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	0.5136	0.5098	2.09
10000	1 hidden layer 4 nodes	0.073	0.0699	4.9
100000	1 hidden layer 4 nodes	0.0111	0.0104	33.07
1000	2 hidden layers of 4 nodes each	0.4695	0.4583	2.19
10000	2 hidden layers of 4 nodes each	0.0242	0.0244	5.2
100000	2 hidden layers of 4 nodes each	0.006	0.0065	44.71

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

Based on the results presented in the table, the superior deep learning model is the one trained on the 100,000-sample dataset using two hidden layers of four nodes each. This

model achieved the lowest training error (0.006) and lowest validation error (0.0065) among all configurations. Additionally, the training and validation errors are nearly identical, indicating excellent generalization and minimal overfitting.

While this model had the longest execution time at 44.71 seconds, the trade-off is justified given its accuracy and stability on a large dataset. In contrast, models trained on smaller datasets (1,000 and 10,000) produced significantly higher validation errors, suggesting weaker performance and poorer generalization. Similarly, models with only one hidden layer, even when trained on large data, slightly underperformed in validation error compared to the two-layer configuration.

Therefore, the **deep learning model with 2 hidden layers of 4 nodes** each trained on 100,000 observations is considered the most effective, balancing accuracy and model robustness despite a longer training time.

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?**

Predictive Performance: Across all dataset sizes, XGBoost consistently achieves lower validation errors than deep learning models. In particular, XGBoost in R using the basic xgboost() function shows near-perfect performance for smaller datasets (even reporting a perfect 1.0 on 1,000 rows, which may be overfitting), and XGBoost in Python via scikit-learn performs strongly and efficiently across the board.

Execution Time: XGBoost (especially via the raw xgboost() function in R) offers superior speed, with execution times dramatically lower than deep learning for the same dataset

sizes. Even at 100,000 rows, it finishes in under a second, while deep learning takes over 44 seconds.

Deep Learning: The best deep learning result comes from the 2-hidden-layer model trained on 100,000 samples, which achieved a validation error of 0.0065 — very close to the best XGBoost score of 0.006 (R caret). However, this came at a much lower cost in time (44.71s vs. 254.36s), suggesting some merit to neural models when execution time is not a constraint.

XGBoost is the superior model overall in terms of predictive performance and training efficiency. While deep learning performs competitively at larger scales, XGBoost remains more robust and efficient.