DNA-Binding Protein Classification: Comparison between various Traditional MLs vs Deep Learning Models
Group Number: 29

Goh Zhixuan Jeremiah, Christina Zhou, Nguyen Huy Tung, Nguyen My Binh An

e1121522@u.nus.edu, e1375985@u.nus.edu, e1129963@u.nus.edu, e1156700@u.nus.edu

Introduction

DNA-binding proteins (DBPs) are crucial for the regulation of cellular functions, making their accurate identification important for genome studies (Si et al. 2011). Yet traditional lab methods for such a task are costly, time-consuming, and worst of all inaccurate (Fu et al. 2018). If we are able to harness Machine Learning (ML) to perform this task, we could expect great savings in cost and increases in accuracy.

Literature Review

Various ML models have been used for DBP classification, using features such as structural, functional, evolutionary, and physicochemical properties from protein sequences with some encoding techniques (Sun et al. 2024; Jia, Huang, and Zhang 2021; Song et al. 2014). Classifiers such as Support Vector Machines (SVMs) are observed to perform with high accuracy alongside models such as Random Forests (RFs), XGBoost and Artificial Neural Networks (ANN). However, DBP classification remains challenging, where

- DBPBoost (Sun et al. 2024) has high computational costs due to the complexity of the improved differential evolution algorithm.
- KK-DBP (Jia, Huang, and Zhang 2021) is susceptible to overfitting with multiple features using Random Forest classifier.
- nDNA-prot (Song et al. 2014) struggles with handling class imbalance.

Our Aim

Building on previous research, we aim to identify the most reliable model for DBP classification by selecting and comparing key feature extraction methods—namely global and local amino acid distribution & composition, physicochemical properties, and Natural Language Processing techniques like embeddings and integer encoding. Our goal is to evaluate these features and classifiers (both traditional and deep learning) - to determine the best combination of features and model for accurate, reusable, and reliable DBP identification.

Methodology

Training and testing is done on the datasets `Train.fasta` and `Test.fasta` which include training and testing data respectively. Various classifiers and feature extraction methods based on natural language processing and biological scientific knowledge have been employed (Figure 1).

Exploratory Data Analysis & Data Pre-processing

There are around $53,000$ and $17,000$ protein sequences in the training and testing datasets respectively. Ratio between class 1 (positive binding) to class 0 (negative binding) is approximately $2 : 1$ in both training and testing datasets. To deal with class imbalance, Synthetic Minority Over-sampling Technique (SMOTE) is used to generate synthetic samples for the minority class by interpolating between existing minority instances. SMOTE is applied for our traditional MLs (TF-IDF, physicochemical properties) while class weight is applied for Deep Learning to correct class imbalance.

Moreover, while hundreds of amino acids exist, only 20 of them are commonly found in proteins. Thus, dataset is filtered to retain protein sequences that contain the unknown amino acid `X` or the rare amino acid `U`. Glycine, `G` is used to replace these uncommon and unknown amino acids because glycine has negligible impact on the physicochemical properties of the protein (Ahern, Rajagopal, and Tan 2023).

Besides, in NLP feature extractions, amino acids are treated as character-level tokens. As more than 95% of protein sequences has length shorter than 1200 characters (Figure A.6), a maximum length of 1200 is applied during embedding and tokenization methods. Sequences shorter than 1200 will be padded while those longer than 1200 will be truncated.

Feature Extraction & Selection

Statistical amino distribution & composition and scientific physicochemical properties features are used for traditional ML. Sequence-based character-level features in protein extracted from embedding and amino-acid integer encoding are used for Deep Learning models.

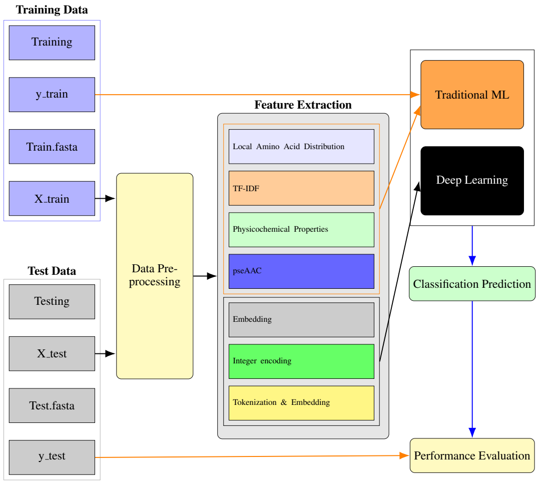


Figure 1: Summary of Feature Extraction Methods and our evaluation process pipeline.

