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	0.8	3.2738
	1000	0.91	1.2084
	10000	0.963	1.5408
	100000	0.9786	3.0505
	1000000	0.9856	17.082
	10000000	Error	Error
XGBoost in R – direct use of xgboost() with simple cross-validation	100	0.9500	0.38
	1000	0.9250	0.42
	10000	0.9535	0.73
	100000	0.9519	4.59
	1000000	0.9506	19.26
	10000000	0.9509	318.03
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	0.7500	0.75
	1000	0.9200	0.57
	10000	0.9510	1.31
	100000	0.9479	3.02
	1000000	0.9512	20.76
	10000000	0.9504	414.88

The three methods demonstrated high predictive accuracy primarily when dealing with large dataset sizes. The accuracy levels of all methods exceeded 0.95 when dealing with datasets larger than 10,000 observations. Despite similar predictive accuracy results the

time to fit models between methods differed along with varying success rates when dealing with extremely big datasets.

When executing xgboost() directly from R it delivered the quickest and most efficient results especially when the dataset grew in size. A dataset containing 10 million rows required 318 seconds to be processed through direct xgboost() execution yet processing it through Caret-based API required 414 seconds. The Caret approach delivered high accuracy but created processing delays because of how it required model optimization combined with multiple cross-validation runs during execution.

When implementing XGBoost through scikit-learn on Python whenever datasets were restricted to 1 million observations or smaller it exhibited comparable processing times. The system stopped working on the 10 million rows dataset because of inadequate resources and insufficient storage capacity. The Python approach provides poor reliability when processing big datasets given modern computing limits even though it delivers exceptional speed for medium-scale datasets.

Standing alone, xgboost() implementation in R serves as the best option for managing extremely big datasets. The system shows both quick training times alongside accurate predictions and handles extensive data volumes effectively without generating errors. When working with datasets containing fewer than one million rows the Python or R-based approaches should work depending on user preference together with the unique computational constraints. The most effective solution for consistency speed and scalability emerges as direct xgboost() in R.