PROTEION FUNCTION CLASSIFIER :

SYNOPSIS:

- **ABOUT THIS PROJECT**
- **PREPROCESSING DATA**
- **LOAD THE DATA AND GETTING THE SHAPE OF THE DATA**
- **TESTING TRAINING**
- **UNDERSTANDING SHAPES**
- **SEQUENTIAL MODEL**
- **FUNCTIONAL MODEL**

ABOUT THIS PROJECT :

Proteins play important roles in living organisms, and their function is directly linked with their structure. Due to the growing gap between the number of proteins being discovered and their functional characterization in particular as a result of experimental limitations, reliable prediction of protein function through computational means has become crucial.

In this we are doing fours steps

- processing the data,
- loading the data,
- testing and
- training the data and finally we will get our model.

PREPROCESSING DATA :

In this we are going to create a protein function classifier by using ML in this we need packages or regular regression, and also we have and glob packages for some implications.

first, we want to do define our file disc. we are our file in the data scrapes folder.

So we use to scrape, in we see Linux for../data scrapes and for windows, **\datascrapes.Os.Path.join** this code we can run our code in Google platforms for free.

Then import data and time reason for this which and when the file is created. we initialize the num_proteins as 0.

we create fasta file1 coz in this we have many **fasta** files.

let’s go for loop with fasta file and each line by line we should add some stuff. now we create protein function annot files same like us in fastafile then and print has a function we will get the separate protein I’d for 10 proteins.

LOAD THE DATA AND GETTING THE SHAPE OF THE DATA :

In this, we are changing the data into another format. we need NumPy, os, KERAS sequence, and JSON packages. first, our has function will going to load by JSON .we will get all proteins list which exhibits the ATP binding behavior. we will set the sequence size .we initiating some values and we will get proteins sequence of lines by function seq of indices. we can change the seq of indices. we can change the seq of lines also and finally print the label with sequence.

TESTING TRAINING :

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. In our Protein function classifier Testing and training is only need for large predictions and this is done by based on the size of the dataset here we have size is 500 we will split into two percentage they are 66% and 33%. we will get our x_shape as (7,500) and y_shape as (7,). Randomize our dataset by repeating the testing and training. for eg (6,2,5…)and print shapes again we will get, (5,500) as train shape (5,)as train shape.

UNDERSTANDING SHAPES :

In this we have initially 5 data points, then we give 500 vectors to each datapoint,

>>i.e [. . . . . 5]

Then we give 500 vectors to 5 data points,

>>[[. . . . . .500] [] [] [] []]

>>Now the shape is (5,500).

Now we have to convert our categorical values to numerical values by “one-hot representation”.It means in our datapoints should contain,

[0 0 0 0 0 1 0 0 0 0 0]

Like this manner, this is called one-hot representation or one-hot encoding. Because all machine learning algorithms can’t accept the categorical values. so we would use this. Now in our Protein function classifier, our shape is (5,500) we will convert our 11 into one hot representation then it will become [. . . . . . . .23] in this we will expand our first dot of the first data point in this each of 500 contains 23 vectors then we will get final shape (5,500,23).

SEQUENTIAL MODEL :

First of al we import KERAS layers they are ,

EMBEDDING LAYER:

It order to use words for natural language processing or machine learning tasks, it is necessary to first map them onto a continuous vector space, thus creating word vectors or word embeddings. The Keras Embedding layer is useful for constructing such word vectors.

>>input_dim : the vocabulary size.

>>input_dim: int > 0. Size of the vocabulary

>>output_dim: int >= 0. Dimension of the dense embedding.

>>input_length: Length of input sequences

FLATTEN LAYER :

A flatten operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension.

DENSE LAYER :

Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

ACTIVATION LAYER :

The activation layer in keras is equivalent to a dense layer with the same activation passed as an argument.

SGD :

Keras SGD Optimizer (Stochastic Gradient Descent) SGD optimizer uses gradient descent along with momentum. In this type of optimizer, a subset of batches is used for gradient calculation. Syntax of SGD in Keras. In this first import layers called embedding, flatten,dense,and activation and our model is sequential model and our optimizer is SGD.then we create a sequential model and add layers first we add embedding layer for to know about the size,dim, length of our Dataset.Here our size is 23,dim is 10 and length f seq is 500.we will get outut as one layer that is embedding in out shape we will get (none,500,10) none means batchsize we will give any value to batch size we will get none.then we go for another layer that is dense layer.and finaly we add flatten layer coz to prevent the collapse between layers then compare the model and fit the model.

FUNCTIONAL MODEL :

Now finally we will change our model from a sequential to a functional model. first, we will give input max size i.e 500. after that, we will add an embedding layer with the same data we don't add input size coz we will pass input to the embedding layer. now we can create a new layer called flatten layer and passed with embedding layer and add a dense layer in 25and 1. Then finally we fit our model.