

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

Predict protein-DNA binding affinity from their sequences

**Student: Lei Zhenghong
Supervisor: Dr. Li Yinghui**

SCHOOL OF BIOLOGICAL SCIENCES

Year 2024

Introduction

- Position weight matrices (PWMs) and position frequency matrices (PFMs) are statistical models used in bioinformatics to represent patterns in DNA, RNA, and protein sequences. They are often used to describe consensus sequence features of gene regulatory elements, such as transcription factor binding sites.
- PFMs: PFM records the frequency of occurrence of individual nucleotides at each position in a collection of sequences. In other words, a PFM might be used to describe the number of occurrences of the four nucleotides A, T, C, and G at each position in a set of transcription factor binding sites.
- PWMs: PWM is based on PFM and converts frequency information into a matrix of weight information through certain mathematical transformations. PWM provides a quantitative way to evaluate how well a given sequence matches a known pattern. Positions with higher weights are more important for pattern recognition. By calculating the weight value of each position on the sequence and summing it, the score of this sequence as an instance of the pattern can be obtained. The higher the score, the better the sequence matches the pattern.
- The problem we are trying to solve is to build a model that can predict protein-DNA binding affinity from their sequences by generating features that related to the sequences (One-Hot encoding, Word2Vec, Evolutionary Scale Modeling,), which are available from JASPAR.

Data

- The data are all collected from the JASPAR website (<https://jaspar.elixir.no>) satisfying Homo sapiens, collection is “CORE” and tax_group is “vertebrates” [1].

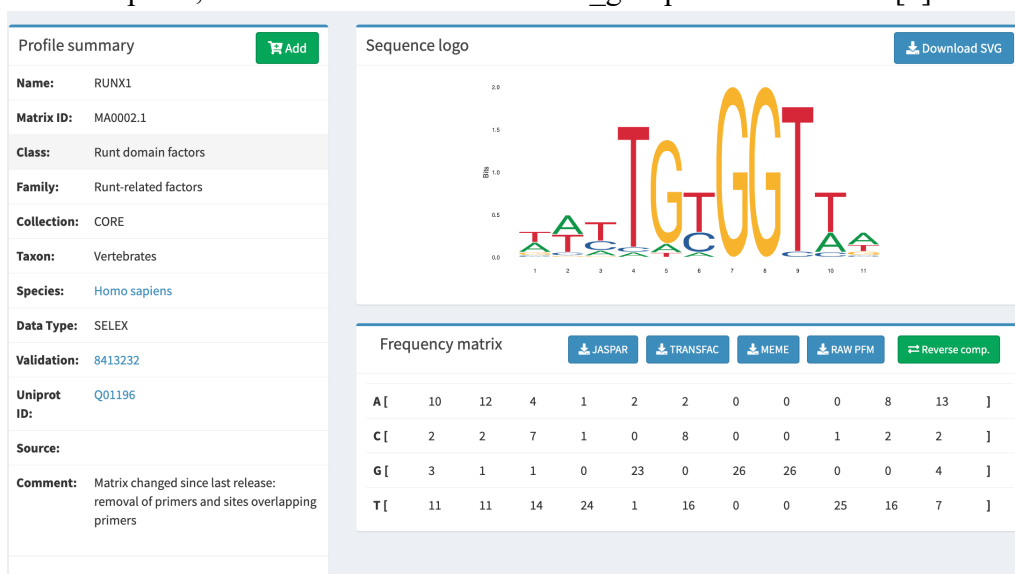


Figure 1: The example page of JASPAR [1]

- The collected data including matrix id, Name, Class, Family, Data type and uniport id. Meanwhile, DNA sequence and protein sequence are being downloaded and stored in the folder, and there are totally 330 DNA FASTA files and 681 protein FASTA files.

ID	Name	Class	Family	Data type	Uniport ID
0	MA0002.1	RUNX1	Runt domain factors	SELEX	Q01196
1	MA0003.2	TFAP2A	Basic helix-span-helix factors (bHSH)	AP-2	P05549
2	MA0003.4	TFAP2A	Basic helix-span-helix factors (bHSH)	AP-2	P05549
3	MA0007.2	AR	Nuclear receptors with C4 zinc fingers	ChIP-seq	P10275
4	MA0014.2	PAX5	Paired box factors	ChIP-seq	Q02548
...
325	MA1973.1	ZKSCAN3	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	Na
326	MA1974.1	ZNF211	C2H2 zinc finger factors	Factors with multiple dispersed zinc fingers	Na
327	MA1979.1	ZNF416	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	Na
328	MA1983.1	ZNF582	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	Na
329	MA1986.1	ZNF692	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	Na

