| **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 | 0.86 | 0.34 |
| 1000 | 0.9470 | 0.27 |
| 10000 | 0.9764 | 0.81 |
| 100000 | 0.9873 | 4.05 |
| 1000000 | 0.9919 | 43.83 |
| 10000000 | 0.9932 | 419.97 |
| XGBoost in R – direct use of xgboost() with simple cross-validation | 100 | 0.95 | 0.142 |
| 1000 | 0.955 | 0.301 |
| 10000 | 0.976 | 0.599 |
| 100000 | 0.987 | 2.948 |
| 1000000 | 0.992 | 9.555 |
| 10000000 | 0.993 | 77.47 |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.9099 | 11.33 |
| 1000 | 0.9680 | 10.03 |
| 10000 | 0.9825 | 13.75 |
| 100000 | 0.99075 | 53.43 |
| 1000000 | 0.99233 | 635.531 |
| 10000000 |  |  |

The XGBoost in R - direct use of xgboost() with simple cross-validation approach stands as my recommended solution for the task. The implementation achieves outstanding predictive performance at 0.993 accuracy on the largest dataset through efficient computation times. The direct use of xgboost() in R required only 77.47 seconds to process 10 million data points whereas Python took 419.97 seconds and R's caret implementation failed to finish the same task. The direct use of xgboost() in R provides optimal results regarding prediction accuracy and computational speed which makes it the most suitable selection for practical applications that handle extensive datasets.

Implementation details together with overhead appear to explain these performance differences. Caret package in R boosts computation time through its comprehensive preprocessing and validation features which causes its execution times to run significantly longer than other options. The Python implementation through scikit-learn includes processing layers which provide helpful connectivity to other Python libraries though they create additional computational requirements. The direct xgboost() implementation in R functions at a low level near the C++ algorithm base without many abstraction layers which enables efficient computation and preserves accurate predictions. Your model performance together with computational efficiency makes this solution the most realistic choice for practical machine learning applications as your dataset expands.