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	The Iteratively Reweighted Least Squares (IWLS) method applies an iterative solution of weighted least squares problems until it reaches convergence. Fisher scoring serves as an equivalent to Newton-Raphson yet it substitutes the Hessian with its expected value calculated under the model framework. IWLS presents itself as Iteratively Reweighted Least Squares through a process that develops weights from predicted mean values and performs re-estimation until convergence occurs.	The IWLS algorithm in Base R operates best with traditional statistical modeling needs and datasets that can be stored in memory. The statistical analysis capabilities of this implementation surpass basic Python implementations because it provides superior deviance residuals and influence measures for hypothesis testing.
Big Data version of R	Multiple R packages support distributed workload along with big data analysis and out-of-memory processing including biglm, snow, and foreach/doParallel. The HPC view in R provides multiple methods to distribute Generalized Linear Models computations across multiple cores or clusters.	The biglm package enables GLM fitting to external data storage through an incremental processing system. The statistical interface of R remains familiar through this approach which delivers better performance than base R for datasets exceeding RAM capacity. Biglm provides the capability to process massive datasets and genomic research with millions of observations which standard R cannot handle.

Dask ML	ADMM (Alternating Direction Method of Multipliers) divides complex problems into separate sub-problems which run in parallel before combining their solutions. The Proximal Gradient Method serves as a non-smooth regularizer in Dask ML while the platform supports distributed L-BFGS implementations.	The ADMM algorithm in Dask ML delivers superior results when processing data that exceeds RAM capacity on one machine yet remains smaller than what needs a complete Spark cluster. Dask ML provides a hybrid solution between single-machine and full cluster computing by processing out-of-core data through its familiar scikit-learn API. The ability to handle data science teams during their transition from laptop-based to server-based analytics makes this feature highly beneficial.
Spark R	The distributed L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) method uses quasi-Newton optimization to compute parameter updates across a Spark cluster in distributed fashion. The method calculates parameter updates through Hessian matrix approximations derived from previous gradient evaluations while avoiding the construction of the complete Hessian.	The superior performance trait of SparkR functions best for datasets of terabyte size which require parallel processing across a distributed cluster. The system delivers better performance than base R when processing massive datasets located on distributed file systems such as HDFS. Insurance companies handling millions of claims data and telecommunications firms processing customer behavior data can execute previously impossible analyses because of SparkR's distributed computing capabilities.
Spark optimization	SparkR provides three distributed algorithms including Gradient Descent with distributed data chunks, SGD with miniBatchFraction sampling and Distributed L-BFGS which distributes cluster workloads. Spark	Spark performs L-BFGS implementation faster than scikit-learn SGD methods in handling distributed large-scale datasets. Spark MLlib enables distributed optimization of large clickstream and IoT sensor

	MLlib provides three optimization methods through its implementation including gradient descent and stochastic gradient descent with mini-batches and L-BFGS.	data through optimizations which outperform first-order methods in both speed and scalability. Users can use the miniBatchFraction parameter to optimize between the speed of each iteration and the rate of convergence.
Scikit-Learn	The library offers a multi-solver system which includes the 'lbfgs' Limited-memory BFGS and 'liblinear' and 'newton-cg/cholesky' second-order solvers and the 'sag/saga' Stochastic average gradient procedures. The Scikit-learn platform offers four different solvers: LBFGS as the default option and Liblinear for L1 regularization and Newton-CG/Cholesky and SAG/SAGA methods.	SAGA delivers superior performance for working with large datasets that need L1 regularization because it surpasses R's glmnet package in processing large databases with highly dimensioned features. Text classification tasks that work with sparse high-dimensional matrices demonstrate better convergence rates through SAGA over the coordinate descent methods while enabling sophisticated regularization options. The excellent functionality of Scikit-learn enables users to integrate pipelines with preprocessing blocks and cross-validation tests as well as hyperparameter optimization methods.