ALGORTHIMS IN BIO INFORMATICS

Homework Assignment 5 – 05/09/2023

Haridas Aravind – 02071139

1. **Term project:**

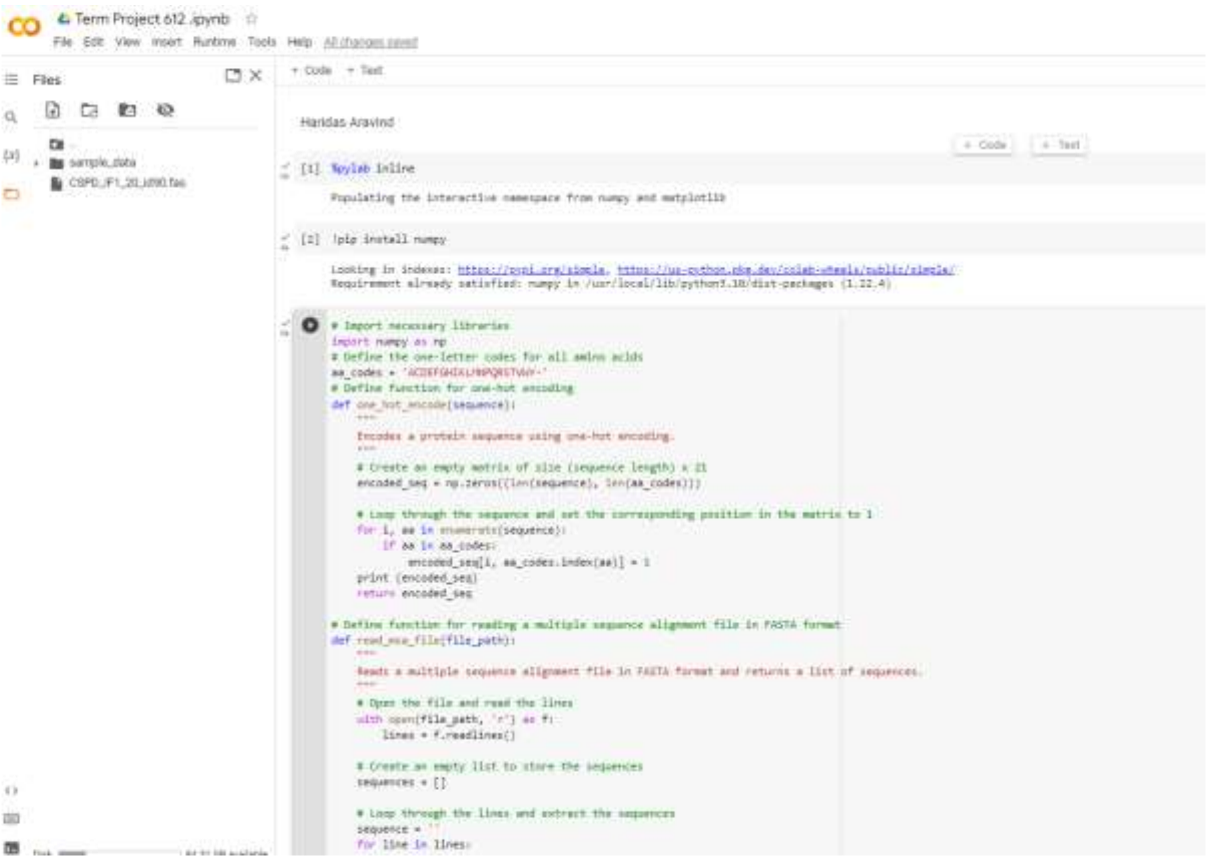
I have written a code on the one-hot encoding from the one-letter codes for all amino acids take from the last assignment and I have taken the sequence from the sequence folder from the course web page.

Python Code:

```python
# Import necessary libraries
import numpy as np
# Define the one-letter codes for all amino acids
aa_codes = 'ACDEFGHIKLMNPQRSTVWY-'
# Define function for one-hot encoding
def one_hot_encode(sequence):
    """
    Encodes a protein sequence using one-hot encoding.
    """
    # Create an empty matrix of size (sequence length) x 21
    encoded_seq = np.zeros((len(sequence), len(aa_codes)))

    # Loop through the sequence and set the corresponding position in the matrix to 1
    for i, aa in enumerate(sequence):
        if aa in aa_codes:
            encoded_seq[i, aa_codes.index(aa)] = 1
    print (encoded_seq)
    return encoded_seq

# Define function for reading a multiple sequence alignment file in FASTA format
def read_msa_file(file_path):
    """
    Reads a multiple sequence alignment file in FASTA format and returns a list of sequences.
    """
    # Open the file and read the lines
    with open(file_path, 'r') as f:
        lines = f.readlines()

    # Create an empty list to store the sequences
    sequences = []

    # Loop through the lines and extract the sequences
    sequence = ''
    for line in lines:
        line = line.strip()
```

```
            if line.startswith('>'):
                if sequence:
                    sequences.append(sequence)
                    sequence = ''
            else:
                sequence += line

        print(sequences)
        sequences.append(sequence)

        return sequences

# Example usage
file_path = '/content/CSPD_IF1_20_id90.fas'
msa_sequences = read_msa_file(file_path)
encoded_sequences = [one_hot_encode(seq) for seq in msa_sequences]

#Print the first encoded sequence
print("Sequences done.")
```

Google Collab Output:

