**Q) 5) a):**

The hyperparameters for my final neural network are:

**No of Hidden Layers = 1, No of hidden nodes = 10, Learning Rate = 0.1, no of iterations/epochs = 1000, solver = ‘adam’, activation function = ‘relu’**

To make the algorithm simple, I used rectangular weight matrix i.e. the number of nodes in all hidden layers is same

I used the following grid for the Grid Search method to select the best hyperparameters:

1. hidden = [1,2,5]
2. nodes = [2,5,10]
3. lr =[0.1,1]
4. af =['tanh', 'relu']
5. solver =['sgd','adam']

I tried every combination of the above (72 combinations) and calculated the cross-validation error. Then, I picked the hyperparameters which gave the least cross validation error.

I applied the best model on the test dataset and got **77** percent accuracy

**While doing the search, I got a few convergence warnings which say that the no of iterations = 1000 is not enough for the model to converge. This may be due to the 0.1 learning rate**

**Q) 5) b):**

The concentration bound obtained at 95% confidence interval using Hoeffding bound is **0.096**

**Q) 5) c):**

I am getting different results (accuracy) every time I run my neural network because of

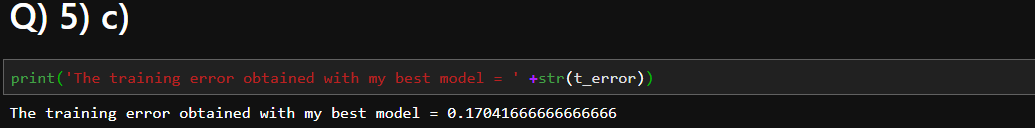
1. Random initialization of weights
2. Randomness in the training data (I set shuffle = True in the MLPClassifier)
3. Due to the adaptive Learning Rate (I set Learning Rate = ‘adaptive’ in the MLP Classifier)
4. Multiple Local Minima (in case of Non-Convex cost functions)

**Q) 5) d):**

We can say whether a model is overfitting or not by checking the Training error.

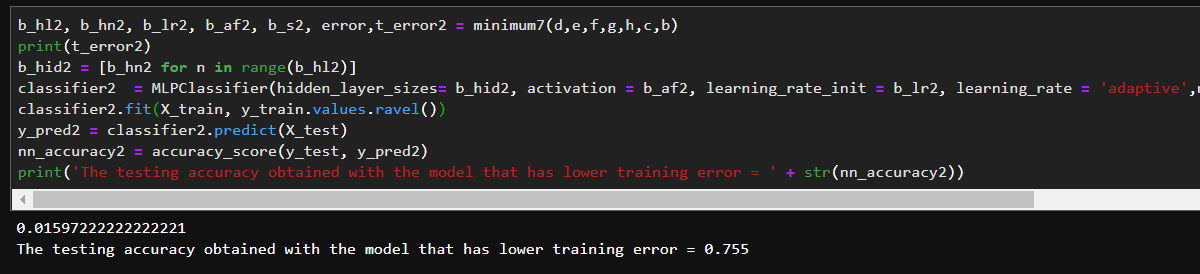
I saved the training error of all the models I built using the cross\_validate function of the Sklearn.model\_selection. To get the training error, I set the return\_train\_score parameter of cross\_validate to True.

For the model with the least validation error (the model that I picked as the best) has training error of **0.17.**



Since, the training error is not zero, I can say that my model is not overfitting.

I checked the testing accuracy of the model that has low training error:



I got lesser testing accuracy of 0.755 with the model that has low training error of 0.016.

**Note: I used random\_state =100 in the MLP classifier in the code to check the testing accuracies of my best model and the model that has low training error of 0.016**