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	0.85	0.01s
	1000	0.94	0.016s
	10000	0.9515	0.78s
	100000	0.95070	0.88s
	1000000	0.95047	6.5s
	10000000	0.965	69.5s
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.919	2.20
	1000	0.978	4.86
	10000	0.983	26.47
	100000	0.990	242.19
	1000000	0.996	987.52
	10000000	0.991	1956.62

Most production applications should utilize XGBoost through caret with 5-fold cross-validation according to the provided results. both approaches display different processing speeds with XGBoost through caret handling 10M samples in 69.5 seconds compared to 1956.62 seconds by direct XGBoost implementation but caret shows superior accuracy at all dataset sizes reaching 0.996 at 1M samples than XGBoost at 0.95047 In most practical applications where precision takes priority over processing speed the approximately 4-5% performance gain becomes critical.

The choice depends on particular use case needs. XGBoost in its direct form remains the best choice when real-time predictions are essential or when dealing with large datasets that need limited computational power. Most analytical tasks require superior model performance during training which occurs rarely so the extra computational expense of caret implementation proves worthwhile. Product environments can benefit from the standard interface of the caret wrapper which enables robust hyperparameter tuning and model evaluation thus ensuring critical reproducibility needs.