Deep Learning Model Evaluation

Dataset Size	Model Config	Training Error	Validation Error	Time Taken (s)
1000	1-layer	0.0950	0.0900	6.49
1000	2-layer	0.0612	0.0700	8.18
10000	1-layer	0.0034	0.0005	26.01
10000	2-layer	0.0038	0.0025	25.50
100000	1-layer	0.0014	0.0010	176.49
100000	2-layer	0.0018	0.0018	193.81

Q1: Which deep learning model configuration do you consider superior?

The 2-layer at 100000 provides very low train and validation error (0.0018 and 0.0018 respectively), though slightly higher than the 1-layer at 100000. However, given the negligible difference and the fact that 1-layer does the same with less computation (176.49s compared to 193.81s), the 1-layer at 100000 model is potentially better for efficiency without any loss in accuracy.

XGBoost Comparison Table

Method used	Dataset Size	Testing-set predictive performance	Time taken for the model to be fit
XGBoost in Python via scikit-learn (5-fold CV)	1000	0.9412	2.05 sec
	10000	0.9727	1.35 sec
	100000	0.9861	4.67 sec
XGBoost in R (direct use of xgboost())	1000	0.0560	0.017 sec
	10000	0.0615	0.019 sec
	100000	0.0698	0.080 sec
XGBoost in R (via caret, with 5-fold CV)	1000	1.0000	29.62 sec
	10000	0.9946	55.75 sec
	100000	0.9954	303.89 sec

Q2: Comparing Deep Learning and XGBoost, which model is superior and why?

For both sizes of datasets, XGBoost outperforms deep learning in predictability (with the highest R caret-based accuracy >0.99) and execution speed. Even Python-based XGBoost yields >0.97 accuracy with very less training time (~4.67s for 100k records), whereas deep learning models are accurate but extremely time-consuming to train. XGBoost is thus better in this configuration as it is always accurate, fast, and interpretable for tabular data.