Amino Acid Distribution Local frequency of each of common 20 amino acids is studied here using this formula:

$$f(a,p) = \frac{N_a}{N_p}$$

where N_a is the number of amino acid a in a protein sequence p of length N_p. Overall, we obtain a matrix of size $N \times 20$ with N being the number of sequences.

Logistic Regression is used to study the significance of mean local distribution of 20 amino acids in classifying the binding property of protein via the p-value. However, all p-values of 20 amino acids distribution are higher than 0.3, which is much higher than the usual threshold of 0.05 (Figure A.2), suggesting that local distribution of 20 amino acids are insignificant in predicting the binding response. Therefore, a comparative bar graph is plot to compare the mean local distribution of 20 amino acids between those of class 0 and those of class 1. (Figure A.1). Amino acids with a mean distribution difference greater than 0.03 between the classes were retained in the model due to their stronger association with binding properties compared to the rest.

Term Frequency-Inverse Document Frequency (TF-IDF) TF-IDF is used to capture the distribution of amino acids within sequences and their uniqueness across the dataset (Tasnim et al, 2024). The formula is:

$$\text{TF-IDF}(c,p) = \text{TD}(c,p) \times \log\left(\frac{N}{n_c}\right)$$

where $\text{TD}(c,d)$ is the frequency of amino acid c in a protein sequence q, N is total number of protein sequences in the training dataset, and n_c is the total number of sequences containing amino acid c.

Physicochemical Properties The main physicochemical properties that affect binding include hydropathy and net charge. Additionally, we examined other properties that may affect binding like instability index, average flexibility, isoelectricity, and molecular weight (Gromiha and Nagarajan 2013). The built-in Biopython library is used to retrieve physicochemical properties information (Figure A.4). We add physicochemical feature columns and save the datasets as `DNA_Train.csv` and `DNA_Test.csv` to avoid recalculating these features.

Pseudo-Amino Acid Composition (pseAAC) Unlike traditional amino acid composition, which only counts the frequency of each amino acid, pseAAC incorporates additional factors such as sequence order, physicochemical properties, and correlations between amino acids, which normal AAD does not use (Chou 2001). The builtin propy library is used to generate the feature vectors (Figure A.5).

Embedding Character-level embedding converts protein sequences into numerical vectors. We used `tokenize()` to map each of the 21 amino acids (20 common, 1 unknown 'X') to a unique integer. Padding and truncation is applied on each sequence to keep to a uniform length of 1200.

Amino Acid Integer Encoding A unique integer from 1 to 20 is assigned to each of 20 common amino acids. For rare `U` and unknown amino acids `X`, they will be mapped to 0. Zero padding and truncation is applied to standardize sequence length of 1200.

Machine Learning & Deep Learning Classifiers

Traditional MLs classifiers selected for DBP classification tasks include: Logistic Regression (LR), Naïve Bayes (NB), K Nearest Neighbors (KNN), Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM) because they are usually used for binary classification tasks. Notably, KNN is fine-tuned to find the best k in a range of $1 - 250$ that yields best MCC. RF and DT are also limited to a max depth of 10 to avoid overfitting.

Convolution Neural Network (CNN) which include a simple CNN and more sophisticated architecture ProtCNN which is designed for protein sequence classification are used as classifiers for Deep Learning models. Large Language Model using pre-trained ProtBert, which is more specialised for protein sequences are also employed.

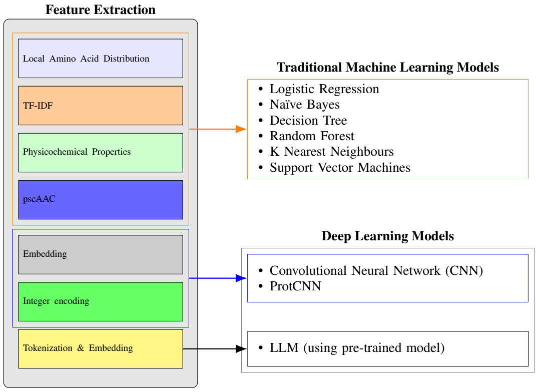


Figure 2: Summary of Features & Classifiers to be used

Logistic Regression (LR) is our baseline for it being commonly used for Binary Classification tasks.