Figure 2: The download data

```

>MA0002=RUNX1→1
gtAATTGTGGTTAatataaaaaactcgtga
>MA0002=RUNX1→2
gtgaaattactATCTGTGGTTAttttcgtc
>MA0002=RUNX1→3
ttaccattaatcgttcAATTGTGGTAAG
>MA0002=RUNX1→4
ctggtcttaataacTTCTGCGGTTAattta

>sp|A0AVK6|E2F8_HUMAN Transcription factor E2F8 OS=Homo sapiens OX=9606 GN=E2F8 PE=1 SV=1
MENKENLFCCEPHKRGMLKPTLKESTANIVLAEIPDFGLTTPTKPKESQGEPTPTANLKMILISAVSPEIRNRDQKRGFLFDRNSGLPEAKDCIHEHLSGDEF
EKSQPSRKEKSLGLLCHKFLARYPNYPAPAWNDDICLDEVAEELNVERRRIYDINVLLESLHMSRLAKNRYTWGRHNLNKTGLTKLSIGEEKNYAEQIMIKKK
EYEQDFDTKSYIEDHIIKSNTPGNGHPDPCFVELPGVEFRAASVNSRDKSLRVMSQKFMVLFVSTPQIVSLVAAKILLIGEDHVEDLDKSKFKTKIRRLYDI
ANWLSLDLIKVVHYTEERGRKPAFKWTGPEISPTSGSSPVYHFTPSDLEVRSSKENCANLKFSTRGKPNFTRHPSLILKVKIESDRRKINSAPSSPIKTNKA
ESSQNSAPFSPKMAQLAAICKMQLLEEQSSERQKVKVQLARSGPCKVPAPLDPVNAEMELTAPSLIQLPLGMVPLIPSLSSAVPLILPQAPSGPSYAIYLOPTQA
HQSVTPPGGLSPTVCTTHSSKATGSKDSTATTTEKAANDTSKASASTRPGSLLPAPERQGAQSRTRERAGERGSKRASMLEDGSGSKKFKEDLKLGNVSATLFP5
GYLIPLTQCSLGAESILSGKENSALSHPNRIYSSPIAGVLPVTSSELTAQVNFPSFHVTPKLHVSPTSVAAPVGNPSALASSHPVPPIQNPSSAIVNFTLQHLG
LISPNVQLSASPGSGIVPVSPRIESVNAPENAGTQGRATNYDSVPVGQSQPNQGSQVAVTGAQQPVPVTPKGSQLVAESFFRTPGGTPKTPSSSCMDFEAGANKTS
LGTFLFVQRKLEVSTEDVH

```

Figure 3: The example FASTA data of DNA sequence and protein sequence

Data Exploration

- The DNA data are assigned into three different groups based on the data type:
 - Selex data: SELEX, High-throughput SELEX SAGE, HT-SELEX
 - Chip data: ChIP-seq, ChIP-chip
 - Other data: COMPILED, bacterial 1-hybrid, NA
- Since the length of the uppercase sequences for different DNA is different, so we first do the data exploration toward the three groups of the DNA data in order to find out the best appropriate length for modeling. Below are the bar plots of DNA sequences length for all three group of data and the summary bar plot for all the DNA data.

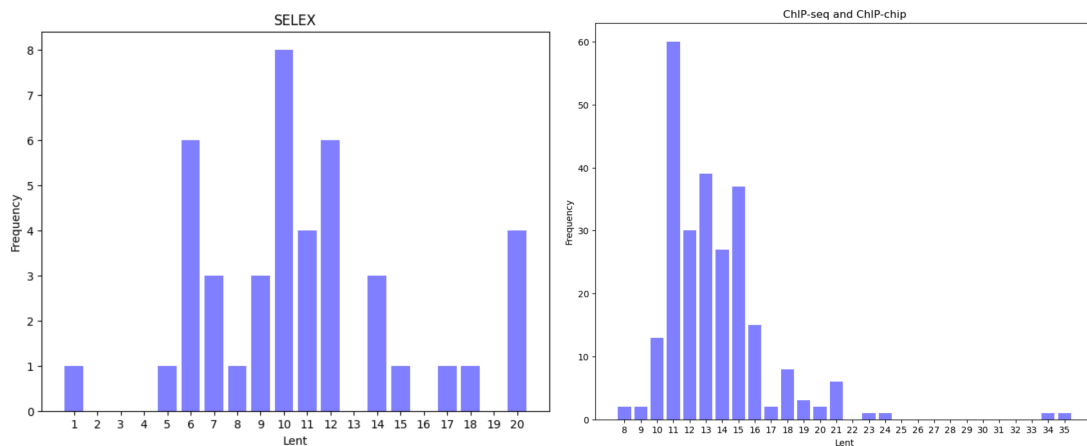


Figure 4 and 5: The bar plots of the DNA sequences length for SELEX data type and CHIP data type

