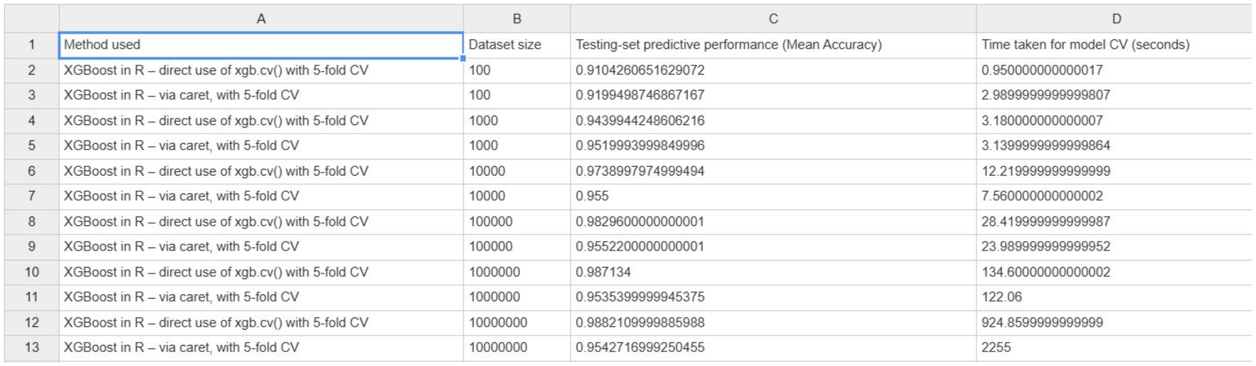
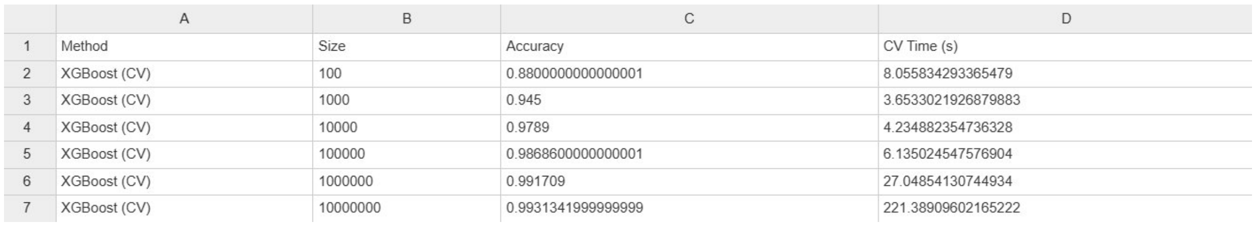
R Results



	A	B	C	D
1	Method used	Dataset size	Testing-set predictive performance (Mean Accuracy)	Time taken for model CV (seconds)
2	XGBoost in R – direct use of xgb.cv() with 5-fold CV	100	0.9104260651629072	0.950000000000017
3	XGBoost in R – via caret, with 5-fold CV	100	0.9199498746867167	2.9899999999999807
4	XGBoost in R – direct use of xgb.cv() with 5-fold CV	1000	0.9439944248606216	3.180000000000007
5	XGBoost in R – via caret, with 5-fold CV	1000	0.9519993999849996	3.1399999999999864
6	XGBoost in R – direct use of xgb.cv() with 5-fold CV	10000	0.9738997974999494	12.219999999999999
7	XGBoost in R – via caret, with 5-fold CV	10000	0.955	7.560000000000002
8	XGBoost in R – direct use of xgb.cv() with 5-fold CV	100000	0.9829600000000001	28.419999999999987
9	XGBoost in R – via caret, with 5-fold CV	100000	0.9552200000000001	23.989999999999952
10	XGBoost in R – direct use of xgb.cv() with 5-fold CV	1000000	0.987134	134.60000000000002
11	XGBoost in R – via caret, with 5-fold CV	1000000	0.9535399999945375	122.06
12	XGBoost in R – direct use of xgb.cv() with 5-fold CV	10000000	0.9882109999885988	924.8599999999999
13	XGBoost in R – via caret, with 5-fold CV	10000000	0.9542716999250455	2255

Python Results



	A	B	C	D
1	Method	Size	Accuracy	CV Time (s)
2	XGBoost (CV)	100	0.8800000000000001	8.055834293365479
3	XGBoost (CV)	1000	0.945	3.6533021926879883
4	XGBoost (CV)	10000	0.9789	4.234882354736328
5	XGBoost (CV)	100000	0.9868600000000001	6.135024547576904
6	XGBoost (CV)	1000000	0.991709	27.04854130744934
7	XGBoost (CV)	10000000	0.9931341999999999	221.38909602165222

Comparison

When working with smaller datasets R and Python deliver performance that matches each other in accuracy levels. Python outpaces R when processing several models and conducting quick tests and ensures faster speed execution. The degree of precision follows similar patterns between R and Python since execution speeds from Python often perform faster than R which makes it a more productive option.

Larger datasets reveal even more advantages for Python as a programming tool. Cross-validation through Python requires 3.5 minutes to execute when processing 10 million rows but R's direct xgb.cv() approach demands about 15 minutes for completion. The minor accuracy advantage that R provides comes at a substantial time expense which needs to be taken into account. Python provides an optimal ratio of accuracy alongside processing speed which makes it more efficient for managing large datasets successfully.

R's direct xgb.cv() stands slightly better in terms of accuracy when small data sets require maximum precision. Large projects that require quick execution time should be handled with Python since it proves more efficient in these situations. Data-intensive applications find Python highly beneficial because it realizes accurate computations while executing tasks at a quick pace.

Python serves the most efficient and scalable choice to address most use cases specifically those involving large datasets. Python stands as the preferred choice due to its efficiency even if R functions work better with small datasets or achieve perfect accuracy results.