CM3

Here LSTM is used as an improved model. The number of epochs was set to 100 because I believe the network has converged at this stage. Categorical Cross Entropy is the loss function I use for training. The batch size is set to 32, and the training method is defined as the "adam" optimizer, which is a common form of gradient descent that automatically tunes itself and produces good results.

Network Architecture:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 32)                5888
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 22
=================================================================
Total params: 6,240
Trainable params: 6,240
Non-trainable params: 0
_________________________________________________________________
None
```

Optimizers:

Adam Optimizer learns the learning rates itself. It is generally regarded as being fairly robust to the choice of hyper parameters through the learning rate.

Activation Function:

Relu – It is the activation function that improves the network by speeding up training, the computational steps of relu is easy and it does not activate all the neurons at the same time

Softmax - Softmax is implemented at the output layer as an activation function. The softmax layer must have the same number of nodes as the output layer. In a multi-class problem, it assigns decimal probability to each class. Since SoftMax is heavily used in the output layer of multiclass classification, it has been used in the output layer in this problem.

Regularization methods:
Here in this architecture the dropout is not used as there are no large weights in the network as the sign of complexity the over fit the training data. Probabilistically dropping out the node is the simplest and most effective method

Why LSTM is used?

The recurrent layer of all RNNs has feedback loops. This allows them to keep information in 'memory' for a long time. However, training typical RNNs to handle issues that involve learning long-term temporal dependencies can be difficult. This is due to the fact that the loss function's gradient decays exponentially with time (called the vanishing gradient problem).

LSTM networks are a sort of RNN that employs a combination of special and standard units. A 'memory cell' is included in LSTM units, which can store data for lengthy periods of time. When information enters the memory, it is output, and it is forgotten, it is controlled by a set of gates.