Predicting DNA Binding Proteins with Machine Learning and Deep Learning

Noel Teo Yu Hong[1], Lucius Khor Zhe Yi[2], Del Mundo Nino Hubert Atienza[3], Ryan Phua Wei Jie[4]

National University of Singapore

e0958006@u.nus.edu[1], e0985941@u.nus.edu[2], e0938591@u.nus.edu[3], e0970120@u.nus.edu[4]

1 Introduction

DNA Binding Proteins (DBPs) are essential and crucial cellular proteins that perform important functions within cellular activities such as transcription, translation and DNA repair (Hudson & Ortlund, 2014). Therefore, correctly classifying DBPs are beneficial for drug design and healthcare functions (Ali et. al., 2019).

There has been extensive research into applying Machine Learning (ML) models for DBP prediction. Various ML models were utilised to predict the classes of DNA Binding proteins, with varying results of accuracy. For example, Asghari et al. (2019) analysed a training dataset consisting of DBPs, RNA Binding Proteins (RBP) and non-nucleic binding proteins (Ctrl). They used and trained 9 ML models (including K-Nearest Neighbour (KNN) and L1 Regularized Logistic Regression) finding the model having highest accuracy to be ADTree with 0.956, and the model with the lowest accuracy being SMO with a score of 0.945.

Furthermore, recent Deep Learning (DL) algorithms have also been introduced to predict DBPs more accurately as compared to ML algorithms (Bhandari e.t. al., 2021). Currently, most deep learning methods employ either Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN) based architecture for prediction of DBPs, with the latter being more widespread.

Traditional wet-methods of DBP classification have been known to be time-consuming, costly and sometimes unreliable (Wang, 2017). Therefore, the goal of our project is to compare the performance of different Machine Learning and Deep Learning models that are trained to distinguish DNA Binding Proteins from non-DNA Binding Proteins in an efficient, accurate and cost-efficient way.

In this report, we will discuss three different ML algorithms – K-Nearest Neighbours (KNN), Logistic Regression and eXtreme Gradient Boosting (XGB) and one deep learning model. We will assess each model's performance using Accuracy, Sensitivity, Specificity and Matthews Correlation Coefficient (MCC) score.

2 Dataset

Our dataset consists of 2 CSV files. "Train.csv" contains the training data used to train our models, and "Test.csv" contains the test data for evaluation of our models. Within each CSV file, there are 2 columns. The 1st column contains the protein sequence, a string of variable length (between 50 to 3000 characters), made up of a combination of 20 unique characters (each character representing one amino acid). The 2nd column indicates whether the corresponding protein sequence is identified as a DNA Binding protein or not. The integer '1' is used to indicate a DNA binding protein, and '0' otherwise.

There is almost an equal number of positive and negative protein sequences in the training set. There are 14926 instances in the training set, where 7516 (50.3%) are DNA Binding proteins. The remaining 7410 (49.7%) are non-DNA Binding proteins. In comparison, the test set is not as balanced as the training set. The test set has a total of 57111 protein sequences, where 16378 (28.7%) are DNA binding proteins. The remaining 40733 (71.3%) are not DNA Binding proteins.

3 Methods

Feature Extraction

To convert the protein sequence into valid numerical inputs for our machine learning models, we decided to use k-mers as our features and their counts as our inputs. A k-mer is defined as a sequence of k amino acids, where k is an integer. For example, given a hypothetical sequence "ABCDEF", its 3-mers would be ["ABC", "BCD", "CDE", "DEF"]. Each k-mer becomes a new feature where the count of its occurrence in a sequence determines its value within that feature. To convert each sequence into its respective k-mers, we used the CountVectorizer package from the scikit-learn library. This is done for both the training and test set.

Hyperparameter Tuning

Each model's hyperparameters were tuned using GridSearchCV with MCC as the scoring metric.

Variance Threshold

To further distillate our features for more accurate feature selection, we decided to remove features if their variance was below a certain preset variance threshold value. To find the optimal variance threshold value, we iterated through different threshold values (0.1 to 2.0, with steps of 0.1). For each iteration, we trained, evaluated and saved all relevant metrics within an array. After all iterations, we plotted a graph of the metrics against threshold values to find the optimal variance threshold value.

The metrics we used were:

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

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

Machine Learning

K-Nearest Neighbours (kNN)

kNN is a supervised machine learning algorithm which uses proximity to perform classification. The algorithm takes in testing data, and determines its classification (either positive or negative) of the testing data. The class of the unclassified point is then determined by a "majority vote" of the k closest neighbours and their respective classes. Different values of k from the range of 50 to 200, with steps of 10, were used to fit the kNN model.

Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for binary classification. The algorithm aims to model the relationship between the input features and the probability of the outcome using the sigmoid function.

$$\text{Sigmoid Function: } \sigma(X) = \frac{1}{1 + e^{-X}}$$

The sigmoid function maps any real-valued number into a value between 0 and 1, which can be interpreted as a probability. If the predicted probability is more than 0.5, the model predicts the sample to be class 1. Otherwise, it predicts class 0.

Extreme Gradient Boosting (XGBoost)

XGBoost is a machine learning algorithm that utilises the boosting method. The algorithm works by building a series of decision trees sequentially. Each new tree added to the ensemble is trained on the errors of the current ensemble, with a weighted update that prioritises correcting the mistakes made by previous trees. This is known as a gradient boosting technique, an iterative process that gradually reduces prediction errors.

Deep Learning

Deep learning is a type of machine learning that uses artificial neural networks with multiple layers to gradually pick up more complex patterns from data.

Feature Extraction for Deep Learning

In contrast to other methods, we use character tokenization for the protein sequences followed by one-hot conversion. This means each letter or amino acid is turned into a list of numbers. For example, "ABA" becomes [1, 2, 1]. To keep our data consistent, we limit the input size to 1500 characters, considering both time complexity and 98% of the data is of length less than 1500. Sequences longer than 1500 characters are shortened. After conversion, we use one-hot encoding to ensure the model isn't affected by different encodings.

Model Architecture

Our deep learning model, depicted in Figure 1, includes an embedding layer, a 1D convolutional layer (with 256 filters, a kernel size of 15, a stride of 10, and ReLU activation), Bidirectional LSTM layers (8 units and a dropout rate of 0.5), and a fully connected layer with ReLU activation. The 1D convolutional layer reduces the complexity of the data from a 1500 x 256 tensor to a 149 x 256 tensor. This layer serves as the main feature extractor, similar to how image features are extracted using 2D convolutional layers. We choose 256 filters to capture more features from the sequence. While standard LSTM layers are often used for sequence data, our Bi-LSTM layer considers both past and future inputs, leading to improved performance for our specific task. The subsequent dense layer is then used for classification (Guo et al., 2019).

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 149, 256)	100,096
bidirectional_1 (Bidirectional)	(None, 16)	16,960
dropout_2 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 64)	1,088
dropout_3 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Figure 1: Architecture for Deep Learning Model

4 Results and Discussions

After performing Hyperparameter tuning with GridSearchCV, the resulting Confusion Matrices for each of the ML Models are shown. Additionally, the graphs for the respective scores through iterating different VarianceThreshold values are shown below.

k-Nearest Neighbours (kNN)

After GridSearchCV, the best parameters were found to be {'n_neighbours' = 50, 'weights' = 'distance', 'metric' = 'euclidean', 'p' = 1}.

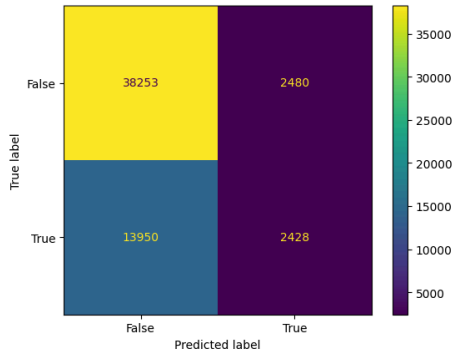

Figure 2: Confusion Matrix of Tuned kNN Model

Best threshold for variance was found to be 0.2

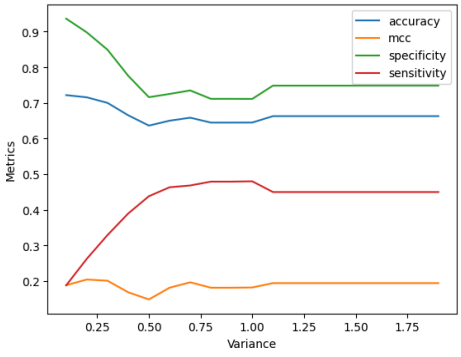

Figure 3: Performance of k-Nearest Neighbors model with different VarianceThreshold values

Logistic Regression

After GridSearchCV, the best parameters were found to be {'C': 0.01, 'max_iter': 100, 'penalty': 'l2', 'solver': 'saga'}.