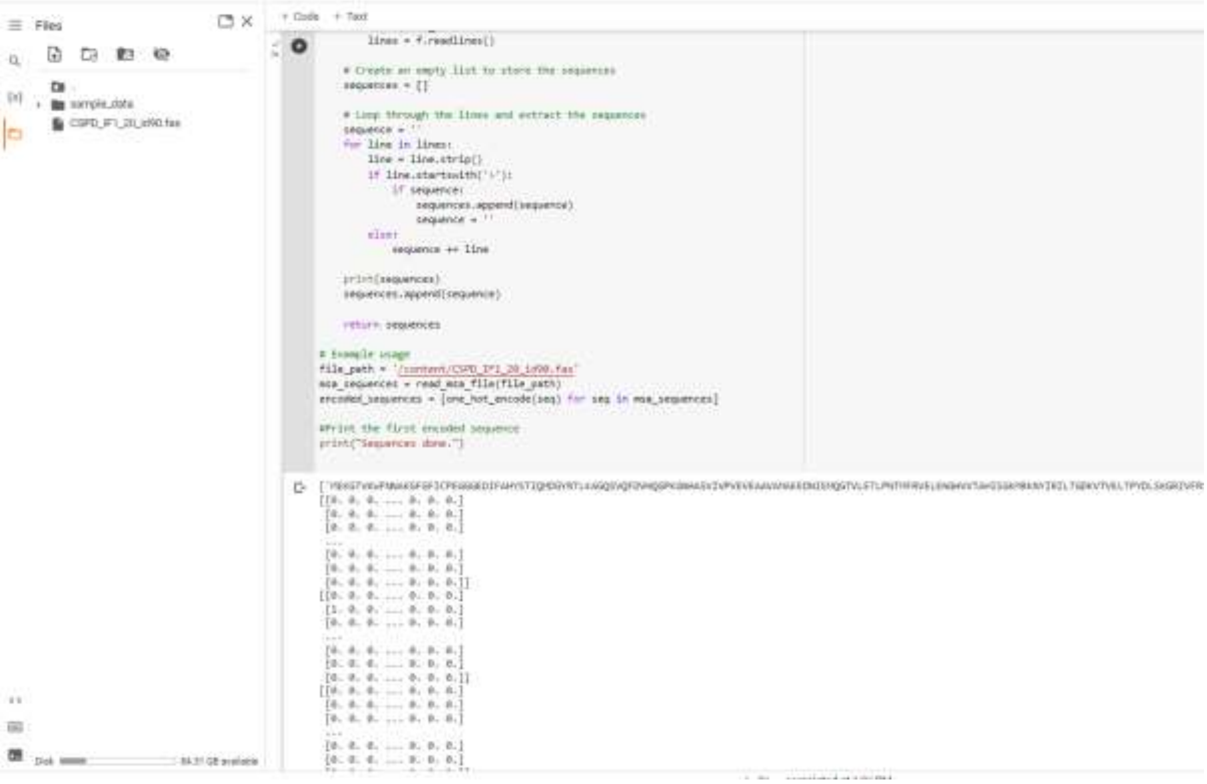


Haridas Aravind

```
[1] %pylab inline

    Populating the interactive namespace from numpy and matplotlib

[2] !pip install numpy

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.22.4)
```

```
# Import necessary libraries
import numpy as np
# Define the one-letter codes for all amino acids
aa_codes = 'ACDEFGHIKLMNPQRSTVWY-'
# Define function for one-hot encoding
def one_hot_encode(sequence):
    """
    Encodes a protein sequence using one-hot encoding.
    """
    # Create an empty matrix of size (sequence length) x 21
    encoded_seq = np.zeros((len(sequence), len(aa_codes)))

    # Loop through the sequence and set the corresponding position in the matrix to 1
    for i, aa in enumerate(sequence):
        if aa in aa_codes:
            encoded_seq[i, aa_codes.index(aa)] = 1
    print (encoded_seq)
    return encoded_seq

# Define function for reading a multiple sequence alignment file in FASTA format
def read_msa_file(file_path):
    """
    Reads a multiple sequence alignment file in FASTA format and returns a list of sequences.
    """
    # Open the file and read the lines
    with open(file_path, 'r') as f:
        lines = f.readlines()

    # Create an empty list to store the sequences
    sequences = []

    # Loop through the lines and extract the sequences
    sequence = ''
    for line in lines:
```

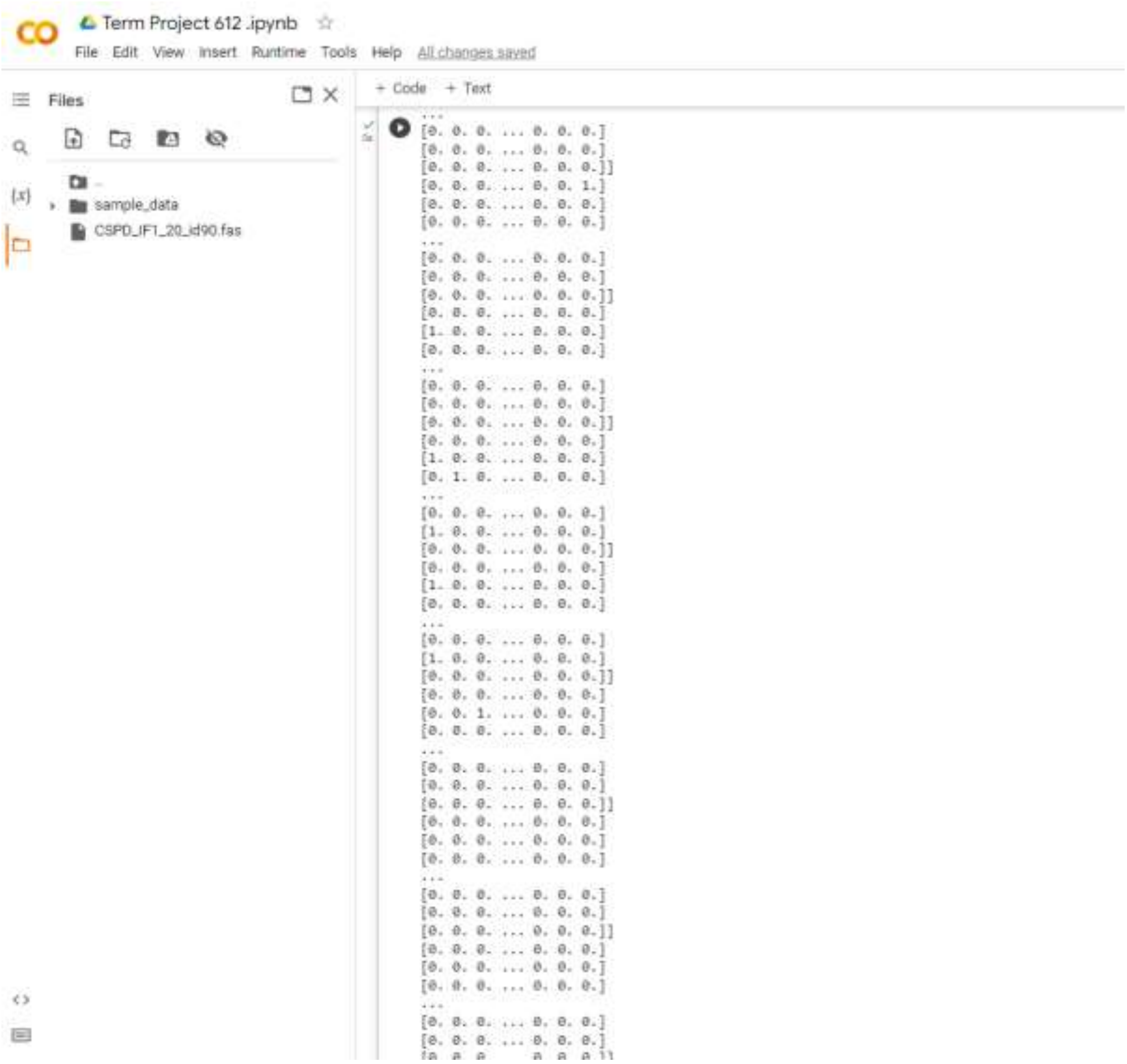
ALGORTHIMS IN BIO INFORMATICS



```
        lines = f.readlines()

    # Create an empty list to store the sequences
    sequences = []

    # Loop through the lines and extract the sequences
    sequence = ''
    for line in lines:
        line = line.strip()
        if line.startswith('>'):
            if sequence:
                sequences.append(sequence)
                sequence = ''
        else:
            sequence += line

    print(sequences)
    sequences.append(sequence)

    return sequences

# Example usage
file_path = '/content/CSPD_IF1_20_id90.fas'
msa_sequences = read_msa_file(file_path)
encoded_sequences = [one_hot_encode(seq) for seq in msa_sequences]

#Print the first encoded sequence
print("Sequences done.")
```

