|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |  |
|  | 1000 |  |  |
|  | 10000 |  |  |
|  | 100000 |  |  |
|  | 1000000 |  |  |
|  | 10000000 |  |  |
| XGBoost in R – direct use of xgboost() with simple cross-validation | 100 | Error Rate – 0.05 | 0.04s |
|  | 1000 | Error Rate – 0.08 | 0.02s |
|  | 10000 | Error Rate – 0.026 | 0.18s |
|  | 100000 | Error Rate – 0.0152 | 1.54s |
|  | 1000000 | Error Rate – 0.0114 | 13.16s |
|  | 10000000 | Error Rate – 0.0107 | 197.56s |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | Error Rate – 0 | 1.87 |
|  | 1000 | Error Rate – 0.05 | 3.86s |
|  | 10000 | Error Rate – 0.0115 | 18.83s |
|  | 100000 | Error Rate – 0.0096 | 90.63s |
|  | 1000000 | Error Rate – 0.0093 | 400.69s |
|  | 10000000 | Error Rate – 0.0089 | 1800.12s |

The XGBoost implementation through caret with 5-fold cross-validation should be used for practical applications based on performance metrics. The xgboost() approach by itself proves to be faster but the caret implementation produces better predictive results combined with reduced error rates at all dataset scales. The error rate decreases by 17% when using caret at 10-million observations compared to direct implementation which results in a 0.0089 error rate versus 0.0107.

The choice depends on particular project requirements together with priority factors. The direct use of xgboost() achieves good error rates with significantly faster processing times compared to standard implementation (197 seconds compared to 1800 seconds at the 10-million observation level). The sophisticated cross-validation approach of the caret implementation produces better model performance although it requires additional processing time which most production models can accept because accuracy is the main objective.