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	0.9200	1.40
	1000	0.9480	2.28
	10000	0.9772	3.07
	100000	0.9869	13119.65
	1000000	0.9915	23.09
	10000000	0.9936	117.95
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	0.9504	414.88

Tool-based predictions proved highly accurate because they functioned effectively with large data collection sizes. The testing-set accuracy exceeded 0.95 for all methods when the dataset exceed 10000 records. All implementation methods produced equivalent prediction results yet each showed unique model-time and data-stability measures with extensive datasets.

When processing big data sets the direct application of xgboost() through R proved to be the most efficient computing method along with being the most reliable option. The native R xgboost() function trained 10 million row data in three hundred eighteen seconds. The use of caret interface within R required 414 seconds for processing time since multiple model tuning methods and cross-validation steps added substantial overhead.

Scikit-learn implemented XGBoost with great speed and maximum accuracy when processing datasets containing up to 1 million rows in Python. The training process for the 100,000-row dataset required unusually long computation time of more than 13,000 seconds before completion. The 118-second completion time for training on 10 million rows indicates either a resource limitation or environmental system variability because training success occurred. The Python approach demonstrates quick processing

capabilities for datasets of medium sizes yet encounters reliability problems on specific system setups that handle large datasets.

R's direct xgboost() approach stands as the optimal method for big data because it achieves efficient system stability combined with accurate performance during quick computation times. The decision between methods 1, 2 and 3 depends on user comfort and system capacity as they remain equally effective for data ranges under 1 million rows. The direct xgboost() implementation within R maintains the most reliable solution for larger data processing because of its scalability alongside consistent performance and optimal runtime.