Evaluation Metrics

Accuracy, Sensitivity, Specificity, AUC/ROC, and MCC are used as evaluation metrics. Due to class imbalance, MCC is more desirable to ensure classification is reliable. While accuracy provides a general measure of a model's performance, in cases of class imbalance, it can be misleading. Thus, Sensitivity (true positive rate) and Specificity (true negative rate) are more critical for our classification task.

Results

Traditional MLs with Amino Acid Distribution & Composition Features

Table 1 shows the results of using traditional MLs with AAD features. RF on both the full and selected AAD, and SVM for the full AAD, produced the highest MCC (0.43) with relatively high sensitivity, specificity, and AUC. Choosing only selected AAD causes greatest change in performance when LR and SVM is used, evident by a huge jump in sensitivity and specificity. Overall, traditional MLs with amino acid distribution as features do not have strong classification ability for DBP.

Traditional MLs with Physicochemical Features

Table 1 shows the results of using traditional MLs with physicochemical properties. MCC is consistently low for all models, suggesting poor predictive ability for DBP in such an imbalanced training dataset. Nevertheless, SVM performs best when physicochemical properties are used as features due to the highest MCC and AUC. In contrast, NB performs the worst. Thus, coupled with the fact of high p-value, physicochemical properties alone are not effective in affecting protein-DNA interactions and binding.

Traditional MLs with pseAAC

Table 1 shows the results of using traditional MLs with AAD features. Overall performance of all models using pseAAC is average. Yet, RF stood out, achieving the highest accuracy of 0.73 and a specificity of 0.82, but low sensitivity of 0.54. In contrast, Logistic Regression performed the worst across all metrics, particularly in sensitivity. Another notable drawback was the long computational time of ~ 1 day.

CNN Models with Encoding Techniques

Table 3 shows the results of using Deep Learning (CNN) and NLP feature extraction techniques. Embedding is a better feature extraction technique with $2-6\%$ higher in accuracy, sensitivity, and specificity, with a 5% higher in MCC. In addition, ProtCNN performs better with higher score in all metrics than normal CNN. Overall, Integer Encoding and ProtCNN together performs the best.

LLM ProtBert with Encoding

Table 3 shows the results of using a pretrained Large Language Model (LLM) of ProtBert and NLP feature extraction techniques, similar to that of CNN. However, our results were not ideal and extensive tuning is required to ensure it produces workable results.

Discussion

Among all traditional MLs using biological features, LR consistently performs worst in all aspects except when using physicochemical properties. In contrast, RF is more recommended to be used with AAD due to the comparatively higher sensitivity, accuracy, MCC, and specificity.

Deep learning algorithms, including CNN and ProtCNN, show consistently higher performance metrics compared to traditional models, with remarkably high for MCC.

From a biological understanding, features like AAD and physicochemical properties has a good interpretability to predict DBP. However, our results show they are less effective in computational sense. In long protein sequences, DNA-binding sites constitute only a small portion of the overall structure (Sigrist et al. 2009). When analyzing the overall amino acid frequency, the significance of the binding sites, which contains sequential order of amino acid-diminishes (Alberts et al. 2002).

NLP techniques that numerically encode amino acids, while less biologically interpretable, often outperform biologically driven features in predictive accuracy. This may be due to tokenization, truncation, and embedding processes that retain sequence-specific information in a vector. Since amino acid order and sequence endings are crucial for DBP-sites (Ahern, Rajagopal, and Tan 2023), NLP's capacity to preserve this order likely contributes to high performance, particularly in models like ProtCNN, which is specifically adapted for protein data. However, while DL models, especially ProtCNN, perform well, it may still require extensive fine-tuning to fully capture DBP-specific nuances than traditional MLs. Similarly, LLM-ProtBert also requires extensive tuning to achieve its full potential. Thus, the concern of high computational load may offset the strong predictive ability when applied in real-life settings, particularly with large datasets like ours, where fine-tuning is essential for achieving top performance in DL framework.

We recommend using ProtCNN with embedding for feature extraction, as it delivers reliable results even with large, imbalanced datasets. However, when computational resources are limited, RFs with AAD offer a faster alternative, though this approach should be applied carefully.

To further evaluate the model's performance, we could have done cross validation. However, since our data is already split into a training and testing portion, taking a portion of training data to train the model then apply this on the test model is not ideal. Along with the high computational demands of this task, we were not able to perform cross validation for our dataset.

