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	0.91	0.26
	1000	0.9490	0.41
	10000	0.9725	2.69
	100000	0.9869	4.11
	1000000	0.9927	42.41
	10000000		
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.9	0.57
	1000	0.96	2.91
	10000	0.976	3.8
	100000	0.9818	16.77
	1000000	0.9865	147.6
	10000000	0.9895	356.2
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.9	1.3
	1000	0.94	2.1
	10000	0.967	4.16
	100000	0.9831	14.67

Method used	Dataset size	Testing-set predictive performance	Time taken for the model to be fit
	1000000	0.9863	128.04
	10000000	0.987	1230.18

The XGBoost implementation through Python scikit-learn with 5-fold cross-validation proves to be the most efficient solution for both accuracy and speed performance when working with large-scale datasets. The predictive model achieves 0.9927 accuracy on 1,000,000 examples with training times that remain lower than those of R-based implementations. The model exhibits excellent scalability when dealing with larger datasets because it demonstrates acceptable computational time increases correspondingly to accuracy enhancements making Python/scikit-learn pipeline performance plots positive.

When using the `xgboost()` function directly in R the model achieves comparable accuracy to other implementations while requiring significantly longer processing times when working with datasets containing more than 1,000,000 observations. The R implementation of XGBoost takes 356.2 seconds to train at 10,000,000 examples which results in an accuracy of 0.9895 yet it requires eight times more processing time than the Python version (42.41 seconds at 1,000,000 examples). The tool falls short when used for time-critical or efficient computational applications despite demonstrating robust predictive functions.

The implementation of XGBoost through the `caret` package in R produces longer training times without improving accuracy levels. The training process of `caret` takes longer than both the R implementation and Python implementation for every dataset size. The use of `caret`-based pipelines with XGBoost should be avoided unless complete integration with other `caret`-based pipelines is necessary because it delivers poor accuracy-runtime performance. XGBoost through Python and scikit-learn stands as the preferred method for practical applications that handle big data or model development cycles since it delivers the best speed and prediction capabilities.