1.

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		
	1000		
	10000		
	100000		
	1000000		
	10000000		
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.8	0.576
	1000	0.92	0.201
	10000	0.96950	0.388
	100000	0.98535	1.52
	1000000	0.98852	19.101
	10000000	0.98625	96.53
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.8	17.39
	1000	0.9450	29.194
	10000	0.9755	50.80143
	100000	0.9901	219.52184
	1000000	0.9889	396.253
	10000000	0.9856	1142.26

2.

The direct XGBoost approach demonstrates superior performance to the caret implementation according to the provided data so I would advise using the direct XGBoost approach. XGBoost performs directly with results similar to caret but needs significantly less time to complete computations. The direct XGBoost method finished processing 10 million observations in 96.53 seconds while the caret implementation took 1142.26 seconds which amounts to a speedup of

12 times. The speed becomes essential when analyzing big datasets together with situations that demand model reevaluations.

The convenient all-in-one interface provided by caret together with automatic cross-validation consumes a large amount of computing resources. The performance gain from decreasing dataset sizes fails to justify the increased computation time because the predictive metrics eventually match (0.986-0.989) between the two methods as dataset sizes increase. The direct XGBoost implementation delivers the best performance in terms of predictive accuracy and computational speed for production model development and fast model creation.