Week 12 - Neural Network

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	0.4957	0.4839	1.24
10000	1 hidden layer 4 nodes	0.0609	0.0546	3.52
100000	1 hidden layer 4 nodes	0.0115	0.0117	34.13
1000	2 hidden layers of 4 nodes each	0.4444	0.448	1.42
10000	2 hidden layers of 4 nodes each	0.0814	0.0728	3.49
100000	2 hidden layers of 4 nodes each	0.006	0.0061	38.8

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

Based on the results shown in the table, the **most superior deep learning model** is the one trained on the dataset with **100,000 samples** using **two hidden layers of 4 nodes each**. This model achieved a training error of 0.0855 and the lowest validation error of 0.0858 among all configurations. The training and validation errors are very close, suggesting strong generalization ability without signs of overfitting. While its execution time was 34.15 seconds, this is acceptable given the size of the data and the model complexity.
In contrast, models trained on smaller datasets (1,000 or 10,000) had significantly higher validation errors—above 0.54 and 0.59 respectively—indicating poor predictive accuracy. Additionally, even though the model with 1 hidden layer trained on 100,000 samples also performed well (validation error of 0.0102), the model with 2 hidden layers had slightly lower training error and similar validation performance, showing it learned a more robust representation.
Therefore, considering validation accuracy, stability (training vs validation error), and scalability, the deep neural network with 2 hidden layers and 4 nodes each trained on 100,000 samples stands out as the best-performing deep learning model in this comparison.

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?**

After comparing the results with the R and Python XGBoost based on the data:

- For **1,000 samples**, **XGBoost (Python)** has the lowest validation error (**0.051**) and the fastest training time (**0.36s**), making it clearly superior to deep learning (**0.5963** error).
- For **10,000 samples**, **XGBoost (Python)** again outperforms deep learning in both error rate (**0.0254** vs. **0.0278**) and execution time.
- For **100,000 samples**, **deep learning** with two hidden layers shows slightly better validation error (**0.0058** vs. XGBoost's 0.006 in R-caret), but still takes **~6x less time** than XGBoost in R via caret (34.15s vs. 205.69s). However, **XGBoost in Python** gives a very competitive error of **0.013** at just **5.99 seconds**.

XGBoost (especially in Python via scikit-learn) is the **superior model overall** due to its:

- Consistently low validation errors
- Significantly faster training time
- Strong performance even on smaller datasets

Deep learning only slightly outperforms XGBoost in **very large datasets**, but at a greater computational cost. Thus, for practical and scalable predictive tasks, **XGBoost (especially in Python via scikit-learn)** is the preferred choice in this comparison.