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.**

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	0.9200	0.97
	1,000	0.9500	0.86
	10,000	0.9777	0.76
	100,000	0.9872	3.87
	1,000,000	0.9919	46.43
	10,000,000	0.9932	412.06
XGBoost in R – direct use of xgboost()	100	1.0000	0.30
	1,000	1.0000	0.42
	10,000	1.0000	0.60
	100,000	0.9997	2.61
	1,000,000	0.9959	33.47
	10,000,000	0.9942	398.94
XGBoost in R – via caret, with 5-fold CV	100	0.8509	0.43
	1,000	0.9240	0.42
	10,000	0.9338	0.56
	100,000	0.9358	3.03
	1,000,000	0.9362	28.31
	10,000,000	Skipped	Too slow to run

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

Based on the above results, the best approach for leveraging XGBoost depends on the balance between accuracy, scalability, and execution time. The direct use of xgboost() in R consistently produced the highest accuracy across all dataset sizes, with relatively low training times, even for the largest dataset (10 million rows). It achieved near-perfect predictive performance with minimal configuration, making it highly effective and efficient for large-scale modeling tasks.

The Python implementation using scikit-learn also performed very well, offering high accuracy and competitive training times across all dataset sizes. It scaled efficiently to 10 million rows and remained user-friendly, which makes it a strong choice for those working in Python environments.

In contrast, the caret implementation in R, while accurate on small datasets, showed limitations as dataset size increased. The accuracy plateaued around 93–94% and the method was unable to process the 10 million-row dataset due to computational constraints. While caret is useful for hyperparameter tuning and model evaluation in smaller projects, it is not well-suited for very large datasets.

Therefore, I would recommend using xgboost() directly in R for its superior performance and scalability, or scikit-learn's XGBoost in Python for a balance of performance and ease of use. The caret-based approach should be limited to small datasets or educational purposes where interpretability and tuning convenience are more important than execution speed.