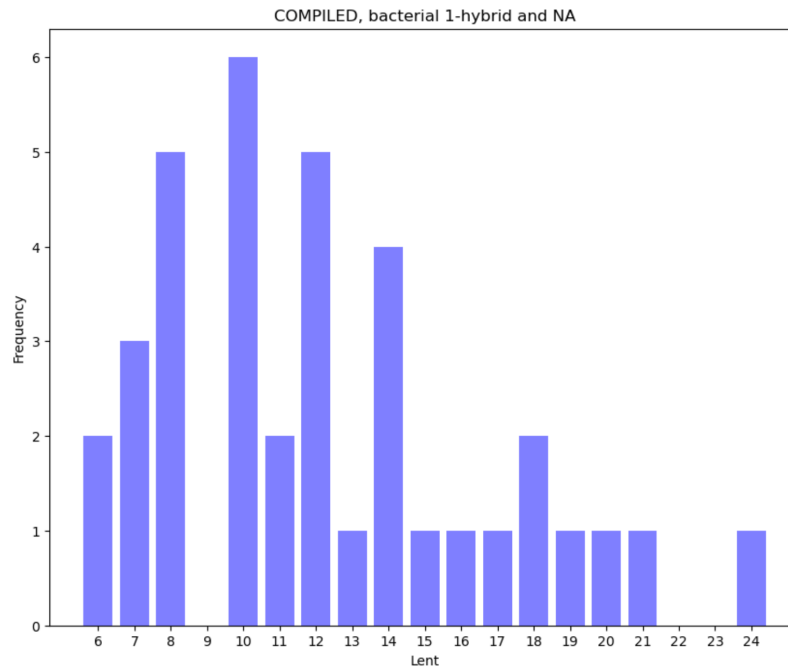


Figure 6: The bar plots of the DNA sequences length for other data type

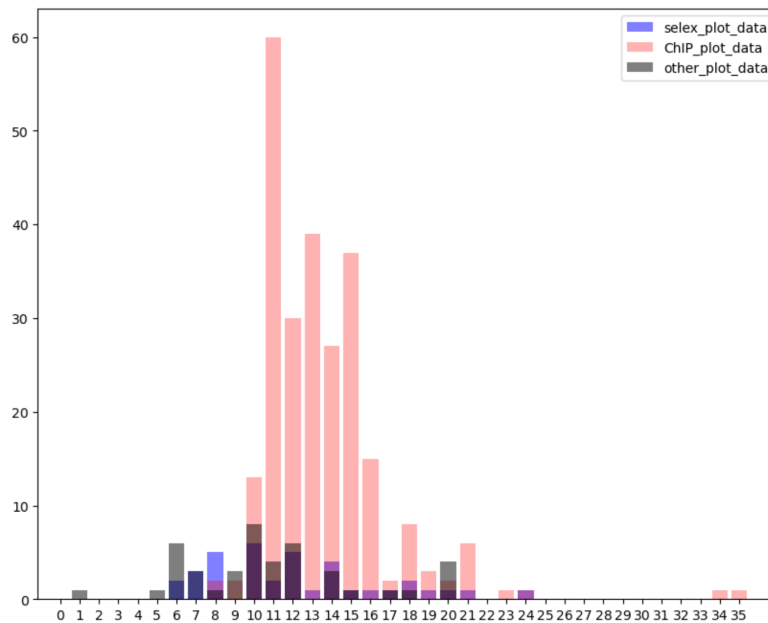


Figure 7: The bar plots of the DNA sequences length for all the data types

- From the plots we can see that the most appropriate length of the DNA sequence is 11. Therefore, in the later data pre-processing, the length for all the DNA sequence will be 11.

Data Pre-processing

- **Handling high similarity sequences:** Since there are many sequences that have high similarity with others. The sequences that have over 80% similarity with others will be removed and each sequence will be assigned a number that represents how many sequences used to have over 80% similarity with it.

```

>Cluster 0
0→15aa, >hg19_chr8:10345717-... *
1→15aa, >hg19_chr20:44144379... at 100.00%
2→15aa, >hg19_chr13:83635184... at 100.00%
3→15aa, >hg19_chr2:19570731-... at 100.00%
>Cluster 1
0→15aa, >hg19_chr10:12241300... *
1→15aa, >hg19_chr5:136960821... at 100.00%
2→15aa, >hg19_chr2:216953754... at 100.00%
>Cluster 2
0→15aa, >hg19_chr15:85874606... *
1→15aa, >hg19_chr8:6593335-6... at 100.00%
>Cluster 3
0→15aa, >hg19_chr12:62553254... *
>Cluster 4
0→15aa, >hg19_chr3:185376611... *
>Cluster 5
0→15aa, >hg19_chr20:46339929... *
>Cluster 6
0→15aa, >hg19_chr9:96897729-... *
1→15aa, >hg19_chr3:141131302... at 100.00%
2→15aa, >hg19_chr6:160860648... at 100.00%

```

Figure 8: The example figure of high similarity sequences

- **Remain uppercase and keep same length:** Since we have found that the best appropriate sequence length is 11, so that all the sequences have been modified into same uppercase sequence length by removing extra uppercase from both sides if over 11 or transforming lowercase into uppercase from both sides if lower 11. After that, all the lowercases in the sequences have been deleted.

