|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 0.85 | 0.14 |
|  | 1000 | 0.92 | 0.19 |
|  | 10000 | 0.9385 | 0.16 |
|  | 100000 | 0.9464 | 0.74 |
|  | 1000000 | 0.9503 | 8.16 |
|  | 10000000 | 0.9635 | 45.23 |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.85 | 25.52 |
|  | 1000 | 0.93 | 37.38 |
|  | 10000 | 0.982 | 63.72 |
|  | 100000 | 0.98935 | 263.59 |
|  | 1000000 | 0.975 | 690.25 |
|  | 10000000 | 0.992 | 965.23 |

The data suggests implementing direct XGBoost with simple cross-validation as a better solution instead of using the caret implementation. Direct implementation executes tasks much faster than the caret approach regardless of dataset size as indicated by its 0.74 second time versus 263.59 seconds when analyzing 100000 observations. The direct method delivers a satisfactory 0.9635 accuracy in 45.23 seconds for 10 million observations yet caret requires 965.23 seconds to achieve the same result. The efficiency advantage proves essential for production purposes as well as situations that require regular model retraining.

The predictive accuracy advantage of caret over direct at 10 million observations (0.992 vs 0.9635) exists but the excessive processing time (over 21 times longer) makes this advantage negligible in practical use. XGBoost implementation through direct execution provides optimal efficiency coupled with strong predictive power when working with datasets that exceed millions of observations. Most practical applications should select direct XGBoost as their preferred choice because the minimal performance benefits of caret are not worth the extended processing time it requires.