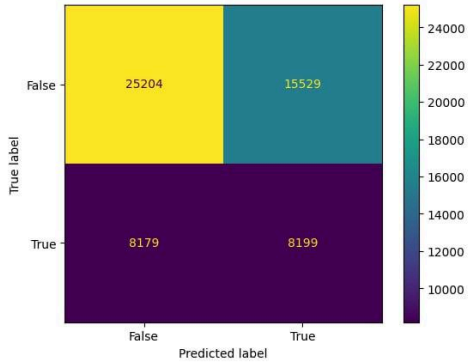

Figure 4: Confusion Matrix of Tuned Logistic Regression Model

Best threshold for variance was found to be 0.8

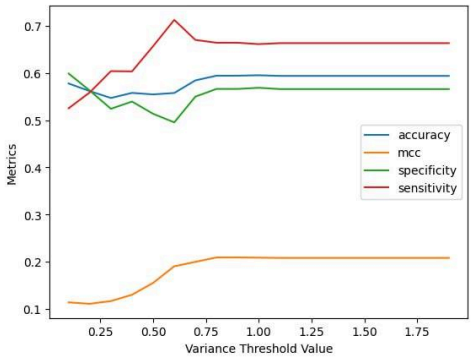

Figure 5: Performance of Logistic Regression Model with different VarianceThreshold values

Extreme Gradient Boosting (XGBoost)

After GridSearchCV, the best parameters were found to be {'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 100, 'colsample_bytree': 0.8, 'subsample': 1, 'gamma': 0}.

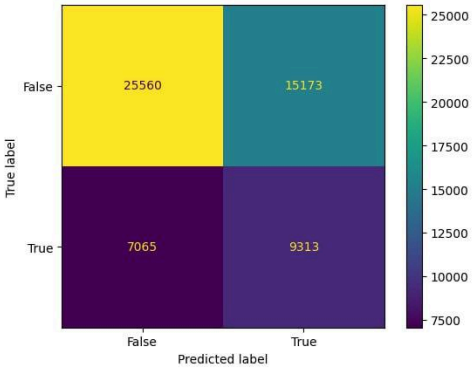

Figure 6: Confusion Matrix of Tuned XGBoost Model

Best threshold for variance was found to be 0.6

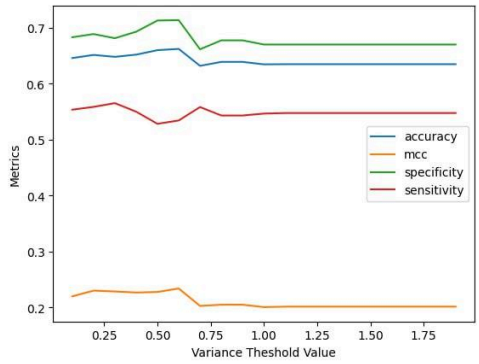

Figure 7: Performance of XGBoost Model with different VarianceThreshold values

Deep Learning Model

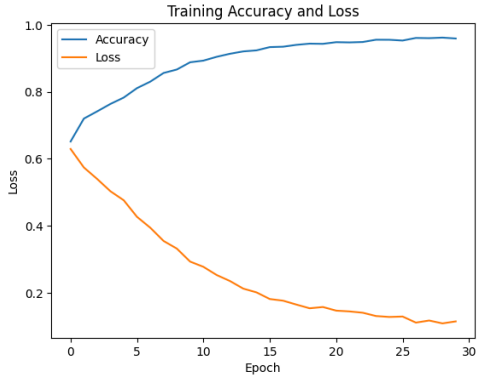

Figure 8: Training Accuracy and Loss of DL model over 30 epoch

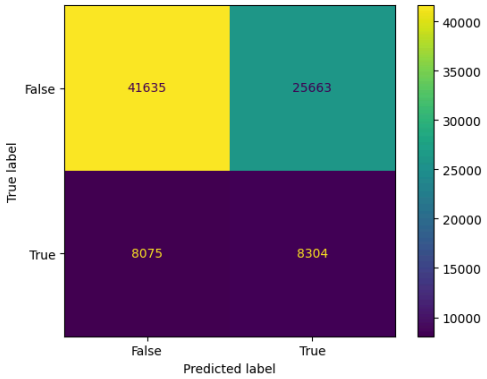

Figure 9: Confusion matrix of DL Model

	kNN	Logistic Regression	XGBoost	Deep Learning
Accuracy	0.71	0.59	0.66	0.60
Specificity	0.90	0.57	0.53	0.62
Sensitivity	0.26	0.66	0.71	0.51
MCC	0.20	0.21	0.23	0.10

Table 1: Results of Tuned ML Models and DL Model

Comparison of performance across different models

Our analysis has shown that in terms of accuracy and specificity, kNN has the best performance, while in terms of sensitivity and MCC, XGBoost has the best performance. Notably, the MCC score for deep learning was significantly lower than the other all three ML models.

