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	0.98	0.0322
	1000	0.9470	0.0868
	10000	0.9776	0.2227
	100000	0.9871	0.7634
	1000000	0.9919	7.8348
	10000000	0.9957	79.6787
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.85	0.20
	1000	0.9350	0.13
	10000	0.9705	0.17
	100000	0.9772	0.74
	1000000	0.9788	6.08
	10000000	0.9783	91.40
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.85	1.43
	1000	0.945	2.38
	10000	0.975	4.1
	100000	0.9823	15.39

Method used	Dataset size	Testing-set predictive performance	Time taken for the model to be fit
	1000000	0.9863	111.14
	10000000	0.987	1366.62

The Python implementation maintains superior predictive performance throughout all dataset sizes where it reaches accuracy scores of 0.9957 for 10 million records better than R implementation scores of 0.9783 and 0.987. The Python implementation shows its most reliable performance across all data sizes particularly when working with both small datasets of 100 records and large datasets with 10+ million records.

The Python implementation provides outstanding time-performance balance during computational efficiency assessments. The Python version of the analysis shows better data processing capabilities compared to R when working with large datasets. The Python implementation finishes processing 10 million records in 79.67 seconds while R with caret requires 1366.62 seconds leading to a 17 times faster execution. The superior scalability of Python implementation makes it more usable for large-scale applications which require both high performance and short processing times.