**1. Using the data synthesis R script provided by the instructor as part of the week 11 assignment instructions, produce datasets of the following sizes, and fit deep learning models with the configurations shown below. Associated with each model, record the following performance characteristics: training error, validation (i.e., holdout set) error, time of execution. Use an appropriate activation function.**

| Data size | Configuration | Training error | Validation error | Time of execution |
|---|---|---|---|---|
| 1000 | 1 hidden layer 4 nodes | 0.447662 | 0.386414 | 3.975 |
| 10000 | 1 hidden layer 4 nodes | 0.439978 | 0.421220 | 2.765 |
| 100000 | 1 hidden layer 4 nodes | 0.067075 | 0.066131 | 9.446270 |
| 1000 | 2 hidden layers of 4 nodes each | 0.031297 | 0.028314 | 8.4929 |
| 10000 | 2 hidden layers of 4 nodes each | 0.010647 | 0.009515 | 74.007096 |
| 100000 | 2 hidden layers of 4 nodes each | 0.005884 | 0.004808 | 71.358379 |

**2. Based on the results, which model do you consider as superior, among the deep learning models fit?**

The 2 hidden layers with 4 nodes configuration proves to be the most effective solution especially when dealing with larger datasets. Its execution times remain elevated but this configuration produces consistently lower training and validation errors regardless of data size.

The 2 hidden layer configuration with 100,000 samples reaches validation error of 0.004808 while the single hidden layer model achieves 0.066131 - representing a 90% improvement. The double-hidden layer architecture exhibits superior learning potential since it achieves substantial validation error decreases when data size increases from 1,000 to 100,000 samples (beginning at 0.028314 and ending at 0.004808). By contrast, the single layer network experiences a more limited improvement in error (0.386414 becomes 0.066131). The higher computational expense of the 2-layer architecture remains acceptable because its improved accuracy benefits large datasets particularly.

**3. Next, report the results (for the particular numbers of observations) from applying xgboost (week 11 – provide the relevant results here in a table). Comparing the results from XGBoost and deep learning models fit, which model would you say is superior to others? What is the basis for your judgment?**

| Method Used | Dataset Size | Testing-set Predictive Performance | Time taken for the model to be fit |
|---|---|---|---|
| XGBoost in Python via scikit-learn and 5 fold CV | 1000 | 0.9490 | 0.5 |
| | 10000 | 0.9746 | 0.81 |
| | 100000 | 0.9871 | 5.22 |

In tests across all metrics XGBoost delivered superior results than the deep learning models. The deep learning models present error rates as their main output but XGBoost provides predictive performance scores of 0.9490, 0.9746, and 0.9871 for 1,000, 10,000, and 100,000 samples respectively which indicates accuracy rates above 94% in all cases. XGBoost requires only 5.22 seconds to fit 100,000 samples whereas the 2-layer neural network takes 71.36 seconds. The superior choice for this problem becomes XGBoost because it delivers better predictive results and executes 14 times faster on large datasets.