1.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 0.03s |
|  | 1000 | 0.925 | 0.07s |
|  | 10000 | 0.9775 | 1.43s |
|  | 100000 | 0.9862 | 13.66s |
|  | 1000000 | 0.9924 | 126.29s |
|  | 10000000 | 0.9850 | 653.23s |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.800 | 0.28s |
|  | 1000 | 0.9550 | 0.23s |
|  | 10000 | 0.9770 | 0.51s |
|  | 100000 | 0.9773 | 3.97s |
|  | 1000000 | 0.9786 | 39.91s |
|  | 10000000 | 0.9862 | 180.36s |

2.

The application of direct XGBoost using simple cross-validation yields better results than implementing it through caret. The direct XGBoost implementation offers faster runtimes than the other methods yet produces equivalent or better predictive results when operating on increased dataset sizes.

XGBoost performance metrics demonstrate better productivity with growing data volumes than indirect XGBoost implementation. The direct implementation outpaces the caret approach by 19 times when processing the 10M record dataset with a performance speed of 102.5 seconds compared to 1985.7 seconds. Operating at a rate that is two orders of magnitude faster than caret becomes critical for production use and working with big datasets that require hours instead of seconds to process. The trade-off in performance becomes unacceptable because the unified interface functionality of caret fails to deliver adequate speed when handling large datasets exceeding few thousand records.