Question

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.

1. XGBoost in Python via scikit-learn and 5-fold CV

Dataset Size	Testing-set predictive performance	Time taken for this model to be fit
100	0.8800	0.27
1000	0.9480	0.30
10000	0.9789	0.73
100000	0.9867	2.04
1000000	0.9919	21.27
10000000	0.9931	193.78

2. XGBoost in R – direct use of xgboost() with simple cross-validation

Dataset Size	Testing-set predictive performance	Time taken for this model to be fit
100	0.8500	0.32
1000	0.9600	0.35
10000	0.9720	0.62
100000	0.9842	1.63
1000000	0.9885	19.02
10000000	0.9896	110.27

3. XGBoost in R – via caret, with 5-fold CV simple cross-validation

Dataset Size	Testing-set predictive performance	Time taken for this model to be fit
100	0.7368	1.96
1000	0.9200	1.86
10000	0.9750	2.97
100000	0.9844	8.21
1000000	0.9888	129.30
10000000	0.9897	689.55

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

Answer: Based on the results, I would choose XGBoost in Python with scikit-learn and 5-fold CV. Why? Because compared to the similar XGBoost in R with caret and 5-fold CV, the time taken for computation is much less in Python.

Meanwhile, XGBoost in R with basic cross-validation also takes the same amount of time as the 5-fold CV Python approach, but loses out on reliability of performance estimate, and hyperparameter tuning may not be as robust.