1. **Compare the accuracy values of XGBoost models fit on the newly created data, for the following sizes of datasets. Along with accuracy, report the time taken for computing the results. Report your results in a table with the following schema.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method used** | **Dataset size** | **Testing-set predictive performance** | **Time taken for the model to be fit (seconds)** |
| **XGBoost in Python via scikit-learn and 5-fold CV** | **100** | 0.87 | 0.22 |
|  | **1000** | 0.949 | 0.81 |
|  | **10000** | 0.9746 | 1.51 |
|  | **100000** | 0.9871 | 4.48 |
|  | **1000000** | 0.9917 | 74.22 |
|  | **10000000** | 0.9931 | 267.07 |
| **XGBoost in R – direct use of xgboost() with simple cross-validation** | **100** | 1 | 0.142 |
|  | **1000** | 1 | 0.101 |
|  | **10000** | 0.988 | 0.118 |
|  | **100000** | 0.981 | 0.485 |
|  | **1000000** | 0.978 | 4.888 |
|  | **10000000** | 0.979 | 91.407 |
| **XGBoost in R – via caret, with 5-fold CV simple cross-validation** | **100** | 1 | 19.93 |
|  | **1000** | 0.991 | 34.48 |
|  | **10000** | 0.995 | 51.47 |
|  | **100000** | 0.994 | 254.36 |
|  | **1000000** | 0.993 | 2109.08 |
|  | **10000000** | 0.9927 | 11496.8 |

1. **Based on the results, which approach to leveraging XGBoost would you recommend? Explain the rationale for your recommendation.**

Based on the comparison of XGBoost implementations in Python and R across different dataset sizes, I would recommend using XGBoost in R through direct use of the xgboost() function with simple cross-validation. This method provides the best balance between speed and predictive performance. It consistently has the shortest model fitting time across all dataset sizes. For example, with a dataset of 1,000,000 rows, it only takes about 4.9 seconds to fit the model, while the Python implementation takes 74.2 seconds and the R caret method takes over 2,100 seconds.

In terms of predictive accuracy, the direct R method performs just as well as the other methods on small datasets and remains highly competitive on larger datasets. While the caret method in R shows slightly higher accuracy on some large datasets, the time it takes to train the model is disproportionately high and inefficient, especially for practical or time-sensitive tasks.

The Python approach using scikit-learn is a reasonable alternative, especially if you're working in a Python-based environment, but it is not as time-efficient as the native R implementation. Therefore, for most scenarios—especially those involving large datasets or limited computing time—the direct xgboost() approach in R is the most efficient and reliable option.