>hg38_chr1:222548163-222548173(+)	>hg38_chr1:222548163-222548173(167)
tccccgccccct	TCCCCGCCCT
>hg38_chr19:34134524-34134534(+)	>hg38_chr19:34134524-34134534(52)
CCCCCGCCCAC	CCCCGCCAC
>hg38_chr2:219552722-219552732(-)	>hg38_chr2:219552722-219552732(642)
GCCCCGCCCG	GCCCCGCCCG
>hg38_chr5:148826447-148826457(+)	>hg38_chr5:148826447-148826457(111)
GCCCCGCCCG	GCCCCGCCCG
>hg38_chr14:96391983-96391993(+)	>hg38_chr14:96392023-96392033(33)
gccccgccccg	TCCCCGCCCG
>hg38_chr14:96392023-96392033(+)	>hg38_chr14:96392053-96392063(782)
tccccgccccg	GCCCCGCCCG

Figure 9: The example figure of remaining uppercase and keeping same length

Generating Negative Data

- **Random shuffle the sequence:** In order to generate the negative data, we random shuffle all the DNA sequence of the positive data. During the random shuffle, the total numbers of each “ATCG” are the same, as the position has been randomly changed.

1 >MA0002-RUNX1→1(1)	1 >MA0002-RUNX1→1
2 AATTGTGGTTA	2 TGTTAATTAGG
3 >MA0002-RUNX1→2(1)	3 >MA0002-RUNX1→2
4 ATCTGTGGTTA	4 TTATGTGTGAC
5 >MA0002-RUNX1→3(1)	5 >MA0002-RUNX1→4
6 AATTGTGGTAA	6 CCTTTGATGTG
7 >MA0002-RUNX1→4(1)	7 >MA0002-RUNX1→5
8 TTCTGCGGTTA	8 TGCGAAGATAT
9 >MA0002-RUNX1→5(1)	9 >MA0002-RUNX1→6
10 AATTGCGGTAA	10 TTGCGATTGTG
11 >MA0002-RUNX1→6(1)	11 >MA0002-RUNX1→7
12 TATTGCGGTTT	12 AATTGCGTTGG

Positive data Random shuffle data

Figure 10: The example figure of random shuffle the sequence

- **Negative Data based on the Similarity:** This type of negative data is generated by the iterations that are made to check the similarity of each random shuffle sequence

with all the positive sequence according to the longest common subsequence. If there is one negative sequence that has over 0.7 similarity with the positive sequences, this negative sequence will be re-shuffled and the checking of the iterations will start from the beginning again.

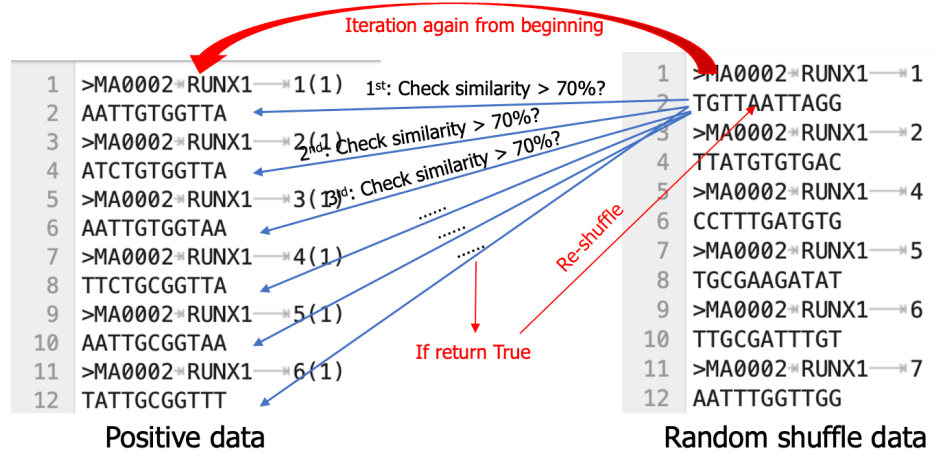


Figure 11: The example figure of the algorithm for negative data based on the similarity

- For example, if now the negative sequence we are checking is “TGTTAATTAGG” and the positive sequence now we are checking is “AATTGTGGTTA”, then the longest common subsequence is “AATTGG” and the similarity is lower than 0.7 ($6/11 < 0.7$). After that we move to the next positive sequence, for example “CCCTAATTAGG”, and now the longest common subsequence is “TAATTAGG” and the similarity is higher than 0.7 ($8/11 > 0.7$). Since now the similarity is over 0.7, so the negative sequence will be random shuffled again and check from the first positive sequence.
- However, this method takes too much time since the complexity is too high for the iteration (about n^3), so the negative data generated in this way is not being used in the further modeling.
- Negative Data based on the frequency matrix:** Since JASPAR has provided the frequency matrix of each DNA sequence, another type of negative data is designed to be generated based on this frequency matrix, as less frequency in one position of the amino acid will lead to the higher probability of that nucleotide being chosen at that position [1].






