Module/framework/package	**Name and brief description of 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)**
Base R	Iteratively Reweighted Least Squares (IRLS) The base GLM implementation in R depends on IRLS to perform model fitting through an iterative procedure of updating weights by solving least squares problems until convergence occurs. The method performs calculations for working responses and weights using current parameter estimates before solving weighted least squares problems.	Base R IRLS works best for datasets that fit into memory while needing exact statistical inference for standard errors and p-values and other diagnostics. R's implementation surpasses Python's statsmodels implementation in statistical diagnostics computation because R offers more detailed deviance residuals and influence measure diagnostics.
Big Data version of R	Distributed IRLS and SGD variants The distributed GLM algorithms for big data in R exist through implementations provided by sparklyr and H2O. Rmpi provides parallel execution for IRLS through its package capabilities as well as biglm implements chunked algorithms to work with large out-of-memory data. H2O provides users with IRLS and gradient descent algorithms which distribute computations across multiple clusters.	The systems deliver exceptional performance when dealing with massive datasets that exceed memory capacity. Sparklyr enables processing of GLMs on terabyte-scale distributed clusters through its connection to MLlib in Spark which performs better than base R which crashes due to memory limitations. The statistical accuracy of Dask-ML GLM processing in Python

		matches H2O but remains superior to Python's Dask-ML approach.
Dask ML	Gradient Descent and L-BFGS Dask ML offers users four different optimization methods for GLMs: ADMM (Alternating Direction Method of Multipliers), L-BFGS, proximal gradient descent and Newton's method. The parallel computing framework of Dask allows these operations to process distributed arrays.	Dask ML provides superior capabilities than scikit-learn for processing large datasets which exceed the memory capacity of standalone machine RAM. Dask ML enables clusters to handle 100+ GB logistic regression model fitting operations through a scikit-learn API interface. The Python ecosystem integration makes this solution perform better than both base R and big data R implementations.
Spark R	L-BFGS and Stochastic Gradient Descent The default optimizer for GLM in Spark R is L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) yet it provides additional options for Normal Equation and Stochastic Gradient Descent for particular scenarios. The framework of distributed computing in Spark supports these operations.	Spark R effectively operates on data collections exceeding terabytes in size which are stored in distributed systems including HDFS and S3. The performance of Spark R exceeds base R by a large margin when running logistic regression on datasets containing billions of observations. The integration of big data performance with R syntax is a key advantage of this system over the Python-based scikit-learn framework.
Spark optimization	Multiple algorithms: L-BFGS, SGD, LBFGS-OWLQN The optimization module of Spark	The optimization features of MLlib within Spark accelerate the execution of machine

	implements L-BFGS for smooth objectives and LBFGS-OWLQN (Orthant-Wise Limited-memory Quasi-Newton) for L1 regularization and SGD/minibatch SGD for large-scale learning. The algorithms operate efficiently within distributed systems according to their design.	learning programs at enterprise-level scales. The performance of Spark MLlib exceeds both R and Python standalone implementations when working with datasets that contain billions of features and observations such as click prediction models. The OWLQN implementation delivers superior efficiency for L1-regularized problems than both coordinate descent in scikit-learn and glmnet in R.
Scikit-Learn	Multiple solvers: LBFGS, Liblinear, Newton-CG, SAG/SAGA Scikit-learn provides multiple GLM solvers including LBFGS for general use and Liblinear for small datasets and Newton-CG for multinomial problems and SAG and SAGA for large datasets. The solvers function specifically for different problem features such as data scale combined with regularization strengths and support for multiple categories.	Scikit-learn yields faster performance compared to R when processing data sets of average dimensions through its optimized C/Cython implementation. When running L1 regularization with SAGA solver on large sparse data sets (like text TF-IDF features) it achieves superior performance when compared to R's glmnet implementation because SAGA supports various regularization types alongside high speed processing.