**Overview:**

Using TensorFlow and Keras, we are going to make and optimize a neural network model for a medical data. The data is extracted via a Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins: <https://archive.ics.uci.edu/ml/datasets/Ecoli>

**Preprocessing:**

Using Pandas and NumPy our model reads and preprocess the data. It splittes the data based on white spaces, and adding the columns names. Then it filtered them based om just two classes, as mentioned in the assignment. After that the model needed to convert the classes to numeric values (0,1) to be readable by the pc. So, we though to implement target/mean encoding on the first column "sequence name" but at the end we chose to drop out this column because it has no effect on the classes/outputs. In other words, this column is just a collection of unique ID of each sample/row. After splitting the data into x and y, the next step is dealing with our unbalanced data. Using undersampling is not a good idea here because we have a small dataset upsamling is better approach. However, a combination of the two works best. For this purpose, we used a library called SMOTETomek.

**Architecture and parameter tuning:**

In general, we tried to reduce the complexity of our network as much as possible with taking in consideration the accuracy, loss, and efficiency. Our input layer has neurons as many as the features of our data. Some people add one more neuron to the input layer for bias but it makes almost no difference in our case. We also choose to go with just one hidden layer to avoid overfitting, make it easier for optimization, and compute more efficiently. It has 9 neurons because the hidden neurons are recommended to be between the size of the input neurons and the output layer. The chosen activation function in this layer is Relu. It is preferred in hidden layer, and works fine for a variety of the problems. Another reasons to choose it is because it is simple to implement and efficient computationally. It overcomes the problems of other activation functions like exploding and vanishing gradients. Regarding the output layer, we choose one neuron as an output because of our problem is binary. Sigmoid is the used activation function in this layer because it is simple to implement and works fine for this kind of problems. Now we will move to talk about backpropagating stage. Here we have two approaches to optimize the network. The first one is by using Stochastic gradient descent as an optimizer and Mean Squared Error as a loss function. While the second one is by using the optimizer Adam together with cross entropy as a loss function. When choosing the optimizer in the backpropagating stage, we need to consider the loss function given in the output layer because there is correlation between them. Both MSE and cross entropy are compatible with sigmoid function and good with binary classification. But MSE is simpler and gives lower loss for our model. However, it shows wider variance locally. We mean that cross entropy show more stability across epochs er can see the figure1 and 2 shows. Regarding the optimizers, Adam is the most popular one in deep learning nowadays because it shows that it works well with a wide range of problems. However, some experts think that Stochastic gradient descent performs better than Adam in generalizing, but Adam is fastest. For our data, SGD gives much higher accuracy than Adam does after just 15 epochs. While Adam gives good result after 100 epochs, please have a look on *figure1* and *2*. As you know the lower epochs and less complexity the more reliable model is. Therefore, we prefer the first approach for optimizing the model.

Chart, line chart

Description automatically generated

*Figure1 (approach* 1*)*

*Chart, line chart

Description automatically generated*

*Figure2 (approach* 2*)*