Week 11 - Comparison of R/Python for XGBoost

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	0.17
	1000	0.949	0.36
	10000	0.9746	0.76
	100000	0.9871	5.99
	1000000	0.9917	28.71
	10000000	0.9931	267.17
XGBoost in R – direct use of xgboost() with simple cross-validation	**100**	0.8313	0.09
	1000	0.932	0.08
	10000	0.9252	0.12
	100000	0.9285	0.57
	1000000	0.9308	5.81
	10000000	0.9304	66.46
XGBoost in R – via caret, with 5-fold CV simple cross-validation	**100**	0.8894	23.76
	1000	0.952	27.84
	10000	0.9834	42.61
	100000	0.9899	205.69
	1000000	0.9924	2567.5
	10000000	0.9927	10496.8

2) **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.**

Based on the updated results, I would recommend using XGBoost in R – direct use of xgboost() with simple cross-validation.

The key reason is that this method offers an excellent trade-off between predictive performance and computational efficiency. As seen in the table, although the testing-set predictive performance of this method is slightly lower than that of the caret approach (especially for very large datasets), the difference is minor (e.g., 0.9304 vs. 0.9927 for 10 million rows). However, the time taken for direct xgboost() is dramatically lower — 66.46 seconds compared to 10,496.8 seconds for caret, which is a huge advantage in real-world applications where computational cost and time matter.

Comparing it with the Python scikit-learn approach, the R direct method also generally runs faster as the dataset grows. For example, at 10 million rows, Python took 267.17 seconds, while direct xgboost() in R took only 66.46 seconds about four times faster with comparable predictive performance (Python: 0.9931, R direct: 0.9304).

Meanwhile, although caret produced the highest accuracy across most dataset sizes, its training time grows exponentially, making it impractical for large-scale data modeling. Waiting over 3 hours to fit a model (as with 10 million rows) is not feasible when faster alternatives are available with almost equivalent accuracy.

Thus, considering the balance between accuracy and computational efficiency, XGBoost in R using direct simple cross-validation is the most practical and efficient approach. It would be the recommended method for both medium- and large-scale datasets where time and resource management are crucial.