Impact of DBP classification by ML models

ML models, especially DL, are transforming DBP classification by providing efficient alternatives to human expertise. Although human experts can identify binding sites with high accuracy, their approach is time-consuming and less scalable (Alipanahi et al. 2015). Therefore, proper implementation of a ML model has the tremendous benefit of saving human time and effort, along with cost savings.

Feature	LR		NB		KNN		SVM		DT		RF	
Full AAD	Sensitivity	0.64	Sensitivity	0.74	Sensitivity	0.92	Sensitivity	0.74	Sensitivity	0.75	Sensitivity	0.74
	Specificity	0.64	Specificity	0.60	Specificity	0.42	Specificity	0.71	Specificity	0.66	Specificity	0.71
	Accuracy	0.64	Accuracy	0.65	Accuracy	0.59	Accuracy	0.72	Accuracy	0.69	Accuracy	0.72
	MCC	0.27	MCC	0.32	MCC	0.34	MCC	0.43	MCC	0.39	MCC	0.43
	AUC	0.70	AUC	0.74	AUC	0.80	AUC	0.80	AUC	0.78	AUC	0.81
Selected AAD	Sensitivity	0.18	Sensitivity	0.69	Sensitivity	0.58	Sensitivity	0.34	Sensitivity	0.79	Sensitivity	0.79
	Specificity	0.92	Specificity	0.66	Specificity	0.78	Specificity	0.92	Specificity	0.62	Specificity	0.67
	Accuracy	0.67	Accuracy	0.67	Accuracy	0.72	Accuracy	0.72	Accuracy	0.68	Accuracy	0.71
	MCC	0.16	MCC	0.33	MCC	0.36	MCC	0.32	MCC	0.39	MCC	0.43
	AUC	0.69	AUC	0.74	AUC	0.76	AUC	0.76	AUC	0.78	AUC	0.81
TF-IDF	Sensitivity	0.73	Sensitivity	0.63	Sensitivity	0.91	Sensitivity	0.70	Sensitivity	0.55	Sensitivity	0.57
	Specificity	0.63	Specificity	0.62	Specificity	0.37	Specificity	0.72	Specificity	0.68	Specificity	0.80
	Accuracy	0.66	Accuracy	0.62	Accuracy	0.55	Accuracy	0.72	Accuracy	0.64	Accuracy	0.72
	MCC	0.34	MCC	0.24	MCC	0.30	MCC	0.41	MCC	0.22	MCC	0.37
	AUC	0.75	AUC	0.68	AUC	0.76	AUC	0.78	AUC	0.61	AUC	0.77
pseAAC	Sensitivity	0.68	Sensitivity	0.76	Sensitivity	0.86	Sensitivity	0.68	Sensitivity	0.54	Sensitivity	0.54
	Specificity	0.62	Specificity	0.62	Specificity	0.41	Specificity	0.74	Specificity	0.67	Specificity	0.82
	Accuracy	0.64	Accuracy	0.64	Accuracy	0.56	Accuracy	0.72	Accuracy	0.63	Accuracy	0.73
	MCC	0.28	MCC	0.30	MCC	0.27	MCC	0.4	MCC	0.20	MCC	0.37
	AUC	0.70	AUC	0.73	AUC	0.71	AUC	0.78	AUC	0.60	AUC	0.76

Table 1: Comparison of performance metrics between various traditional MLs using AAD features.

Logistic Regression		Naïve Bayes		KNN		SVM		Decision Tree		Random Forest	
Sensitivity	0.62	Sensitivity	0.44	Sensitivity	0.79	Sensitivity	0.74	Sensitivity	0.69	Sensitivity	0.61
Specificity	0.72	Specificity	0.83	Specificity	0.56	Specificity	0.63	Specificity	0.66	Specificity	0.74
Accuracy	0.69	Accuracy	0.70	Accuracy	0.64	Accuracy	0.67	Accuracy	0.67	Accuracy	0.70
MCC	0.33	MCC	0.29	MCC	0.33	MCC	0.35	MCC	0.33	MCC	0.34
AUC	0.73	AUC	0.69	AUC	0.75	AUC	0.75	AUC	0.74	AUC	0.74

Table 2: Comparison of performance metrics between various traditional MLs using physicochemical properties.