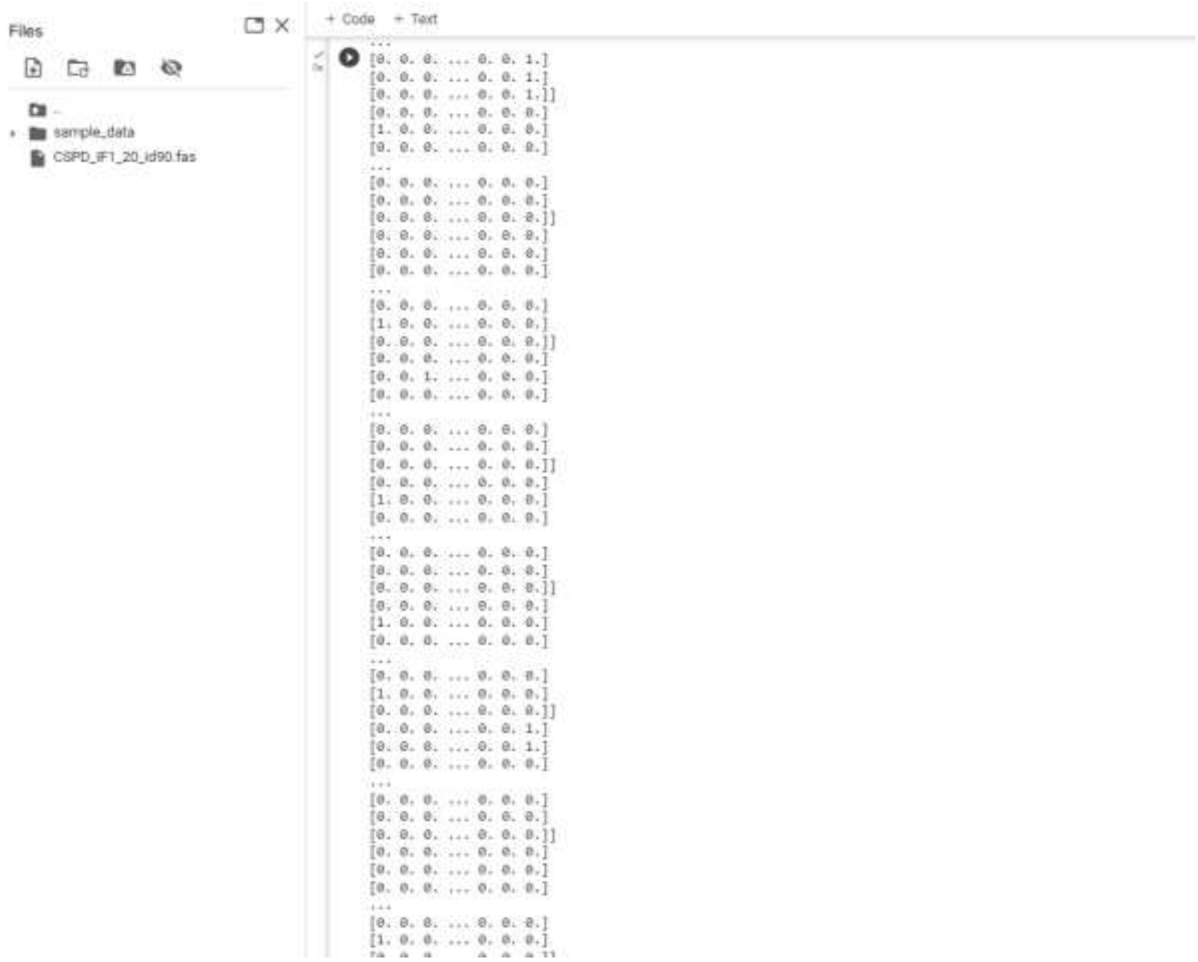
```
['[illegible]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
```

ALGORTHIMS IN BIO INFORMATICS



```
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 1.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 1. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 1. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
```

ALGORTHIMS IN BIO INFORMATICS



```
...
[0. 0. 0. ... 0. 0. 1.]
[0. 0. 0. ... 0. 0. 1.]
[0. 0. 0. ... 0. 0. 1.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 1. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 1.]
[0. 0. 0. ... 0. 0. 1.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
```

Here is the link for my code and the output is very big sequence I have attached a collab link here:
https://github.com/Haridasaravind/CS612/blob/main/Term_Project_612_.ipynb

ALGORTHIMS IN BIO INFORMATICS

2. Term project:

Our project is based on the Project: Autoencoder for Protein Multiple Sequence Alignments

Autoencoder

- Autoencoders are a type of neural network that can be used to learn the essential features of a dataset.
- Autoencoders have been used in bioinformatics to predict protein-protein interactions, protein secondary structure, protein function, and protein dynamics.
- Autoencoders are a powerful tool for analysing protein sequences and gaining insights into their structure and function.
- Autoencoders are still under development, but they have the potential to revolutionize the field of bioinformatics.

Variational Auto Encoders (VAE)

- VAEs are a type of neural network that can be used to learn the essential features of a dataset, while also preventing overfitting.
- VAEs do this by learning a distribution over the latent space, rather than a single point.
- This distribution is learned by minimizing a loss function that is a combination of a reconstruction loss and a regularization loss.
- VAEs have been used in protein studies for a variety of tasks, including predicting protein structure, generating protein variants, and exploring protein conformational space.

Encoding Protein Sequences

Amino acid encoding is the process of converting protein sequences into digital representations that can be used by machine learning models.

The most common amino acid encodings are one-hot encoding, PSSM encoding, and physico-chemical properties encoding.

- One-hot encoding is a high-dimensional and sparse vector representation, while PSSM encoding and physico-chemical properties encoding are lower-dimensional and more dense representations.
- The choice of amino acid encoding depends on the specific machine learning model being used and the desired accuracy.

My Analysis: I have successfully constructed an autoencoder for protein multiple sequence alignments. The autoencoder can encode a latent space of a small size, anything between 5 and 32, up to you, and decode the sequences back. The reconstruction error is measured by the following parameters:

The learning error (cross-entropy, kl divergence, etc.).

The ability to reconstruct the same amino acid in each position.

The ability to reconstruct the most common amino acid in each position in the alignment.

Difficulties: I ran into a few difficulties while constructing the autoencoder. One difficulty was choosing the right encoding for the protein sequences. I’ve decided to use one-hot encoding, which is a high-dimensional and sparse vector representation. Another difficulty was choosing the right parameters for the autoencoder. I’ve experimented with different values for the number of hidden layers, the number of neurons per layer, and the activation functions.

Choice of programming language: I’ve chosen to use Python for the implementation of the autoencoder. Python is a popular programming language that is well-suited for machine learning tasks. It is also a relatively easy language to learn, which made it a good choice for this project.

Implementation: I’ve implemented the autoencoder using the Keras library. Keras is a high-level neural network library that is built on top of TensorFlow. It is a popular library for machine learning tasks, and it is relatively easy to use.

I’m continuing to work on making the autoencoder more accurate. I’m trying different ways of encoding the protein sequences, as well as different parameters for the autoencoder. I’ve also working on ways to visualize the latent space. I believe that the autoencoder has the potential to be a valuable tool for analysing protein sequences and gaining insights into their structure and function.

3. Geometric Hashing:

A method for indexing and retrieving geometric data structures is geometric hashing. It is used in many areas of Bio informatics.
It is used to index and retrieve protein structures. It is a powerful technique that is fast, efficient, scalable, and robust. However, it can be computationally expensive to create the hash tables for large datasets of protein structures, and it can be difficult to choose the right features for the protein structures.
It has two stages:

Pre-processing or Training: In the training stage, a set of geometric data structures are pre-processed and stored in a hash table. The pre-processing step involves extracting features from each data structure and then quantizing the features. The hash table is then created by mapping each feature to a unique hash value.

Recognition stage: In the recognition stage, a new geometric data structure is compared to the data structures in the hash table. The features of the new data structure are extracted and quantized. The quantized features are then used to look up the hash value in the hash table. If the hash value is found in the hash table, then the corresponding data structure is returned.