Differences and weaknesses in features representation

In terms of feature selection, the choice of features for ML model is numerical, where each vector represents the count of a specific k-mer; whereas the features used in DL are categorical, where each k-mer is represented by its respective one-hot encoding. One disadvantage of using the count of k-mers is the loss of information as the size of the protein grows. For longer proteins, the weightage of the subsequence that classifies the protein becomes lower, yielding data with lower quality. However, the categorical representation of the sequences using the one-hot encoding of k-mers yields data with dimensions that increase exponentially with k, which is not feasible to train the data for $k > 1$ on our local machine. Thus, we have fixed $k = 1$ for the DL model.

Due to the nature of our data set, significant noise arises from the long protein sequences, which may result in lower MCC scores. Furthermore, as a result of the high complexity of DL Models, there is a high chance of overfitting due to the multiple hidden layers, DL models often require large amounts of data to generalise effectively (Yu et al., 2022). If the dataset available for training lacks diversity, DL models may struggle to capture the underlying patterns adequately, leading to lower MCC scores compared to the other ML model. DL models have more hyperparameters to tune compared to simpler models. Due to time limitations, it is more difficult to explore the search space adequately.

With reference to MCC as our scoring metric, XGBoost yields the best performance when compared to kNN, logistic regression, and our DL model. However, more research is needed in developing new machine learning and deep learning models with our current knowledge of protein sequences.

References

Ali, F., Ahmad, S., Swati, Z. N. K., & Akbar, S. (2019). DP-BINDER: machine learning model for prediction of DNA-binding proteins by fusing evolutionary and physicochemical information. Journal of Computer-Aided Molecular Design, 33(7), 645–658. https://doi.org/10.1007/s10822-019-00207-x

Bhandari, N., Khare, S., Walambe, R., & Kotecha, K. (2021). Comparison of machine learning and deep learning techniques in promoter prediction across diverse species. *PeerJ. Computer science*, *7*, e365. https://doi.org/10.7717/peerj-cs.365

Bhardwaj, N., Langlois, R., Zhao, G., & Lu, H. (2005). Kernel-based machine learning protocol for predicting DNA-binding proteins. Nucleic Acids Research, 33(20), 6486–6493. https://doi.org/10.1093/nar/gki949

Guo, L., Wang, S., Li, M., & Cao, Z. (2019). Accurate classification of membrane protein types based on sequence and evolutionary information using deep learning. BMC Bioinformatics, 20(Suppl 25), 700. https://doi.org/10.1186/s12859-019-3275-6

Hudson, W. H., & Ortlund, E. A. (2014). The structure, function and evolution of proteins that bind DNA and RNA. *Nature reviews. Molecular cell biology*, *15*(11), 749–760. https://doi.org/10.1038/nrm3884

Veltri, D., Kamath, U., & Shehu, A. (2018). Deep learning improves antimicrobial peptide recognition. Bioinformatics (Oxford, England), 34(16), 2740–2747. https://doi.org/10.1093/bioinformatics/bty179

Wang, W., Sun, L., Zhang, S., Zhang, H., Shi, J., Xu, T., & Li, K. (2017). Analysis and prediction of single-stranded and double-stranded DNA binding proteins based on protein sequences. *BMC bioinformatics*, *18*(1), 300. https://doi.org/10.1186/s12859-017-1715-8

Wang, W., Zhang, Y., Liu, D., Zhang, H., Wang, X., & Zhou, Y. (2022). Prediction of DNA-Binding Protein-Drug-Binding Sites Using Residue Interaction Networks and Sequence Feature. *Frontiers in bioengineering and biotechnology*, *10*, 822392. https://doi.org/10.3389/fbioe.2022.822392

Yu, S., Peng, D., Zhu, W., Liao, B., Wang, P., Yang, D., & Wu, F. (2022). Hybrid_DBP: Prediction of DNA-binding proteins using hybrid features and convolutional neural networks. Frontiers in pharmacology, 13, 1031759. https://doi.org/10.3389/fphar.2022.1031759