	Module/framework/package	Name and a brief description of the algorithm	An example of a situation where using the provided GLM implementation provides superior performance compared to that of base R or its equivalent in Python (identify the equivalent in Python)
a.Base R (in the stats library)	stats packages	glm() is used to fit generalized linear models, specifying a symbolic description of the linear predictor and a description of the error distribution.	When working with a dataset exceeding 10 GB, such as customer transaction records, base R's glm() struggles due to memory limitations and single-threaded computation. In contrast, the equivalent in Python—Dask-ML—enables scalable GLM training by distributing the computation across multiple workers, processing data in parallel and in chunks, thus offering significantly better performance.
b. Big data version of R	biglm packages,	By processing data in pieces, GLMs using biglm() enable modeling on datasets that don't fit in memory.	When working with large survey or census data that cannot fit in memory, using biglm allows for chunk-wise computation, enabling GLM fitting without crashing or slowing down—something base R’s glm() can't handle efficiently. The equivalent in Python would be Dask-ML or PySpark, which also allow for distributed GLM training on large datasets.
c. Dask ML	dask-ml	Using Dask arrays/dataframes, parallel/distributed computing implements scalable machine learning including LinearRegression and LogisticRegression.	In scenarios like training a logistic regression model on an over 100GB dataset of IoT sensor data, Dask ML can efficiently process and fit the model using distributed memory, while scikit-learn would crash or require manual data chunking. The equivalent in R is SparkR, which also supports distributed GLMs.

d. Spark R	SparkR	Using distributed computing, spark.glm() fits GLMs on Spark DataFrames, supporting many families such as binomial, gaussian.	For training models on a Spark DataFrame with hundreds of millions of ad impressions, spark.glm() can leverage the distributed environment to train a GLM efficiently. Base R or Python equivalents would be limited by memory or single-machine constraints. The equivalent in Python is pyspark.ml.classification.LogisticRegression.
e. Spark optimization	Spark MLlib	Fits GLMs using scalable optimizers such as stochastic gradient descent (SGD) and L-BFGS, designed for large-scale machine learning pipelines.	In a real-time recommendation system where GLMs must be retrained periodically on streaming data from millions of users, Spark MLlib’s optimizers like SGD allow fast convergence across a distributed cluster, outperforming single-machine algorithms in Python or R. The equivalent in Python would be scikit-learn, which lacks scalable optimizers like Spark's SGD for distributed data.
f. Scikit-learn	Scikit-learn	Implements LogisticRegression(), LinearRegression(), and other GLMs via sklearn.linear_model, with support for various solvers like liblinear, lbfgs, sag, and saga.	When dealing with a clean, moderately sized (~1GB) customer churn dataset, scikit-learn’s LogisticRegression can quickly fit a model with multiple solver options, offering better usability and solver flexibility than base R’s glm(). The equivalent in R is glm() in the stats package.