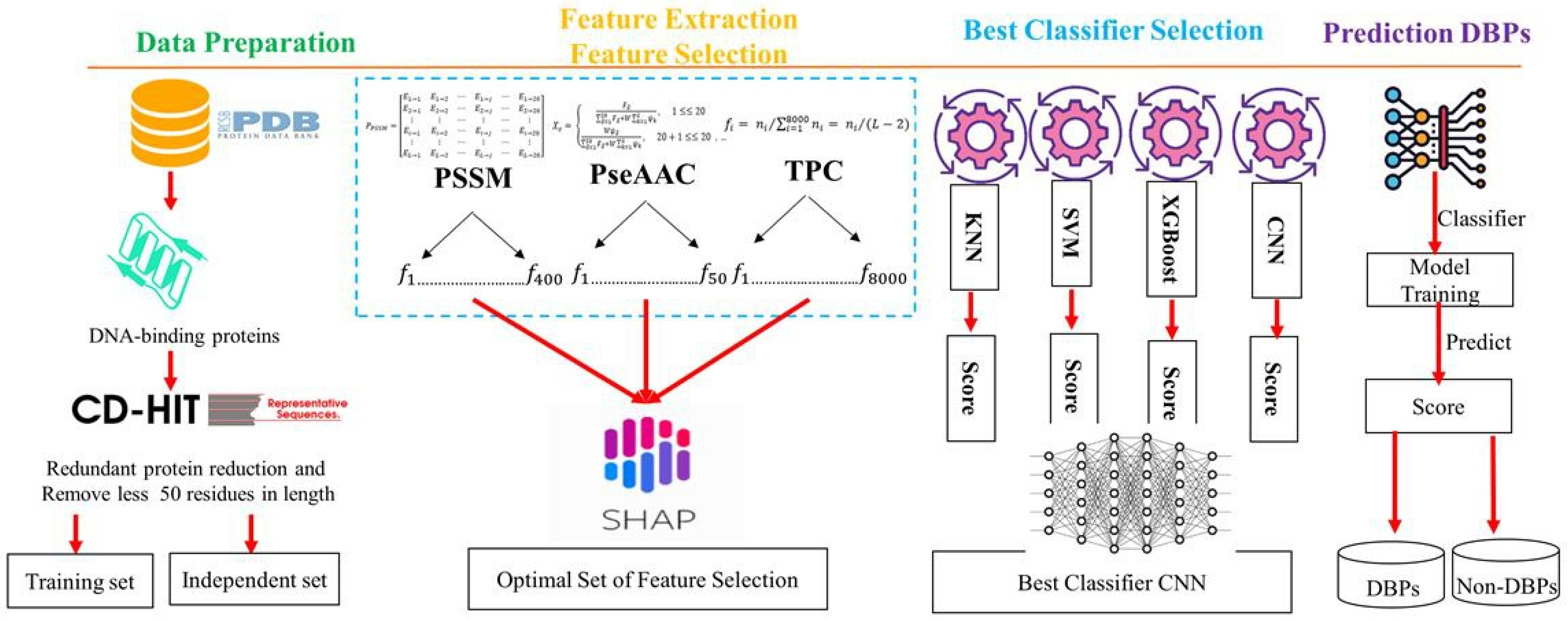
A


PSSM21
PSSM20
PSSM221
PSSM31
PSSM236
PSSM231
PSSM220
PSSM27
PSSM311
PSSM141
PSSM64
PSSM339
PSSM100
PSSM5
PSSM361
PSSM151
PSSM320
PSSM162
PSSM11
PSSM33
PSSM362
PSSM1
PSSM180
PSSM279
PSSM319
PSSM379
PSSM19
PSSM209
PSSM51
PSSM30
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175
mean(|SHAP value|) (average impact on model output magnitude)

B


PSSM20
PSSM21
PSSM151
PSSM221
PSSM31
PSSM339
PSSM311
PSSM236
PSSM119
PSSM162
PSSM384
PSSM231
PSSM27
PSSM120
PSSM11
PSSM1
PSSM196
PSSM320
PSSM100
PSSM26
PSSM220
PSSM361
PSSM129
PSSM43
PSSM141
PSSM200
PSSM279
PSSM41
PSSM5
PSSM223
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16
mean(|SHAP value|) (average impact on model output magnitude)


A
B
PSSM21
PSSM20
PSSM221
PSSM31
PSSM236
PSSM231
PSSM220
PSSM27
PSSM311
PSSM141
PSSM64
PSSM339
PSSM100
PSSM5
PSSM361
PSSM151
PSSM320
PSSM162
PSSM11
PSSM33
PSSM362
PSSM1
PSSM180
PSSM279
PSSM319
PSSM379
PSSM19
PSSM209
PSSM51
PSSM30
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6
SHAP value (impact on model output)
High
Low
Feature value
PSSM20
PSSM21
PSSM151
PSSM221
PSSM31
PSSM339
PSSM311
PSSM236
PSSM119
PSSM162
PSSM384
PSSM231
PSSM27
PSSM120
PSSM11
PSSM1
PSSM196
PSSM320
PSSM100
PSSM26
PSSM220
PSSM361
PSSM129
PSSM43
PSSM141
PSSM200
PSSM279
PSSM41
PSSM5
PSSM223
−0.4 −0.2 0.0 0.2 0.4 0.6
SHAP value (impact on model output)
High
Low
Feature value


Data Preparation
Feature Extraction
Feature Selection
Best Classifier Selection
Prediction DBPs
PDB
DNA-binding proteins
CD-HIT
Representative Sequences.
Redundant protein reduction and Remove less 50 residues in length
Training set
Independent set
PSSM
PseAAC
TPC
f_1 ... f_{400}
f_1 ... f_{50}
f_1 ... f_{8000}
$f_i = n_i/\sum_{i=1}^{8000} n_i = n_i/(L-2)$
SHAP
Optimal Set of Feature Selection
KNN
SVM
XGBoost
CNN
Score
Best Classifier CNN
Classifier
Model Training
Predict
Score
DBPs
Non-DBPs