(a) 2-dimensions, translation.

We need two variables to establish a reference frame in 2D translation. One variable for the x-axis and one variable for the y-axis.

(b) 2-dimensions, translation and rotation.

We need three variables to establish a reference frame in 2D translation and rotation. One variable for the x-axis, one variable for the y-axis, and one variable for the angle of rotation.

(c) 3-dimensions, translation.

We need three variables to establish a reference frame in 3D translation. One variable for the x-axis, one variable for the y-axis, and one variable for the z-axis.

(d) 3-dimensions, translation and rotation.

We need six variables to establish a reference frame in 3D translation and rotation. One variable for the x-axis, one variable for the y-axis, one variable for the z-axis, one variable for the angle of rotation around the x-axis, one variable for the angle of rotation around the y-axis, and one variable for the angle of rotation around the z-axis.

(e) 2-dimensions, scaling

We need one variable to establish a reference frame in 2D scaling. One variable for the scale factor.

ALGORTHIMS IN BIO INFORMATICS

4. Geometric hashing:

We can divide a two-dimensional space into a series of squares of width w. Each square is called a "bin". The bins are indexed as though they are a two-dimensional array. When data points are read, they are assigned to bins.

Suppose if w = 8, the space would be divided into 16 bins. The bin containing four points and bounded on the left by 8, on the right by 16, on the bottom by 0 and on the top by 8 is bin (1, 0). The bin containing three points bound on the right by -8, on the left by -16, on the bottom by 8 and on the top by 16 is bin (-2,1) etc.

In general, a bin index (i,j) given coordinates (x,y) is $i = \lfloor x/w \rfloor$ and $j = \lfloor y/w \rfloor$

(a). The other non-empty bins are indexed and the number of points in each one is as follows:

Bin (-1,0) contains 3 points and other are:

(-2, 1) | 3

(2, 1) | 2

(-1, 2) | 1

(1, 2) | 1

(b). Hash Function is defined as

$$h\,(i, j) = |i * 378551 + j * 63689|$$

For (8,10), we have $i = \lfloor 8/8 \rfloor = 1$ and $j = \lfloor 10/8 \rfloor = 1$, so the bin index is (1,1). The hash value is h (1,1) = |1378551 + 163689| = 442240.

For (-6,3), we have $i = \lfloor -6/8 \rfloor = -1$ and $j = \lfloor 3/8 \rfloor = 0$, so the bin index is (-1,0). The hash value is h (-1,0) = |-1378551 + 063689| = 378551.

For (27, -18), we have $i = \lfloor 27/8 \rfloor = 3$ and $j = \lfloor -18/8 \rfloor = -3$, so the bin index is (3, -3). The hash value is h (3, -3) = |3*378551 + (-3) *63689| = 1892755.

The square roots of the given numbers are:

$\sqrt{442240} \approx 665.207$

$\sqrt{378551} \approx 614.940$

$\sqrt{1892755} \approx 1376.013$

5. Reference Frame in 2D:

(a) To calculate the translation and rotation associated with this reference frame, we need to first find the vector between the two points, which represents the translation, and the angle between the x-axis and the vector, which represents the rotation.

slope of line p1p2 = (y2-y1)/(x2-x1) = (-17-0)/(14-0) = -1.21
slope of line p1p3 = (y3-y1)/(x3-x1) = (-13-0)/(-6-0) = 2.17
slope of line p1p4 = (y4-y1)/(x4-x1) = (-8-0)/(16-0) = -0.50

Next, we can use the inverse tangent function to find the angle between each line and the x-axis:

angle of line p1p2 with x-axis = arctan(-1.21) = -50.59 degrees
angle of line p1p3 with x-axis = arctan(2.17) = 64.99 degrees
angle of line p1p4 with x-axis = arctan(-0.50) = -26.57 degrees
Therefore, the angles of the points with respect to p1 are:

p2: -50.59 degrees
p3: 64.99 degrees
p4: -26.57 degrees

(b) Rotation:
To find the angle between the x-axis and the vector t, we use the formula:

To find the vector between p1 and p2, we subtract the coordinates of p1 from p2:
Given coordinates are.

Given Points

- p2 = (30, 6)
- p3 = (10, 10)
- p4 = (32, 15)

Translate the points so that p1 is at the origin:

p1' = (0, 0)
p2' = p2 - p1 = (30 - 16, 6 - 23) = (14, -17)
p3' = p3 - p1 = (10 - 16, 10 - 23) = (-6, -13)
p4' = p4 - p1 = (32 - 16, 15 - 23) = (16, -8)

The slope of a line passing through two points (x1, y1) and (x2, y2) is given by:

slope = (y2 - y1) / (x2 - x1)

we already know the slopes from the above

ALGORTHIMS IN BIO INFORMATICS

p1p2: To rotate the angle of p1p2 from 0 to 90 degrees, we need to add 90 degrees to the angle of p1p2 and adjust the result if it is greater than 90 degrees.

angle of line p1p2 with x-axis = -50.59 degrees
angle of line p1p2 rotated = -50.59 + 90 = 39.41 degrees

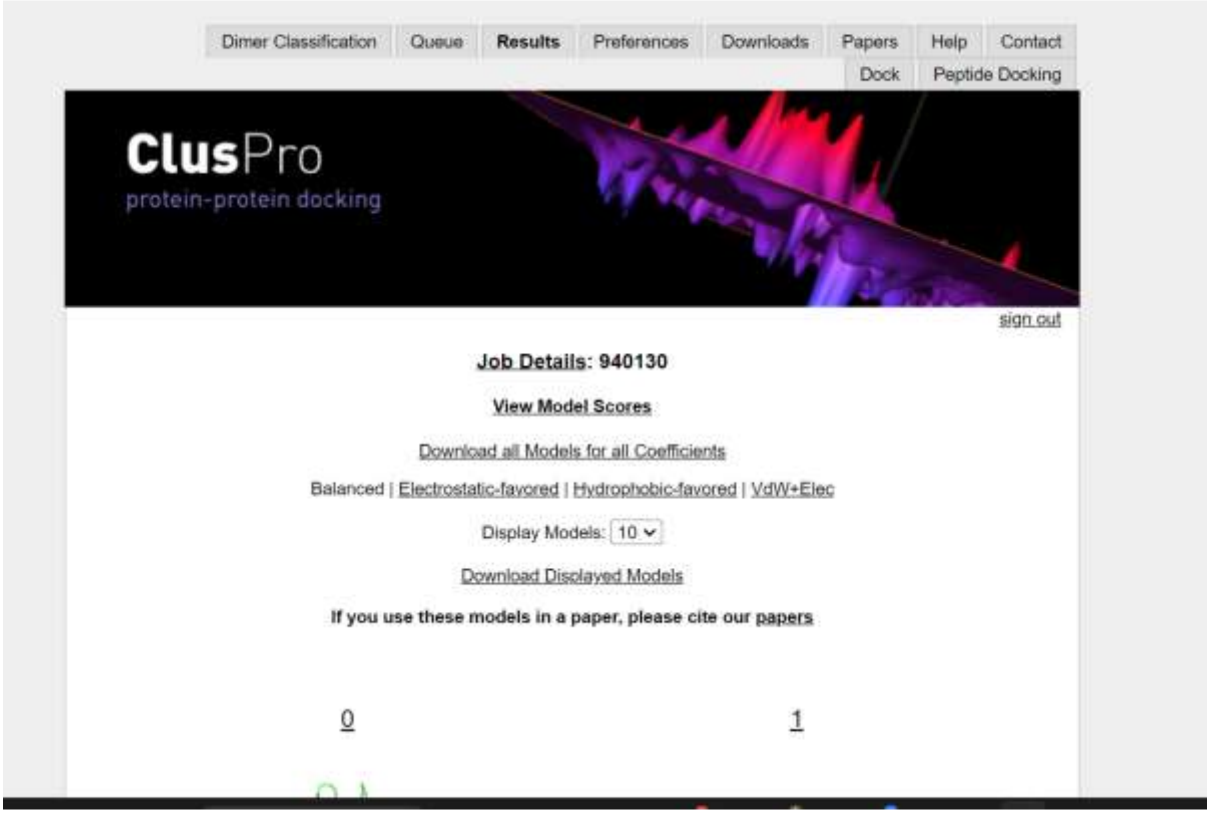
p1p3: angle of line p1p3 with x-axis = arctan(2.17) = 64.99 degrees

