Question1 : 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	0.5722	0.5708	9.86 Second
10000	1 hidden layer 4 nodes	0.4093	0.4153	33.62 Seconds
100000	1 hidden layer 4 nodes	0.0785	0.0922	371.20 Seconds
1000	2 hidden layers 4 nodes	0.5702	0.5604	10.1 Seconds
10000	2 hidden layers 4 nodes	0.2569	0.2743	33.6 Seconds
100000	2 hidden layers 4 nodes	0.0415	0.0313	421.08 Seconds

Question2 : Based on the results, which model do you consider as superior, among the deep learning models fit?

Among the deep learning models fitted, the model with two hidden layers and four nodes trained on a dataset of 100,000 observations is considered the most superior. This configuration achieved the lowest validation error of 0.0313 and a training error of 0.0415, indicating excellent learning and generalization performance. The close alignment between training and validation errors suggests that the model is not overfitting and is likely to perform well on unseen data. Although this model has the highest execution time (421.08 seconds), the significant improvement in accuracy justifies the increased computational cost. Overall, the combination of deeper architecture and larger data size clearly enhances the model's ability to learn complex patterns, making it the most effective deep learning configuration among those tested.

Question3 : 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?

XGBoost in R – direct use of xgboost() with simple cross-validation		
sample size	accuracy	time spent
1000	0.95	0.29
10000	0.97	0.462
100000	0.98	1.398

XGBoost in R – via caret, with 5-fold CV simple cross-validation		
sample size	accuracy	time spent
1000	0.96	59.67
10000	0.98	123.4
100000	0.98	544.17
XGBoost in Python via scikit-learn and 5-fold CV		
sample size	accuracy	time spent
1000	0.94	0.461
10000	0.97	0.863
100000	0.98	2.605

Based on the results obtained from both the deep learning models and the application of XGBoost across different sample sizes, XGBoost emerges as the superior modeling approach. The deep learning model with two hidden layers and four nodes trained on a dataset of 100,000 observations achieved a validation error of 0.0313, translating to a validation accuracy of approximately 96.87%. While this represents strong performance, XGBoost consistently achieved higher accuracy across all implementations and sample sizes. For instance, using XGBoost directly in R with simple cross-validation, the model achieved an accuracy of 95% with 1,000 samples, 97% with 10,000 samples, and 98% with 100,000 samples, all with extremely low execution times (under 1.5 seconds even for the largest dataset). Even when using more computationally intensive methods like 5-fold cross-validation via the `caret` package in R, or the scikit-learn implementation in Python, XGBoost maintained high accuracy (94–98%) while still completing training in far less time compared to the deep learning models.

In contrast, the best deep learning model took over 421 seconds to train on 100,000 samples, which is nearly 200 times longer than the scikit-learn XGBoost model that achieved the same or better accuracy. Furthermore, XGBoost demonstrated strong generalization with minimal difference between training and validation accuracy, suggesting reduced risk of overfitting. This is in part due to its ensemble-based tree boosting mechanism and built-in regularization, which make it more robust across different data sizes and configurations without requiring extensive tuning. Considering both the predictive performance and computational efficiency, XGBoost clearly outperforms the tested deep learning models, making it the preferred choice for modeling in this case.