Frequency matrix													 JASPAR	 TRANSFAC	 MEME	 RAW PFM	 Reverse comp.
A[10	12	4	1	2	2	0	0	0	8	13]					
C[2	2	7	1	0	8	0	0	1	2	2]					
G[3	1	1	0	23	0	26	26	0	0	4]					
T[11	11	14	24	1	16	0	0	25	16	7]					

Figure 12: The example frequency matrix of the DNA sequence [1]

- For example, we can see that “C” and “G” are the two nucleotides that have less frequency at the first position, so that “C” and “G” will have higher probability to be chosen at the first position in the negative sequence than “A” and “T”.
- Therefore, the negative data based on the frequency matrix will be used in the further modeling.

Features

- DNA One-Hot encoding:
 - One-Hot encoding of DNA sequence is to transform each nucleotide (A, T, C, G, X) in the sequence into a numerical representation for easy computer processing. This encoding method assigns a unique binary vector to each nucleotide. For example, suppose there is a DNA sequence "ATCGX", after One-Hot encoding, the sequence will be transformed into:
 - A: [1, 0, 0, 0, 0]
 - T: [0, 1, 0, 0, 0]
 - C: [0, 0, 1, 0, 0]
 - G: [0, 0, 0, 1, 0]
 - U: [0, 0, 0, 0, 1]
 - Therefore, “ATCGX” will be transformed into “”1000001000001000001000001”.
- Since the length of the DNA sequence has been made into 11, so that the number of features after One-Hot encoding will be $11 * 5 = 55$.

```

▼ root:
AATTGTGGTTA: "1000010000010000100000010010000001000010010000100010000"
ATCTGTGGTTA: "1000001000001000100000010010000001000010010000100010000"
TTCTGCGGTTA: "010000100000100010000001000100000100001000010000100010000"
AATTGCGGTAA: "1000010000010000100000010001000001000010010001000010000"
TATTGCGGTTT: "0100010000010000100000010001000001000010010000100001000"
TATTGTGGTAG: "0100010000010000100000010010000001000010010001000000010"

```

Figure 13: The example One-Hot encoding result of DNA sequence

- DNA Word2Vec encoding [2]:
 - Word2Vec encoding (<https://github.com/dav/word2vec>) of DNA sequences is a process involving natural language processing techniques designed to capture contextual information about the nucleotides in the sequence [2].
 - Word2Vec is a predictive model used to capture patterns in the sequence by treating each nucleotide or combination of nucleotides as a "word" when dealing with DNA sequences [2].
 - The number of features after Word2Vec encoding will be 99 for every DNA sequences.

```

▼ root:
  ► AATTGTGGTTA: [] 99 items
  ► ATCTGTGGTTA: [] 99 items
  ► TTCTGCGGTTA: [] 99 items
  ► AATTGCGGTAA: [] 99 items
  ► TATTGCGGTTT: [] 99 items
  ► TATTGTGGTAG: [] 99 items
  ► TTTTGTGGTAG: [] 99 items
  ► AACTGCGGTTG: [] 99 items
  ► GACTGTGGTTG: [] 99 items
  ► CTTTGTGGTTA: [] 99 items

```

Figure 13: The example Word2Vec encoding result of DNA sequence

- Protein Embedding encoding:
 - ESM (Evolutionary Scale Modeling, <https://github.com/facebookresearch/esm>) encoding of protein sequences is a method that uses deep learning technology to capture the evolutionary information and functional characteristics of protein sequences [3].
 - ESM models are a series of pre-trained protein language models that can learn rich biological information from large-scale protein sequence data [3]. The number of features after ESM encoding will be 1280 for every protein sequences.

```

▼ root:
  ► Q01196: [] 1280 items
  ► P05549: [] 1280 items
  ► P10275: [] 1280 items
  ► Q02548: [] 1280 items
  ► P10589: [] 1280 items
  ► P16220: [] 1280 items
  ► Q01094: [] 1280 items
  ► Q16649: [] 1280 items

```

Figure 14: The example ESM encoding result of protein sequence

- In order to make the feature generating more convenient, the DNA-Protein table and json files for all the features have been produced and stored. Meanwhile, there are totally 1434 ($55+99+1280 = 1434$) features and one more label.

