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	0.4426	0.4545	6.53
10000	1 hidden layer 4 nodes	0.0294	0.0250	16.46
100000	1 hidden layer 4 nodes	0.0070	0.0069	115.71
1000	2 hidden layers of 4 nodes each	0.1798	0.2111	6.25
10000	2 hidden layers of 4 nodes each	0.0212	0.0165	16.53
100000	2 hidden layers of 4 nodes each	0.0045	0.0044	116.29

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

Based on the results, the deep learning model with **2 hidden layers of 4 nodes each** consistently outperformed the model with **1 hidden layer of 4 nodes** across all dataset sizes when validation error was considered. Validation error, being the key measure of a model's ability to generalize to unseen data, was lower for the 2-layer model at every data size. For instance, at the largest dataset size of 100,000, the 2-layer model achieved a validation error of **0.0044**, whereas the 1-layer model recorded **0.0070**. In addition to better generalization, the 2-layer model also achieved slightly lower training errors, indicating better learning of the underlying patterns in the data. The differences in training errors were small but consistent, showing that the additional layer provided a slight advantage in representational capacity without leading to overfitting.

While the 2-layer model required slightly more time to train particularly on larger datasets (e.g., 116.29 seconds compared to 115.71 seconds for 100,000 samples) this additional time was modest and acceptable given the performance gains. Overall, the model with **2 hidden layers**

was considered superior due to its better generalization, slightly improved training accuracy, and reasonable execution 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?

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	0.9500	2.58
	10000	0.9638	0.27
	100000	0.9702	1.95
XGBoost in R – direct use of xgboost() with simple cross-validation	1000	0.9560	0.57
	10000	0.9787	1.71
	100000	0.9864	6.31
XGBoost in R – via caret, with 5-fold CV simple cross-validation	1000	0.9410	1.52
	10000	0.9649	2.77
	100000	0.9699	12.22

Based on the results, XGBoost emerges as the superior model overall due to its strong balance between predictive performance and computational efficiency. Among the implementations, XGBoost using the direct xgboost() function in R consistently achieves the highest testing accuracy, reaching 0.9864 on the largest dataset (100,000 records), while also maintaining reasonable training times compared to deep learning models. Although deep learning with two hidden layers achieves slightly lower validation errors (as low as 0.0044), it requires significantly more time to train over 116 seconds at the largest scale compared to just 6.31 seconds for the best XGBoost implementation. Overall, XGBoost is more efficient, scalable, and accurate enough for most tasks, making it the preferred choice in this comparison.