Encoding Technique	CNN		ProtCNN		ProtBert	
Embedding	SEN	0.75	SEN	0.78	SEN	0
	SPE	0.80	SPE	0.82	SPE	1
	ACC	0.85	ACC	0.87	ACC	0.67
	MCC	0.60	MCC	0.63	MCC	0
	AUC	0.83	AUC	0.86	AUC	0.50
Integer Encoding	SEN	0.70	SEN	0.72	–	
	SPE	0.78	SPE	0.79		
	ACC	0.82	ACC	0.83		
	MCC	0.55	MCC	0.58		
	AUC	0.81	AUC	0.84		

Table 3: Comparison of Performance between CNN, ProtCNN, and ProtBert using embedding and integer encoding techniques.

However, the adoption of ML for DBP classification presents several challenges:

- **Privacy**: Utilizing pre-trained databases for DBP classification risks genomic data leakage (Gymrek et al. 2013).
- **Fairness**: Models can produce biased results when applied to diverse datasets, leading to skewed outcomes (Mehrabi et al. 2021).
- **Interpretability**: Interpretable models allow researchers to uncover and validate biological mechanisms. Conversely, favoring opaque models can obscure the understanding of factors driving binding (Molnar 2023).
- **Job Impact**: Automation enables researchers to concentrate on more critical tasks, potentially shifting job requirements toward managing and validating advanced ML systems in bioinformatics (Autor 2015).

Conclusion

In summary, ProtCNN with embedding is optimal for DBP classification with sufficient computational resources. For faster computation with less complexity, AAD with RF is suitable. However, our findings reveal a disconnect between biological features and binding ability in DBP classification. To balance interpretability, reliability, and computational efficiency, it’s essential to combine machine learning with laboratory validation.

Appendix

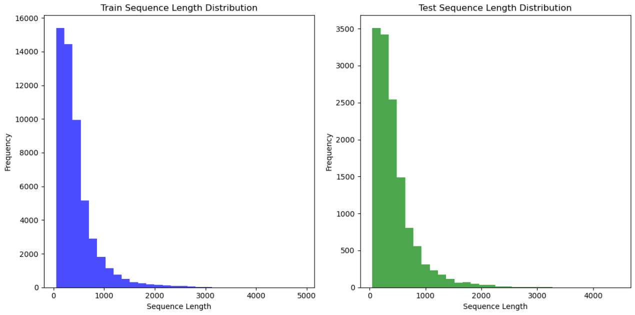
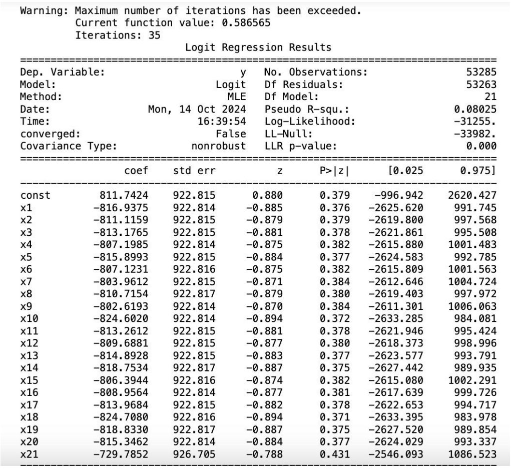
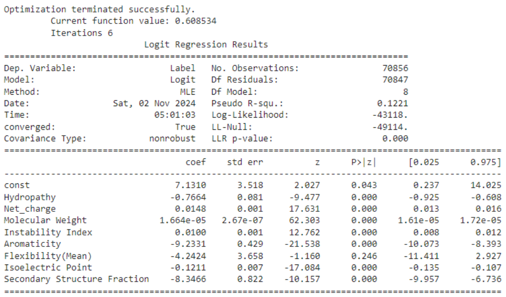


Figure A.1: Sequence Length Distribution of Protein sequences in Train (blue) and Test (green) datasets.