	0	1	2	3	4	5	6	7	8	9	...
0	-0.000844	-0.014036	-0.015465	-0.012514	-0.023681	-0.008265	-0.002055	-0.016170	-0.020259	-0.022946	...
1	-0.018373	-0.022141	-0.023991	-0.019311	-0.021805	-0.008230	-0.004362	-0.022833	-0.007556	-0.020092	...
2	0.001114	-0.010149	-0.016904	-0.019715	-0.017386	-0.005834	-0.007648	-0.021245	-0.031576	-0.009354	...
3	-0.015268	-0.007109	-0.015873	-0.012890	-0.022528	0.001145	-0.001635	-0.025455	-0.017402	-0.023978	...
4	-0.011762	-0.017276	-0.022208	-0.017101	-0.010692	-0.002517	-0.003789	-0.025275	-0.029589	-0.038132	...
...
9908	-0.022664	-0.013557	-0.018619	-0.008504	-0.022804	-0.010535	-0.000848	-0.024650	-0.016125	-0.025386	...
9909	-0.020185	-0.010761	-0.022601	-0.025691	-0.018874	-0.003058	-0.007401	-0.025681	-0.020910	-0.019722	...
9910	-0.017165	-0.014091	-0.032369	-0.025717	-0.011411	-0.004130	-0.013824	-0.030792	-0.031538	-0.025976	...
9911	0.001114	-0.010149	-0.016904	-0.019715	-0.017386	-0.005834	-0.007648	-0.021245	-0.031576	-0.009354	...
9912	-0.014038	-0.014724	-0.018645	-0.010333	-0.018876	-0.004725	-0.003228	-0.025389	-0.022467	-0.027934	...

9913 rows × 1435 columns

Figure 15: The feature table of 9913 samples small try

Modeling

- After having all these features, the aim is to build a model that can classify whether the DNA and the protein are binding or not. In other words, from the figure 1 we can see that each DNA has a binding protein (Uniport ID), so that we want to classify that by given a protein sequence and DNA sequences, whether they are correctly correlated to each other in the JASPAR [1].
- Now we have the positive data (directly downloaded from JASPAR) given label “1”, the negative data (generated by frequency matrix) given label “0” and the protein data.
- **Classification 1: Small try with 9913 samples**
 - Before the real classification begins, a tiny try with 9913 sample has been executed to analysis how much sources will be needed (memory, CPU...).

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9829	0.9977	0.9821	0.9835	0.9828	0.9657	0.9657	4.0400
et	Extra Trees Classifier	0.9784	0.9977	0.9714	0.9851	0.9781	0.9568	0.9569	0.4310
rf	Random Forest Classifier	0.9738	0.9962	0.9668	0.9804	0.9735	0.9475	0.9477	0.5120
lightgbm	Light Gradient Boosting Machine	0.9719	0.9959	0.9691	0.9744	0.9717	0.9438	0.9438	153.7530
gbc	Gradient Boosting Classifier	0.9098	0.9701	0.9040	0.9143	0.9090	0.8196	0.8198	5.3830
dt	Decision Tree Classifier	0.9029	0.9029	0.9034	0.9022	0.9027	0.8057	0.8060	0.4330
knn	K Neighbors Classifier	0.7853	0.8704	0.8728	0.7424	0.8021	0.5708	0.5800	1.7300
lr	Logistic Regression	0.6734	0.7400	0.6739	0.6726	0.6729	0.3469	0.3472	0.7140
ridge	Ridge Classifier	0.6714	0.0000	0.6719	0.6707	0.6709	0.3428	0.3432	0.2320
ada	Ada Boost Classifier	0.6695	0.7289	0.6658	0.6701	0.6676	0.3391	0.3394	0.9320
lda	Linear Discriminant Analysis	0.6671	0.7323	0.6566	0.6702	0.6627	0.3342	0.3348	2.8350
svm	SVM - Linear Kernel	0.6613	0.0000	0.6797	0.6587	0.6656	0.3228	0.3267	0.3010
nb	Naive Bayes	0.5409	0.5523	0.6256	0.5338	0.5759	0.0822	0.0834	0.1640
qda	Quadratic Discriminant Analysis	0.4960	0.4974	0.9592	0.4971	0.6547	-0.0051	-0.0136	0.9390

Figure 16: The classification result of 9913 samples small try

- Extreme Gradient Boosting and Extra Trees Classifier have the overall highest performance. Extreme Gradient Boosting performs well on almost all evaluating characters, especially on AUC (0.9977), which means it is very good at distinguishing different categories.
- On the other hands, the performance of Extra Trees Classifier is very close to Extreme Gradient Boosting, while it has an advantage in training time, requiring only 0.4310 seconds, which is far less than Extreme Gradient Boosting's 4.0400 seconds.
- By the way, the modeling of 9913 samples were done on the lab server. However, the modeling for all the sequences would take a lot of sources, so the rest of the modeling were done on the NSCC (National Super-Computing Center, <https://www.nscc.sg/>) [4].