p1p4: To rotate the angle of p1p4 from 0 to 90 degrees, we need to add 90 degrees to the angle of p1p4 and adjust the result if it is greater than 90 degrees.

angle of line p1p4 with x-axis = -26.57 degrees
angle of line p1p4 rotated = -26.57 + 90 = 63.43 degrees

6. Docking:

(a).



Job Details: 940130

View Model Scores

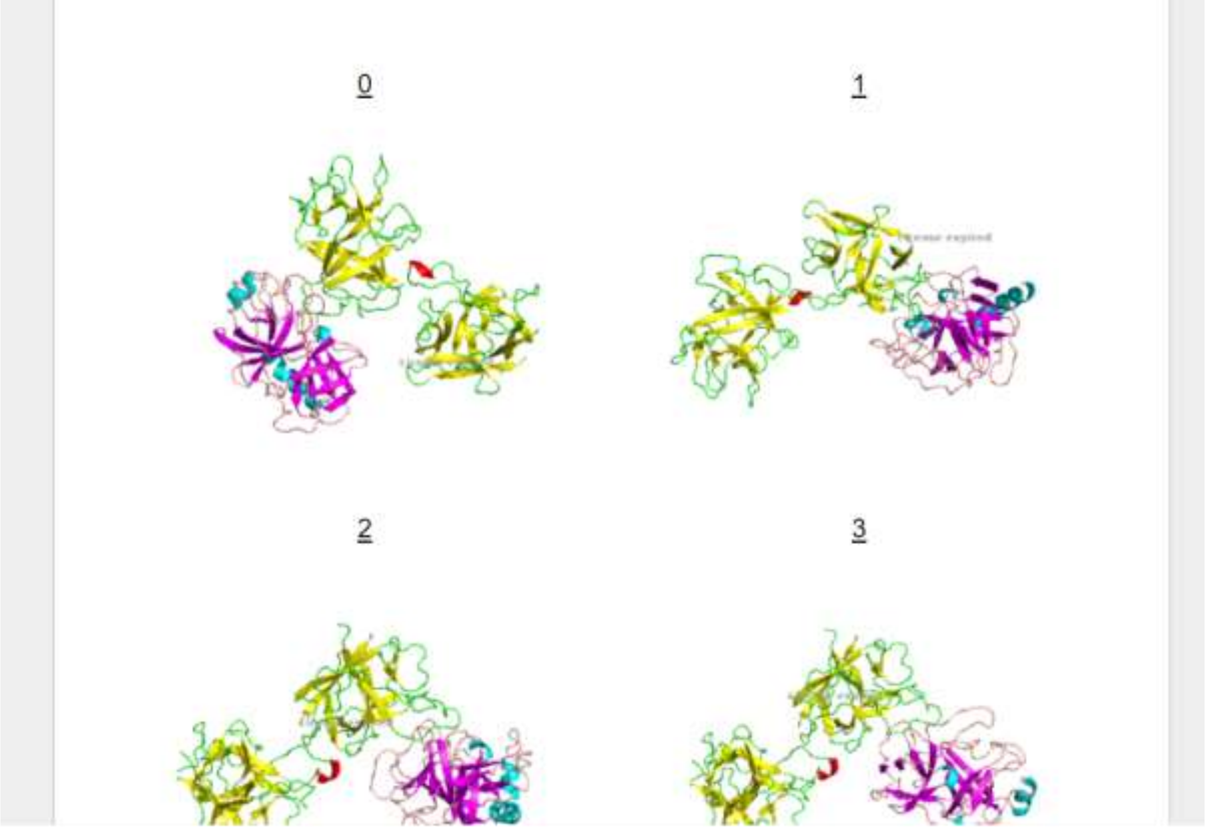
Download all Models for all Coefficients

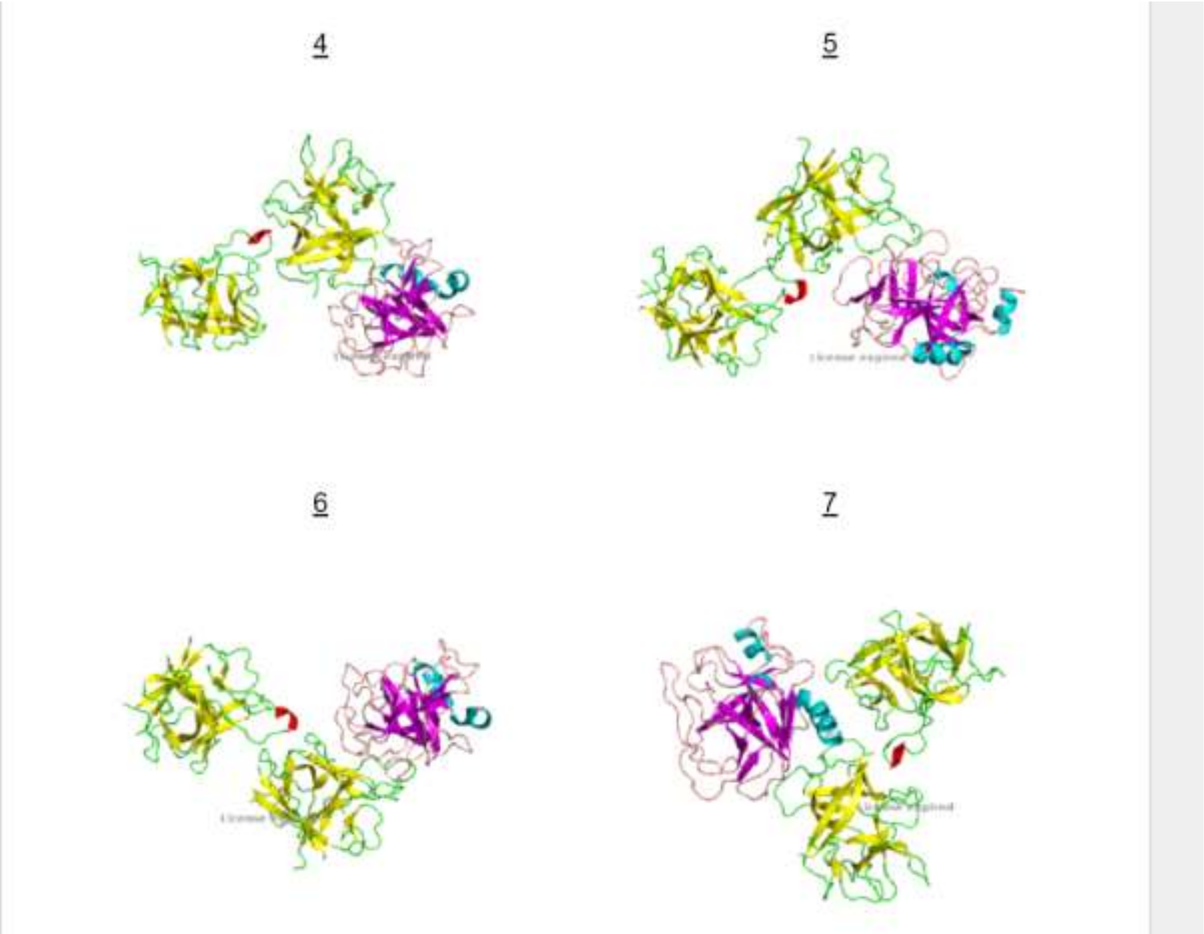
Balanced | Electrostatic-favored | Hydrophobic-favored | VdW+Elec

