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	100	0.90	0.2253
	1000	0.95	0.291959
	10000	0.9739	0.838179
	100000	0.9868	4.16
	1000000	0.991809	43.516355
	10000000		
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.85	0.062
	1000	0.925	0.062
	10000	0.9465	0.12
	100000	0.9468	0.7844
	1000000	0.9505	7.356
	10000000	0.95130	106.838
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.95	1.6
	1000	0.95	1.98
	10000	0.97	3.14
	100000	0.9828	18.18

Method used	Dataset size	Testing-set predictive performance	Time taken for the model to be fit
	1000000	0.9866	148.41
	10000000	0.987	1321.18

2.

The best implementation approach for XGBoost comes from using Python scikit-learn with 5-fold cross-validation. The predictive performance of this approach remains superior throughout all dataset sizes because it produces the highest accuracy scores (ranging from 0.90 to 0.991809). The slightly extended processing time of this approach makes sense because it delivers superior performance benefits for large datasets.

The Python version demonstrates superior scalability because its performance improves dramatically when the dataset size increases (from 0.90 with 100 observations to 0.991809 with 1 million observations). The method demonstrates exceptional value for practical applications with extensive datasets because it delivers superior prediction accuracy although it needs additional computational resources than direct R implementation.