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). IRLS transforms the GLM into successive weighted least squares problems through its iterative process. The process starts by calculating working response and weights from current parameter estimates before solving weighted least squares problems until it reaches convergence.	The GLM implementation from Base R produces optimal results for statistical diagnostic examinations and inferential analysis. The model summaries along with statistical tests and diagnostics presented by Base R exceed the diagnostic capabilities of Python's statsmodels. This approach delivers superior benefits to datasets of small to medium size that require traditional statistical analysis methods.
Big Data version of R	The implementations of IRLS and Gradient Descent run across discrete computing systems and parallel computing architectures. The packages apply distributed computing frameworks to execute GLM computations in parallel. The packages sparklyr applies MLlib algorithms from Spark whereas pbdR executes computations through Message Passing Interface (MPI) for high-performance computing. The platforms deploy data elements and processing requests between different computing units that comprise machines and cores.	The performance of sparklyr surpasses base R when working with datasets that exceed memory capacity of a single machine. The analysis of terabyte-scale customer behavior data for a retail chain becomes possible through distributed R implementations because base R lacks sufficient memory capacity. Users who work with R will likely prefer sparklyr API over PySpark because it matches their familiar modeling syntax from R.
Dask ML	ADMM (Alternating Direction Method of	Dask ML provides better performance than scikit-learn

	Multipliers) and other optimization algorithms. Dask ML provides users with several optimization algorithms for GLMs which include ADMM, L-BFGS, proximal gradient descent and Newton's method. The algorithms function within distributed arrays and dataframes of Dask to enable cluster-based parallelization of computations.	(Python's equivalent to base R for GLMs) for datasets that exceed memory capacity yet fall below the threshold for implementing a full Spark cluster. The capacity of scikit-learn is exceeded by 50-100GB logistic regression model fits while Dask ML can handle these models through either a small cluster or a single multi-core machine that utilizes disk-based data spilling. The solution exists between working with a single machine and deploying a complete distributed computing system.
Spark R	L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno). The GLM implementation in Spark R depends mainly on L-BFGS which functions as a quasi-Newton method that employs limited memory to approximate the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. The system includes Stochastic Gradient Descent as one of its available options. Optimizers function as part of the distributed processing capability of Spark by executing within its framework.	Spark R delivers the best performance in processing extensive datasets which reside in storage systems HDFS or S3. The analysis of telecommunications network data containing billions of records would benefit greatly from Spark R because base R would encounter memory errors and fail to complete the task. Spark R delivers advantages to R users who need to connect their R-based workflows with Spark's distributed computing capabilities while working with R's statistical modeling methods.
Spark optimization	The system uses multiple specialized algorithms which include L-BFGS and OWLQN. MLlib within Spark provides multiple optimization algorithms that operate optimally in distributed computing	The implementation of Spark MLlib provides optimum execution when building regularized GLMs on massive enterprise datasets. The OWLQN algorithm from Spark MLlib achieves faster convergence during L1-

	environments. The optimization algorithms L-BFGS handle smooth objectives and OWLQN handles problems with L1-regularization. The system provides minibatch SGD variants to perform large-scale learning tasks across distributed clusters.	regularized sparse logistic regression modeling of text data consisting of millions of documents and features compared to coordinate descent techniques from R glmnet and scikit-learn LogisticRegression. A distributed computational approach allows the processing of datasets which would exceed the capacity of a single machine system.
Scikit-Learn	The system contains three specialized optimization algorithms named L-BFGS, SAGA and Newton-CG. Scikit-learn provides multiple solvers that have been optimized for distinct problem conditions including L-BFGS for general usage and SAGA for L1 regularization with large data sets and Newton-CG for multinominal problems and specialized solvers that meet specific regularization requirements. The solution uses optimized numerical libraries together with efficient memory management.	The GLM implementations in Scikit-learn deliver optimal speed performance when working with datasets of medium size that fit in memory due to their highly optimized C/Cython code. When using both tens of thousands of samples and thousands of features on elastic-net regularized logistic regression models the SAGA solver within scikit-learn demonstrates faster convergence compared to R's glmnet. The specific solvers in scikit-learn outperform equivalent base R implementations when working with problems that need both L1 and L2 regularization.