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	0.8700	0.38
	1000	0.9490	0.5
	10000	0.9746	0.81
	100000	0.9871	5.22
	1000000	0.9917	27.32
	10000000	0.9931	327.44
XGBoost in R – direct use of xgboost() with simple cross-validation	100	1	0.016
	1000	1	0.0207
	10000	0.989	0.1045
	100000	0.980	0.9992
	1000000	0.979	8.919
	10000000	0.983	50.2369
XGBoost in R – via caret, with 5-fold CV simple cross-validation	100	1	1.774
	1000	0.985	4.79
	10000	0.99450	25.759
	100000	0.99485	240.6272
	1000000	0.99756	1056.23
	10000000	0.99565	3512.6

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

Based on the results, I recommend using XGBoost in R – direct use of xgboost() with simple cross-validation. This method consistently provides high predictive performance across all dataset sizes while maintaining significantly lower model training times compared to the other approaches.

For instance, when working with a dataset size of 10 million, direct use of xgboost() in R required only around 50 seconds to fit the model, whereas XGBoost via caret in R took over 3500 seconds, and XGBoost via scikit-learn in Python took approximately 327 seconds. This clearly shows that the direct approach in R is much faster, especially as dataset size increases.

Accuracy levels remain competitive across all methods. Even with the faster training times, direct xgboost() in R still achieved predictive performances comparable to those of scikit-learn and caret. In fact, at almost all dataset sizes, the accuracy was very close to or above 98%, indicating that faster training did not come at the cost of model quality.

The caret approach, while useful for hyperparameter tuning and flexibility, introduces extra overhead, making it less suitable when speed and efficiency are priorities, especially with very large datasets. Similarly, Python's scikit-learn interface is easy to use but becomes slower compared to R's direct method as data size increases.

In conclusion, for balancing high accuracy and computational efficiency, the direct use of xgboost() in R is the best method among the three approaches evaluated.