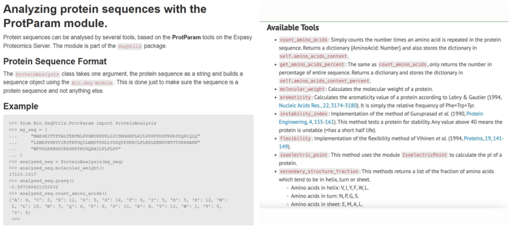
```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.586565
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                53285
Model:                          Logit   Df Residuals:                    53263
Method:                           MLE   Df Model:                           21
Date:                Mon, 14 Oct 2024   Pseudo R-squ.:                 0.08025
Time:                        16:39:54   Log-Likelihood:                -31255.
converged:                      False   LL-Null:                       -33982.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        811.7424    922.815      0.880      0.379    -996.942    2620.427
x1          -816.9375    922.814     -0.885      0.376   -2625.620     991.745
x2          -811.1159    922.815     -0.879      0.379   -2619.800     997.568
x3          -813.1765    922.815     -0.881      0.378   -2621.861     995.508
x4          -807.1985    922.814     -0.875      0.382   -2615.880    1001.483
x5          -815.8993    922.815     -0.884      0.377   -2624.583     992.785
x6          -807.1231    922.816     -0.875      0.382   -2615.809    1001.563
x7          -803.9612    922.815     -0.871      0.384   -2612.646    1004.724
x8          -810.7154    922.817     -0.879      0.380   -2619.403     997.972
x9          -802.6193    922.814     -0.870      0.384   -2611.301    1006.063
x10         -824.6020    922.814     -0.894      0.372   -2633.285     984.081
x11         -813.2612    922.815     -0.881      0.378   -2621.946     995.424
x12         -809.6881    922.815     -0.877      0.380   -2618.373     998.996
x13         -814.8928    922.815     -0.883      0.377   -2623.577     993.791
x14         -818.7534    922.817     -0.887      0.375   -2627.442     989.935
x15         -806.3944    922.816     -0.874      0.382   -2615.080    1002.291
x16         -808.9564    922.814     -0.877      0.381   -2617.639     999.726
x17         -813.9684    922.815     -0.882      0.378   -2622.653     994.717
x18         -824.7080    922.816     -0.894      0.371   -2633.395     983.978
x19         -818.8330    922.817     -0.887      0.375   -2627.520     989.854
x20         -815.3462    922.814     -0.884      0.377   -2624.029     993.337
x21         -729.7852    926.705     -0.788      0.431   -2546.093    1086.523
------------------------------------------------------------------------------
```

Figure A.2: p-values obtained from Logistic Regression using Amino Acid Distribution.



```
Optimization terminated successfully.
         Current function value: 0.608534
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Label   No. Observations:                70856
Model:                          Logit   Df Residuals:                    70847
Method:                           MLE   Df Model:                            8
Date:                Sat, 02 Nov 2024   Pseudo R-squ.:                  0.1221
Time:                        05:01:03   Log-Likelihood:                -43118.
converged:                       True   LL-Null:                       -49114.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                          7.1310      3.518      2.027      0.043       0.237      14.025
Hydropathy                    -0.7664      0.081     -9.477      0.000      -0.925      -0.608
Net_charge                     0.0148      0.001     17.631      0.000       0.013       0.016
Molecular Weight            1.664e-05   2.67e-07     62.303      0.000    1.61e-05    1.72e-05
Instability Index              0.0100      0.001     12.762      0.000       0.008       0.012
Aromaticity                   -9.2331      0.429    -21.538      0.000     -10.073      -8.393
Flexibility(Mean)             -4.2424      3.658     -1.160      0.246     -11.411       2.927
Isoelectric Point             -0.1211      0.007    -17.084      0.000      -0.135      -0.107
Secondary Structure Fraction  -8.3466      0.822    -10.157      0.000      -9.957      -6.736
==============================================================================================
```

Figure A.3: p-values obtained from Logistic Regression using Physicochemical Properties.



Analyzing protein sequences with the ProtParam module.

Protein sequences can be analysed by several tools, based on the **ProtParam** tools on the Expasy Proteomics Server. The module is part of the `SeqUtils` package.

Protein Sequence Format

The `ProteinAnalysis` class takes one argument, the protein sequence as a string and builds a sequence object using the `Bio.Seq module`. This is done just to make sure the sequence is a protein sequence and not anything else.

