| 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 | 0.73 | 0.8 |
| | 1000 | 0.86 | 0.93 |
| | 10000 | 0.84 | 1.6 |
| | 100000 | 0.91 | 2.8 |
| | 1000000 | 0.94 | 13.6 |
| | 10000000 | 0.99 | 120.9 |
| XGBoost in R – direct use of xgboost() with simple cross-validation | 100 | 0.8 | 0.0202 |
| | 1000 | 0.91 | 0.0262 |
| | 10000 | 0.9775 | 0.1903 |
| | 100000 | 0.9850 | 1.7 |
| | 1000000 | 0.988435 | 14.767 |
| | 10000000 | 0.98666 | 156.987 |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.909 | 2.069 |
| | 1000 | 0.960 | 5.093 |
| | 10000 | 0.9827 | 25.710 |
| | 100000 | 0.9903 | 240.926 |
| | 1000000 | 0.9845 | 659.252 |
| | 10000000 | 0.9755 | 1269.62 |

Most use cases should employ direct XGBoost instead of caret implementation according to the data analysis. When working with XGBoost directly the system delivers record-breaking processing speed with results that match or surpass those of caret predictions when working with datasets of larger sizes. The direct XGBoost method obtained 0.988435 accuracy in 14.767 seconds with 1 million observations whereas the caret method reached 0.9845 accuracy in 659.252 seconds (44 times longer).

The time savings from direct XGBoost became most significant with 10M observations as it finished in 156.987 seconds while caret needed 1269.62 seconds which amounts to an 8x difference. The convenient cross-validation of caret comes at a steep computational cost which does not result in superior performance compared to direct XGBoost implementation that achieves better predictive accuracy (0.98666 vs 0.9755) in the largest datasets. The direct XGBoost implementation provides better performance than caret's simplified workflow because its speed advantages become crucial for practical applications and large dataset processing.