| **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 | 0.90 | 0.08 |
| 1000 | 0.94 | 0.0464 |
| 10000 | 0.98 | 0.142963 |
| 100000 | 0.9886 | 0.753442 |
| 1000000 | 0.9922 | 8.827513 |
| 10000000 | 0.9931 | 83.182443 |
| XGBoost in R – direct use of xgboost() with simple cross-validation | 100 | 0.80 | 0.01 |
| 1000 | 0.925 | 0.03 |
| 10000 | 0.971 | 1.52 |
| 100000 | 0.9858 | 13.27 |
| 1000000 | 0.9922 | 126.31 |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.75 | 0.21 |
| 1000 | 0.96 | 0.13 |
| 10000 | 0.97 | 0.47 |
| 100000 | 0.9761 | 3.40 |
| 1000000 | 0.9793 | 33.46 |

The presented data indicates XGBoost in Python through scikit-learn with 5-fold CV should be used as the primary solution for most applications. The performance metrics demonstrate support for implementing XGBoost in Python through scikit-learn with 5-fold CV for most applications.

The Python implementation delivers the best predictive performance at every point of dataset size measurement. The accuracy rate reaches 0.90 when using 100 samples while increasing to 0.9931 when processing 10 million samples. Most machine learning applications consider this outstanding predictive capability as a necessary requirement for their successful operation.

The Python implementation shows outstanding computational performance when processing datasets of all sizes including smaller to medium-sized ones. The system requires 0.08 seconds to finish training 100 samples while keeping acceptable training times across different dataset sizes. The speed and accuracy balance of this approach makes it the most suitable solution when speed alone is not the top priority for small datasets.

The R implementations present several noteworthy performance trade-offs between each other. The direct implementation of xgboost() in R performs faster than Python on small datasets yet struggles to keep up with Python's speed when working with 1 million samples since it requires 126.31 seconds compared to Python's 83.18 seconds. The R version of caret achieves stable predictive results at lower levels compared to other methods.

The Python implementation demonstrates optimal accuracy and reasonable processing speed for extremely large datasets exceeding 10 million samples making it the most practical solution. The ability to scale up operations becomes essential in production environments because data volumes grow steadily with time.

XGBoost through scikit-learn in Python with 5-fold CV stands out as the most optimal solution because it achieves the best combination of predictive accuracy and computational speed and data size scalability. Its widespread use stems from being the best solution for practical machine learning tasks since datasets are expected to grow while model precision is essential.