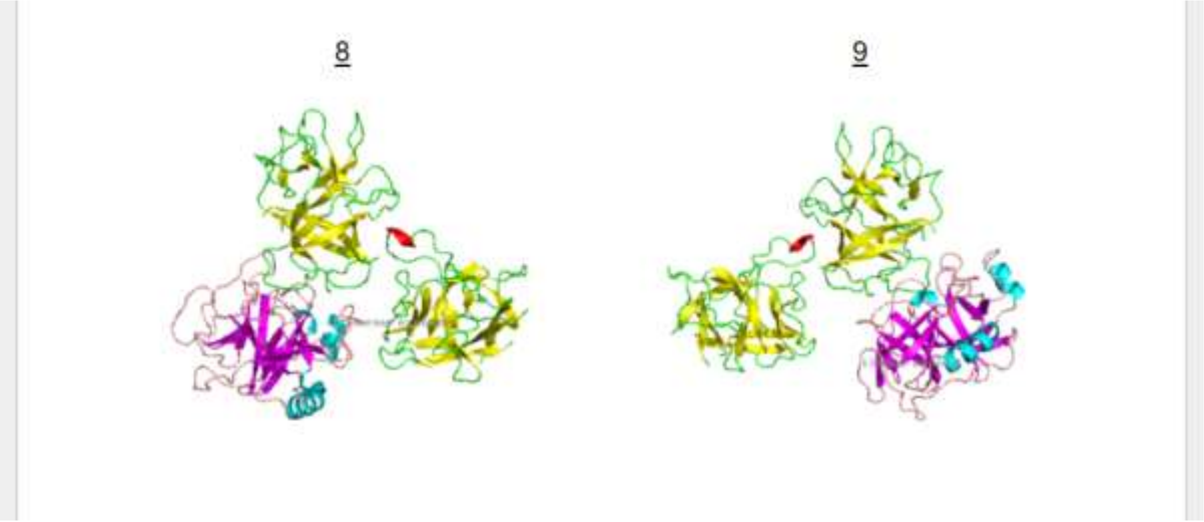
Display Models: 10

Download Displayed Models

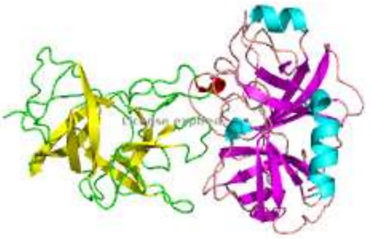
If you use these models in a paper, please cite our papers