Example

```
>>> from Bio.SeqUtils.ProtParam import ProteinAnalysis
>>> my_seq = (
...     "MAEGEITTFTALTEKFNLPPGNYKKPKLLYCSNGGHFLRILPDGTVDGTRDRSDQHIQLQ"
...     "LSAESVGEVYIKSTETGQYLAMDTSGLLYGSQTPSEECLFLERLEENHYNTYTSKKHAKN"
...     "WFVGLKKNGSCKRGPRTHYGQKAILFLPLPV"
... )
>>> analysed_seq = ProteinAnalysis(my_seq)
>>> analysed_seq.molecular_weight()
17103.1617
>>> analysed_seq.gravy()
-0.597368421052632
>>> analysed_seq.count_amino_acids()
{'A': 6, 'C': 3, 'E': 12, 'D': 5, 'G': 14, 'F': 6, 'I': 5, 'H': 5, 'K': 12, 'M':
 2, 'L': 18, 'N': 7, 'Q': 6, 'P': 8, 'S': 10, 'R': 6, 'T': 13, 'W': 1, 'V': 5,
 'Y': 8}
 >>>
```

Available Tools

- `count_amino_acids`: Simply counts the number times an amino acid is repeated in the protein sequence. Returns a dictionary {AminoAcid: Number} and also stores the dictionary in `self.amino_acids_content`.
- `get_amino_acids_percent`: The same as `count_amino_acids`, only returns the number in percentage of entire sequence. Returns a dictionary and stores the dictionary in `self.amino_acids_content_percent`.
- `molecular_weight`: Calculates the molecular weight of a protein.
- `aromaticity`: Calculates the aromaticity value of a protein according to Lobry & Gautier (1994, Nucleic Acids Res., 22, 3174-3180). It is simply the relative frequency of Phe+Trp+Tyr.
- `instability_index`: Implementation of the method of Guruprasad *et al.* (1990, Protein Engineering, 4, 155-161). This method tests a protein for stability. Any value above 40 means the protein is unstable (=has a short half life).
- `flexibility`: Implementation of the flexibility method of Vihinen *et al.* (1994, Proteins, 19, 141-149).
- `isoelectric_point`: This method uses the module `IsoelectricPoint` to calculate the pI of a protein.
- `secondary_structure_fraction`: This methods returns a list of the fraction of amino acids which tend to be in helix, turn or sheet.
  - Amino acids in helix: V, I, Y, F, W, L.
  - Amino acids in turn: N, P, G, S.
  - Amino acids in sheet: E, M, A, L.

Figure A.4: Biopython package function which provides physicochemical property information.

References

Ahern, K.; Rajagopal, I.; and Tan, T. 2023. 202: Structure & Function - Amino Acids. Accessed: 2024-10-31.

Alberts, B.; Johnson, A.; Lewis, J.; Raff, M.; Roberts, K.; and Walter, P. 2002. Molecular Biology of the Cell. https://www.ncbi.nlm.nih.gov/books/NBK21054/. Chapter 8, Regulatory Proteins; Accessed: 2024-10-26.

Alipanahi, B.; Delong, A.; Weirauch, M. T.; and Frey, B. J. 2015. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8): 831–838. Accessed: 2024-10-21.

Autor, D. H. 2015. Why Are There Still So Many Jobs? The History and Future of Workplace Automation. *Journal of Economic Perspectives*, 29(3): 3–30. Accessed: 2024-10-24.

Chou, K. 2001. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, 43(3): 246–255. Accessed: 2024-10-30.

Fu, X.; Zhu, W.; Liao, B.; Cai, L.; Peng, L.; and Yang, J. 2018. Improved DNA-Binding Protein Identification by Incorporating Evolutionary Information Into the Chou's PseAAC. *IEEE Access*, 6: 66545–66556. Accessed: 2024-10-28.

Gromiha, M. M.; and Nagarajan, R. 2013. *Computational Approaches for Predicting the Binding Sites and Understanding the Recognition Mechanism of Protein–DNA Complexes*, 65–99. Elsevier. Accessed: 2024-10-25.

Gymrek, M.; McGuire, A. L.; Golan, D.; Halperin, E.; and Erlich, Y. 2013. Identifying Personal Genomes by Surname Inference. *Science*, 339(6117): 321–324. Accessed: 2024-10-22.

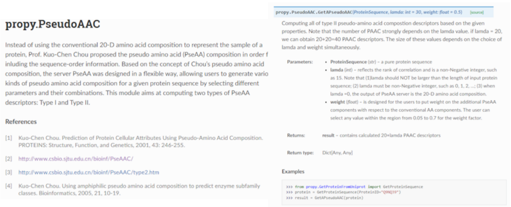
Jia, Y.; Huang, S.; and Zhang, T. 2021. KK-DBP: A Multi-Feature Fusion Method for DNA-Binding Protein Identification Based on Random Forest. *Frontiers in Genetics*, 12. Accessed: 2024-10-18.

