Results:

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	0.8875	0.5786 secs
	1000	0.9412	2.0458 secs
	10000	0.9727	1.3518 secs
	100000	0.9861	4.6678 secs
	1000000	0.9914	46.9417 secs
	10000000	0.991	470 secs
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.0000	0.0208 secs
	1000	0.0560	0.0176 secs
	10000	0.0615	0.0198 secs
	100000	0.0698	0.0806 secs
	1000000	0.0700	0.6055 secs
	10000000	0.0700	5.5000 secs
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	1.0000	19.29897 secs
	1000	1.0000	29.62123 secs
	10000	0.9946	55.74529 secs
	100000	0.9954	303.88965 secs
	1000000	0.99299	2456.75041 secs
	10000000	0.993	20000 secs

Based on the results, which approach to leveraging XGBoost would you recommend? Explain the rationale for your recommendation.

From the findings, I would suggest using XGBoost in Python through scikit-learn with 5-fold cross-validation for big data and using xgboost() in R directly for small to medium data.

The reason is:

On small datasets (100, 1000, 10000 rows), direct xgboost() in R is extremely fast (milliseconds) with excellent predictive performance (test error very low). It is the fastest method for smaller data without loss of accuracy.

Caret in R is extremely accurate (near 1.0) but incredibly slow even for medium-sized data. For instance, for 1 million rows, caret took well over 40 minutes and will take 5+ hours for 10 million rows, which is not practical at all for large-scale modeling.

Python XGBoost with scikit-learn is a decent compromise between predictive capability and speed. As the size of datasets increases above 1 million rows, Python performs much better than caret in R with acceptable training times (around 8 minutes for 10 million rows) and excellent accuracy (around 99.1%).

Accordingly: Small datasets (≤100,000 rows): Favor R with native xgboost().

Large data (≥1,000,000 rows): Use Python with scikit-learn XGBoost, since it processes large data faster in terms of time and stability.