- **Classification 2: Training : Testing = 80 : 20 and positive : negative = 1:1**
 - In this situation, 80% of samples are assigned to be training dataset and rest will be testing dataset. Meanwhile, the number of positive data and negative data are same in both training and testing dataset.
 - The number of training dataset = 224634; The number of testing dataset = 56158
 - The number of positive data in training dataset = The number of negative data in training dataset = 112317; The number of positive data in testing dataset = The number of negative data in testing dataset = 28079.

Model	Accuracy	AUC	Recall	Precision	F1 score	Kappa	MCC	TT (Sec)
Extra Trees Classifier	0.9961	0.9997	0.9953	0.9969	0.9961	0.9922	0.9922	6.263
Random Forest Classifier	0.9959	0.9998	0.9953	0.9965	0.9959	0.9918	0.9918	7.294
Decision Tree Classifier	0.9849	0.9850	0.9874	0.9825	0.9850	0.9698	0.9698	8.296
SVM – Linear Kernel	0.6751	0.0000	0.6821	0.5731	0.6771	0.3502	0.3507	1.745
Naïve Bayes	0.5424	0.6245	0.6433	0.5353	0.5843	0.0848	0.0866	0.865

Table 1: The classification 2 result

- Extra Trees Classifier has extremely high scores on all evaluating characters, achieving almost perfect predictive power, especially on AUC (0.9997). The training time of this model (6.263 seconds) is also the shortest among this group of models.
- **Classification 3: Each protein are divided into 80% training and 20% testing**
 - The DNA sequence of each corresponding protein are divided into 80% training and 20% testing, while also keeping positive : negative = 1:1.

Model	Accuracy	AUC	Recall	Precision	F1 score	Kappa	MCC	TT (Sec)
SVM – Linear Kernel	0.6116	0.0000	0.5901	0.6177	0.6023	0.2232	0.2239	1.883
Random Forest Classifier	0.5872	0.7115	0.2060	0.8415	0.3140	0.1743	0.2560	6.075
Extra Tree Classifier	0.5843	0.7146	0.1895	0.8834	0.2932	0.1687	0.2628	3.794
Decision Tree Classifier	0.5838	0.5839	0.4783	0.5960	0.5285	0.1677	0.1694	5.956
Naïve Bayes	0.5288	0.5888	0.6278	0.5235	0.5704	0.0576	0.0596	0.923

Table 2: The classification 3 result

- The AUC scores of Random Forest Classifier and Extra Trees Classifier (0.7115 and 0.7146 respectively) are better than other models, but still lower than usual

expectations. The accuracy of these two models (0.5872 and 0.5843, respectively) is also below the usual acceptance standards, indicating that the models were having difficulty classifying. However, the Extra Trees Classifier scores higher on Precision (0.8834), indicating that when the model makes positive class predictions, it is generally accurate.

- **Classification 4: No common protein in training set and testing set**
 - There is no common protein in training set and testing set, and the protein structure of training and testing are similar, which is shown in the below graph.

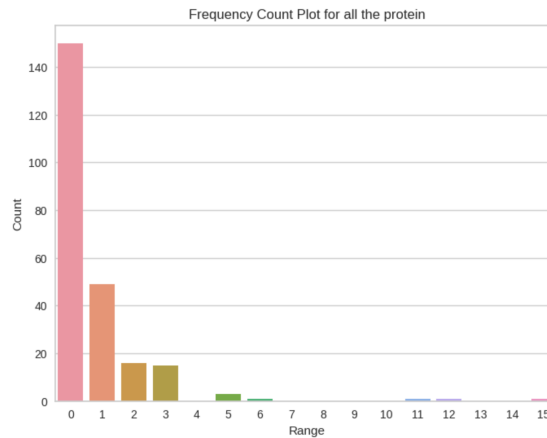


Figure 17: The plot of frequency count plot of all the protein

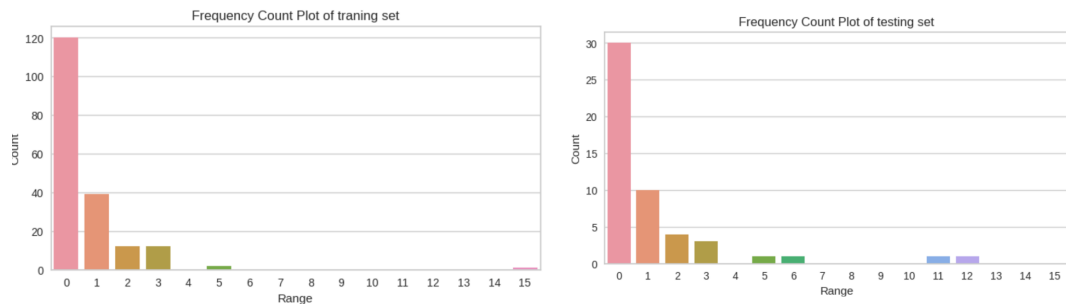


Figure 18 and 19: The plot of frequency count plot of training set and testing set