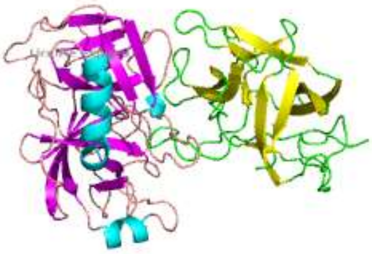

4
5
6
7


8
9

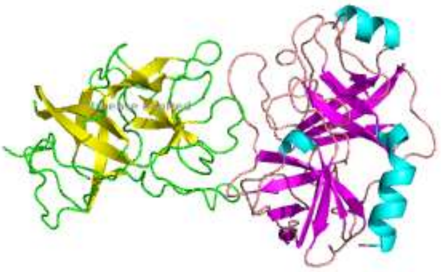
10



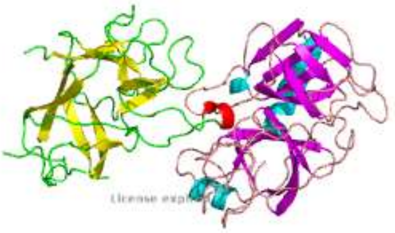
11



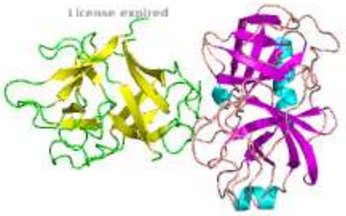
12



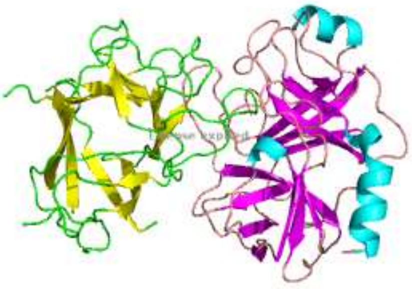
13



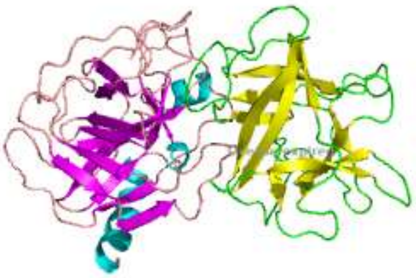
14



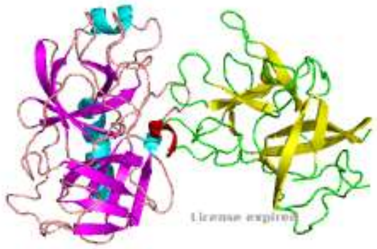
15



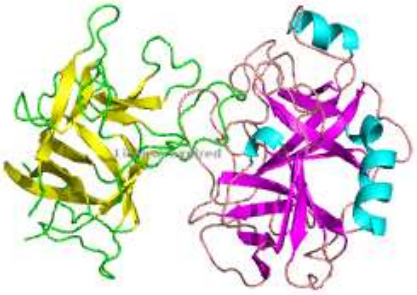
16



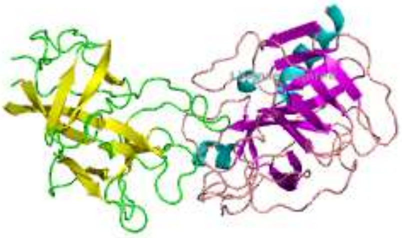
17



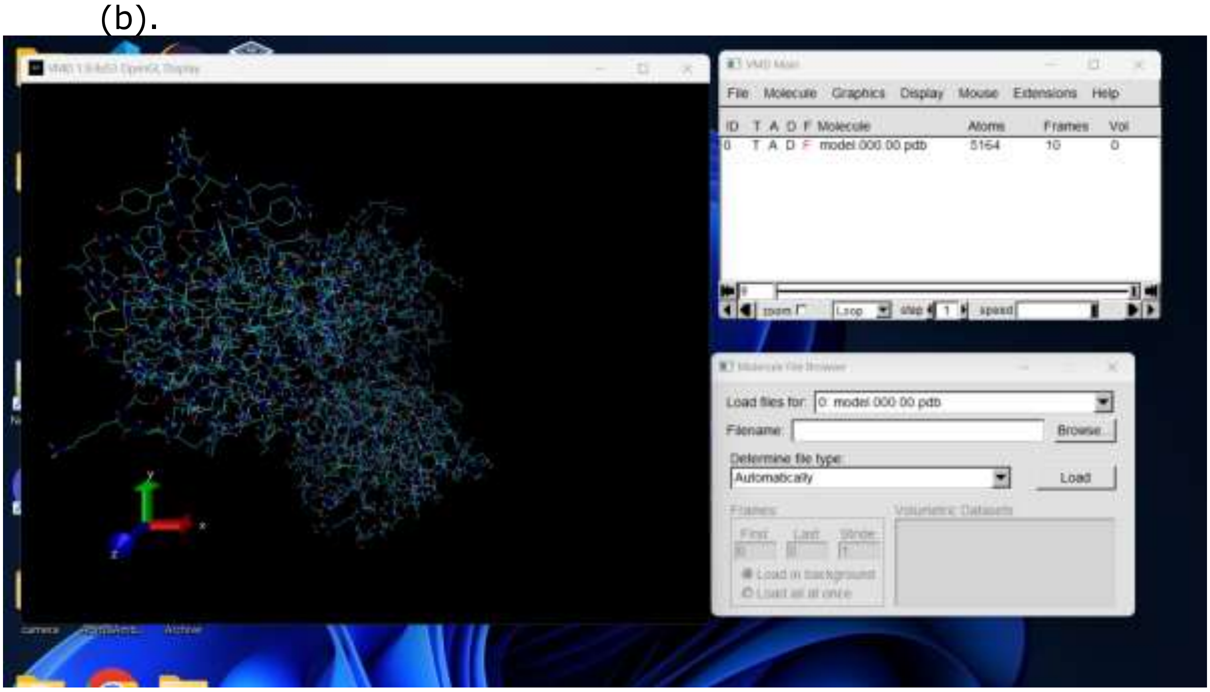
18



19



(b).



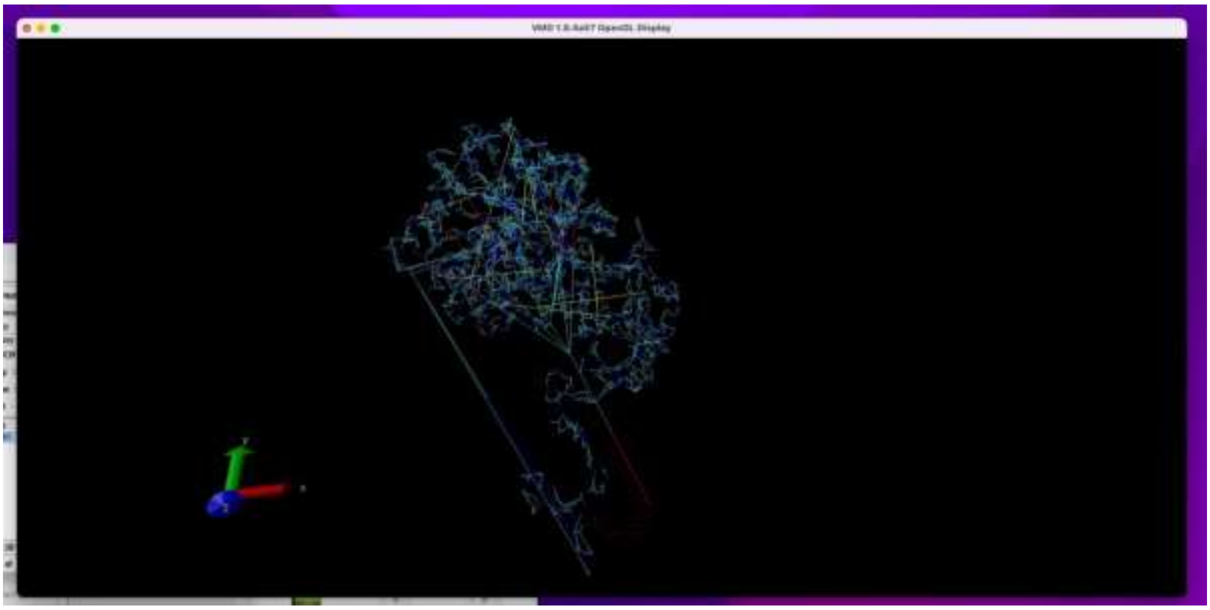
ID	T	A	D	F	Molecule	Atoms	Frames	Vol
0	T	A	D	F	model.000.00.pdb	5164	10	0

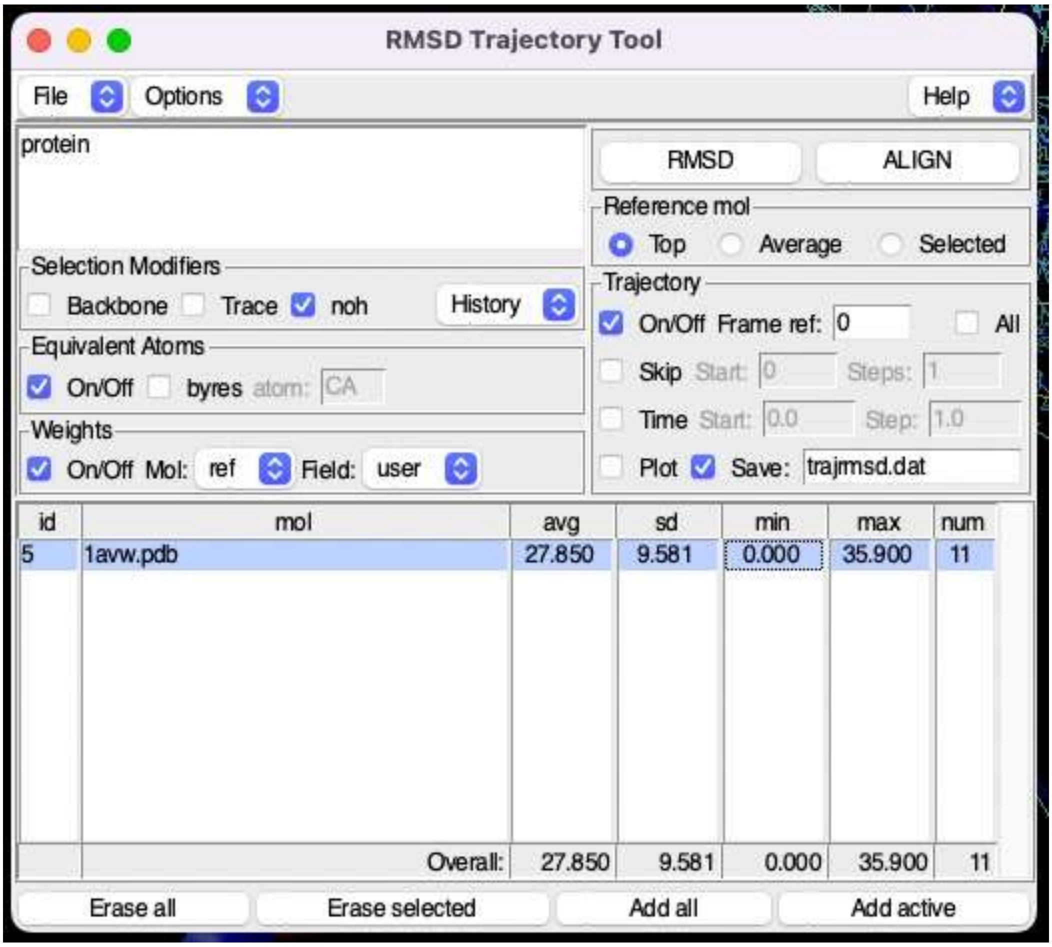
Load files for: 0: model.000.00.pdb

Filename:

Determine file type:
Automatically

ALGORTHIMS IN BIO INFORMATICS





protein

Selection Modifiers

□ Backbone □ Trace ☑ noh

Equivalent Atoms

☑ On/Off □ byres atom: CA

Weights

☑ On/Off Mol: ref Field: user

Reference mol

● Top ○ Average ○ Selected

Trajectory

☑ On/Off Frame ref: 0 □ All

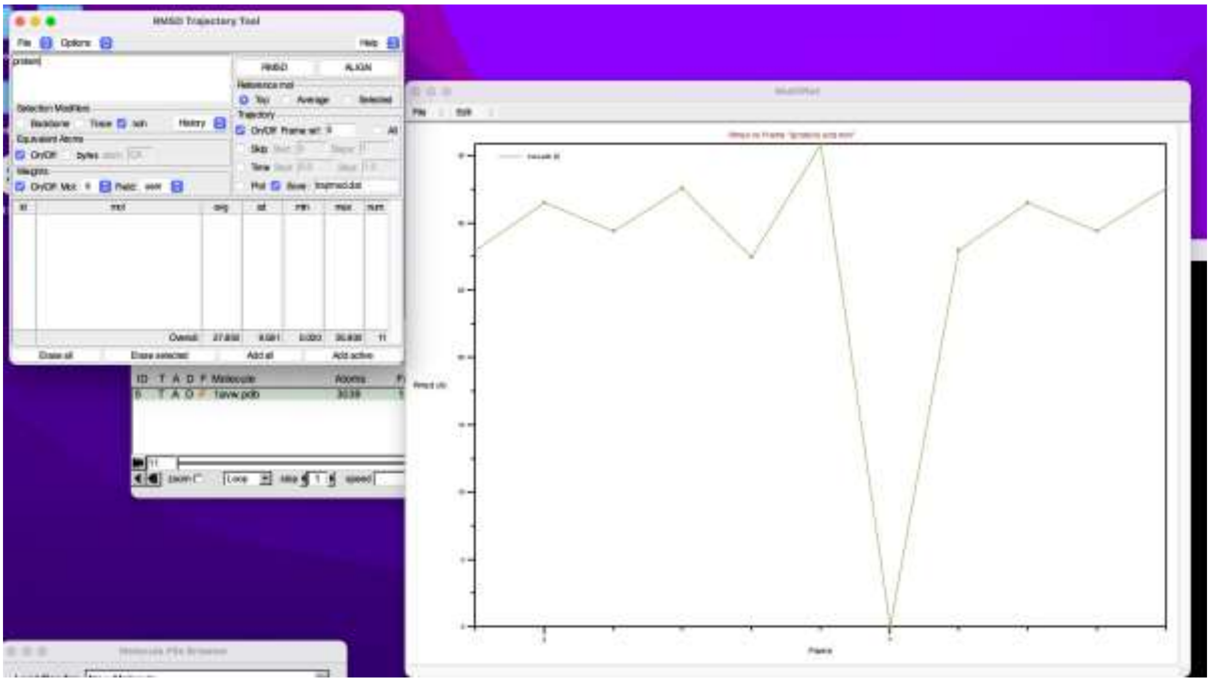
□ Skip Start: 0 Steps: 1

□ Time Start: 0.0 Step: 1.0

□ Plot ☑ Save: trajrmsd.dat

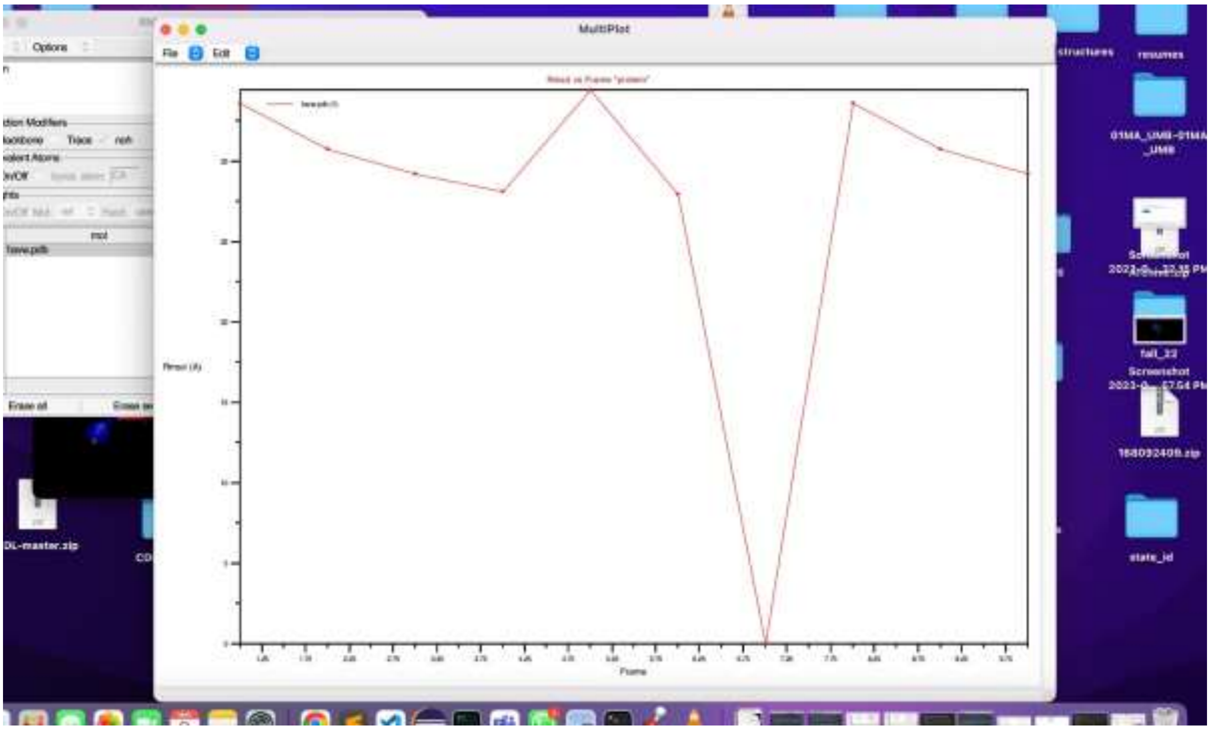
id	mol	avg	sd	min	max	num
5	1avw.pdb	27.850	9.581	0.000	35.900	11
	Overall:	27.850	9.581	0.000	35.900	11

ALGORTHIMS IN BIO INFORMATICS



```
 id mol                         avg       sd       min       max    num
1 1avw.pdb                   27.787    10.023     0.000    34.048    10
```

c. Now that you’ve done that, look at the native complex vs. the 10 complexes. Which one has the smallest RMSD to the native complex
 the smallest one is 7 and highest one is 4.


MultiPlot
File
Edit
Options
Rmsd (Å)
Frame
Erase all