Mehrabi, N.; Morstatter, F.; Saxena, N.; Lerman, K.; and Galstyan, A. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*, 54(6): 1–35. Accessed: 2024-10-23.

Molnar, C. 2023. The Importance of Interpretability. Accessed: 2024-11-02.

Si, J.; Zhang, Z.; Lin, B.; Schroeder, M.; and Huang, B. 2011. MetaDBSite: a meta approach to improve protein DNA-binding sites prediction. *BMC Systems Biology*, 5(S1). Accessed: 2024-10-29.

Sigrist, C. J. A.; Cerutti, L.; de Castro, E.; Langendijk-Genevaux, P. S.; Bulliard, V.; Bairoch, A.; and Hulo, N. 2009. PROSITE, a protein domain database for functional



propy.PseudoAAC

Instead of using the conventional 20-D amino acid composition to represent the sample of a protein, Prof. Kuo-Chen Chou proposed the pseudo amino acid (PseAA) composition in order f inluding the sequence-order information. Based on the concept of Chou's pseudo amino acid composition, the server PseAA was designed in a flexible way, allowing users to generate vario kinds of pseudo amino acid composition for a given protein sequence by selecting different parameters and their combinations. This module aims at computing two types of PseAA descriptors: Type I and Type II.

References

[1] Kuo-Chen Chou. Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. PROTEINS: Structure, Function, and Genetics, 2001, 43: 246-255.

[2] http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/

[3] http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm

[4] Kuo-Chen Chou. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. Bioinformatics, 2005, 21, 10-19.

propy.PseudoAAC.GetAPseudoAAC(*ProteinSequence, lamda: int = 30, weight: float = 0.5*) [source]

Computing all of type II pseudo-amino acid compostion descriptors based on the given properties. Note that the number of PAAC strongly depends on the lamda value. if lamda = 20, we can obtain 20+20=40 PAAC descriptors. The size of these values depends on the choice of lamda and weight simultaneously.

Parameters:
- **ProteinSequence** (*str*) – a pure protein sequence
- **lamda** (*int*) – reflects the rank of correlation and is a non-Negative integer, such as 15. Note that (1)lamda should NOT be larger than the length of input protein sequence; (2) lamda must be non-Negative integer, such as 0, 1, 2, ...; (3) when lamda =0, the output of PseAA server is the 20-D amino acid composition.
- **weight** (*float*) – is designed for the users to put weight on the additional PseAA components with respect to the conventional AA components. The user can select any value within the region from 0.05 to 0.7 for the weight factor.

Returns: **result** – contains calculated 20+lamda PAAC descriptors

Return type: Dict[Any, Any]

Examples

```
>>> from propy.GetProteinFromUniprot import GetProteinSequence
>>> protein = GetProteinSequence(ProteinID="Q9NQ39")
>>> result = GetAPseudoAAC(protein)
```

Figure A.5: propy package function which generates feature vectors for classificiation.

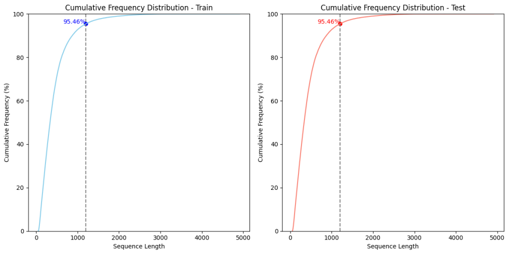


Figure A.6: Cumulative frequency distribution of sequence lengths. Note that 95% of all sequences have length ≤ 1200.

characterization and annotation. *Nucleic Acids Research*, 38(suppl_1): D161–D166. Accessed: 2024-10-27.

Song, L.; Li, D.; Zeng, X.; Wu, Y.; Guo, L.; and Zou, Q. 2014. nDNA-prot: identification of DNA-binding proteins based on unbalanced classification. *BMC Bioinformatics*, 15. Accessed: 2024-10-20.

Sun, A.; Li, H.; Dong, G.; Zhao, Y.; and Zhang, D. 2024. DBPboost: A method of classification of DNA-binding proteins based on improved differential evolution algorithm and feature extraction. *Methods*, 223: 56–64. Accessed: 2024-10-16.