Model	Accuracy	AUC	Recall	Precision	F1 score	Kappa	MCC	TT (Sec)
SVM – Linear Kernel	0.5741	0.0000	0.5897	0.5723	0.5794	0.1482	0.1494	2.199
Random Forest Classifier	0.5580	0.6498	0.1655	0.7360	0.2620	0.1161	0.1735	6.303
Extra Tree Classifier	0.5562	0.6437	0.1490	0.7747	0.2445	0.1123	0.1829	5.429
Decision Tree Classifier	0.5554	0.5554	0.4643	0.5665	0.5084	0.1109	0.1126	6.801
Naïve Bayes	0.5289	0.5602	0.6141	0.5248	0.5658	0.0578	0.0587	1.393

Table 3: The classification 4 result

- Random Forest Classifier has AUC of 0.6498, indicating that it has some predictive power, but it is still not ideal. Although its precision is relatively high (0.7360), the overall F1 score (0.2620) and MCC (0.1735) are relatively low, indicating that the model has room for improvement in distinguishing between positive and negative classes.
- Extra Trees Classifier has AUC of 0.6437, which is similar to Random Forest and also indicates a certain degree of predictive ability. Its precision (0.7747) is the highest among all models, but its F1 score (0.2445) and MCC (0.1829) are lower, possibly due to the model having a high false positive rate.
- **Classification 5: C2H2 zinc finger factors**
 - This classification will be only done on the class of C2H2 zinc finger factors applying the approach of classification 4.

	ID	Name	Class	Family	Data type	Uniprot ID
23	MA0039.3	KLF4	C2H2 zinc finger factors	Three-zinc finger Kruppel-related	ChIP-seq	O43474
24	MA0039.4	KLF4	C2H2 zinc finger factors	Three-zinc finger Kruppel-related	ChIP-seq	O43474
34	MA0056.1	MZF1	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	SELEX	P28698
35	MA0056.2	MZF1	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	ChIP-seq	P28698
51	MA0073.1	RREB1	C2H2 zinc finger factors	Factors with multiple dispersed zinc fingers	SELEX	Q92766
...
325	MA1973.1	ZKSCAN3	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	NaN	Q9BRR0
326	MA1974.1	ZNF211	C2H2 zinc finger factors	Factors with multiple dispersed zinc fingers	NaN	Q13398
327	MA1979.1	ZNF416	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	NaN	Q8NA42
328	MA1983.1	ZNF582	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	NaN	Q96NG8
329	MA1986.1	ZNF692	C2H2 zinc finger factors	More than 3 adjacent zinc fingers	NaN	Q9BU19

84 rows x 6 columns

Figure 20: The figure of data belongs to C2H2 zinc finger factors class

- The accuracy for C2H2 zinc finger factors class is 0.5741, which is not satisfying.

Conclusion

- After all, we have done many classifications toward the DNA-protein binding modeling. And the result shows satisfying in the classification of 80% of samples are assigned to be training dataset and rest will be testing dataset, as the number of positive data and negative data are same in both training and testing dataset (Classification 2).
- However, once we have each protein divided into 80% training and 20% testing (Classification 3) or make there is no common protein in training set and testing set (Classification 4), the performance of the modeling will reduce and would not reach

our expectation. But there are some models perform well in some aspects (precision and MCC) of the classification.

- Besides, the models with decision trees as basic structural showed better performance than other models, so that we need to do more analyzation toward this in the future.
- In conclusion, if we have a huge dataset that including all the DNA sequences and correlated protein sequences, we can train a model that have satisfying performance. However, such huge dataset requires really a lot of sources. Therefore, the features of One-hot encoding, Word2Vec encoding and ESM encoding are not enough for modeling, as we should discover some new features to show the deep-dive relationships of the DNA and protein.

Reference:

- [1] I. Rauluseviciute et al., “Jaspar 2024: 20th anniversary of the open-access database of transcription factor binding profiles,” *Nucleic Acids Research*, vol. 52, no. D1, Nov. 2023. doi:10.1093/nar/gkad1059
- [2] K. W. CHURCH, “Word2Vec,” *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, Dec. 2016. doi:10.1017/s1351324916000334
- [3] R. Verkuil et al., *Language models generalize beyond natural proteins*, Dec. 2022. doi:10.1101/2022.12.21.521521
- [4] “National Super-Computing Center,” NSCC, <https://www.nscg.sg/> (accessed Feb. 21, 2024).