

# MLBA [BIO542]

## Assignment 1

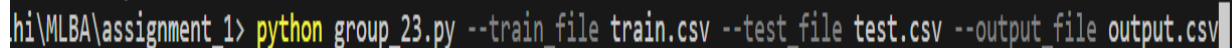
Group No - 23

Group Members - Shyama Goel [2021492]  
Nitin [2020092]  
Vibhu Jain [2020151]

**How to Run the file :**

**Command to be used :**

```
python group_23.py --train_file train.csv --test_file test.csv --output_file output.csv
```

A screenshot of a terminal window with a dark background. The prompt is 'hi\MLBA\assignment\_1>'. The command entered is 'python group\_23.py --train\_file train.csv --test\_file test.csv --output\_file output.csv'. The output of the command is not visible.

```
hi\MLBA\assignment_1> python group_23.py --train_file train.csv --test_file test.csv --output_file output.csv
```

Here we used **train.csv** as for training dataset , **test.csv** as for testing dataset and we will get the output of our test dataset in **output.csv** file .

Since we are using epochs and deep learning model , it will take time to train the data .

**Description of the code :**

This code is designed for protein sequence classification using deep learning techniques. The primary steps and functionalities are outlined below:

### **1.Importing Libraries :**

The code starts by importing necessary libraries. These include NumPy for numerical operations, argparse for handling command-line arguments, pandas for data manipulation, and specific modules from scikit-learn and TensorFlow for machine learning and deep learning tasks.

### **2. Model Building with our\_model Function:**

The core of the code lies in the `our_model` function, responsible for building, training, and making predictions with the deep learning model.

Here's a breakdown of the model architecture:

1. **Embedding Layer:** Converts protein sequences into numerical vectors, facilitating further processing.
2. **Bidirectional LSTM Layers:** Processes sequential data bidirectionally, capturing dependencies in both forward and backward directions
3. **Dense Layers with Activation Functions:** Fully connected layers with ReLU and Sigmoid activation functions for feature extraction and mapping.
4. **Dropout Layer:** Mitigates overfitting by randomly dropping a fraction of neurons during training.
5. **Output Layer:** A single neuron with a Sigmoid activation function for binary classification (predicting 0 or 1).

#### **Model Compilation and Training:**

- The model is compiled using the Adam optimizer and mean squared error loss function.
- Training is performed on the provided training data (X\_train and y\_train) for one epoch with a batch size of 64.

#### **Prediction:**

- The trained model is used to make predictions on the test data (X\_test).
- A threshold of 0.5 is applied to convert continuous predictions to binary labels (1 or 0).

### 3. **parse\_arguments() :**

The parse\_arguments function utilizes the argparse module to parse command-line arguments. It expects the paths to the training data, test data, and the output file where predictions will be saved.

### 4. **Main\_function():**

- Command-line arguments are parsed.
- Training and test data are loaded from CSV files.
- The training data is preprocessed using tokenization and padding.
- The our\_model function is called to make predictions on the test data.
- Predicted labels are converted back to their original form using label encoding.
- A DataFrame is created for the submission file with columns 'ID' and 'Label'.
- The submission file is saved to the specified output path.

This model is chosen for protein sequence classification due to its effectiveness in handling sequential data. It utilizes Bidirectional LSTM layers for capturing sequence patterns, an Embedding layer for numerical conversion of sequences, and Dropout for regularization. The Sigmoid activation in the output layer suits binary classification. The model is compiled with Adam optimizer, Mean Squared Error loss, and employs tokenization with padding for consistent input lengths. The simplicity of Keras facilitates rapid implementation and experimentation. Overall, the architecture is tailored for the unique characteristics of protein sequences and binary classification tasks.