Week 11 Assignment: XGBoost Model Comparison

Comparison of XGBoost Methods

Method	Dataset Size	Accuracy	Time (s)
Python + sklearn (5-fold CV)	100	0.93	0.29
Python + sklearn (5-fold CV)	1000	0.952	0.29
Python + sklearn (5-fold CV)	10000	0.973	0.85
Python + sklearn (5-fold CV)	100000	0.9865	6.35
Python + sklearn (5-fold CV)	1000000	0.9918	46.49
Python + sklearn (5-fold CV)	10000000	0.9932	433.76
Direct XGBoost	100	1.0	0.24
Direct XGBoost	1000	1.0	0.4
Direct XGBoost	10000	1.0	0.8
Direct XGBoost	100000	0.9996	2.58
Direct XGBoost	1000000	0.996	22.1
Direct XGBoost	10000000	0.9942	190.74
Caret XGBoost (5-fold CV)	100	0.9109	1.94
Caret XGBoost (5-fold CV)	1000	0.953	2.32
Caret XGBoost (5-fold CV)	10000	0.9727	4.25
Caret XGBoost (5-fold CV)	100000	0.9865	15.57
Caret XGBoost (5-fold CV)	1000000	0.9922	120.6
Caret XGBoost (5-fold CV)	10000000	0.9935	1206.0

Detailed Explanation and Recommendation

Python + scikit-learn:

This approach is known for very fast execution due to multi-threading capabilities. It consistently delivers high accuracy, which improves as the dataset size increases. Additionally, it integrates seamlessly with pipelines

and grid search tools, making it ideal for production environments. However, one drawback is that hyperparameter tuning must be done manually, unlike R's caret package.

R Direct xgboost():

The direct use of the xgboost() function in R produced the highest accuracy across all dataset sizes and performed efficiently on small to medium datasets. It is lightweight and fast. However, it does not have built-in support for cross-validation, and the accuracy is evaluated on the training data itself, which may lead to overfitting or an overly optimistic performance estimate. This approach is best suited for quick experimentation or when using fixed hyperparameters.

R Caret + XGBoost (5-fold CV):

This method is user-friendly, allowing easy integration of cross-validation and hyperparameter tuning. The accuracy results from this method are more realistic and unbiased. However, as the dataset size grows, caret becomes significantly slower and more memory-intensive. It is most effective for small to medium datasets where tuning is crucial.

Final Recommendation:

Based on the overall results, I recommend using Python with scikit-learn's XGBClassifier for handling large datasets. It strikes an excellent balance between accuracy and speed, while also supporting cross-validation. It is also ideal for production deployments due to its compatibility with cloud-based tools and its extensive library support. For experimentation in R, caret is ideal for tuning, and direct xgboost is suitable for rapid, small-scale tests. Ultimately, for scalable, real-world applications, Python + sklearn remains the most practical and robust solution.