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 glm.fit function applies Iterative Reweighted Least Squares (IRLS) that solves weighted least squares problems through an iterative process until reaching convergence. It functions as a Fisher scoring method through which parameter estimates get updated by performing Hessian matrix operations.	The reference execution of Base R GLM functions exists as the standard implementation. Python users can find equivalent functionality through statsmodels GLM implementation.
Big Data version of R	Packages Rmpi and snow and parallel and pbdMPI provide options for parallel execution of GLMs. The bigglm package through biglm implements chunked IRLS while batchtools distributes GLM computation across clusters with different schedulers.	The system works best for processing very large data sets that exceed memory capacity. The bigglm package along with distributed frameworks allows R to process datasets containing billions of observations that would result in memory errors when using base R. Dask with statsmodels provides Python users with distributed GLM capabilities but R maintains stronger options for distributed GLM processing.
Dask ML	SparkR implements distributed optimization algorithms L-BFGS, OWL-QN and Iterative Residual Rescaling through which users can customize parameters maxIter, tol and regParam. The framework	The system outmatches base R by processing enormous datasets distributed across multiple clusters. The SparkR framework enables the processing of enormous datasets across hundreds of nodes thus making it possible

	enables fitting of GLM models from different families which include Gaussian, Binomial, Poisson, Gamma and Tweedie distributions.	to fit GLMs that base R cannot handle. The Python equivalent to SparkR is PySpark however SparkR provides better compatibility for users who work in R.
Spark R	The system selects from different optimization algorithms to match the model and regularization requirements. The LogisticRegression class in SparkR supports 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' and 'newton-cholesky' as its available solvers. The solvers in this implementation demonstrate optimal performance for L1, L2 and ElasticNet penalties across varying dataset conditions.	Scikit-learn provides superior performance with large datasets through its 'sag' and 'saga' solvers because they scale linearly with the sample size. The 'saga' solver achieves much better performance than base R's GLM when applied to extremely large sparse datasets with L1 regularization. The equivalent package in R is glmnet but scikit-learn provides better integration within the Python data science environment.
Spark optimization	The framework supports different distributed optimization algorithms which include Gradient Descent and L-BFGS and Limited-memory BFGS together with communication-efficient aggregation trees. To achieve better performance the system utilizes standardization of features and stochastic variations and minibatch processing techniques.	The system demonstrates outstanding performance when handling distributed training operations that require data parallelism. The system manages datasets of petabyte scale across extensive clusters that base R chunked processing would not succeed in handling. Particularly efficient for GLMs with large feature spaces (millions of features) due to optimized gradient aggregation. The equivalent tool in R is SparkR which we have already discussed but Spark MLlib enables users to access more advanced optimization capabilities.
Scikit-Learn	The system provides four distributed algorithms including ADMM (Alternating Direction	Python developers who work with large datasets exceeding memory capacity will find Superior as their ideal

	Method of Multipliers) and L-BFGS and proximal gradient descent and Newton's method. The problem characteristics determine which algorithm users choose from the available options. The distributed computing environment benefits the most from ADMM implementations.	solution when they do not need a complete Spark cluster. The task scheduling system and lazy evaluation features of Dask allow users to process data efficiently on either one multi-core machine or a small cluster. The proximal gradient method in Dask ML surpasses scikit-learn performance when dealing with L1-regularized problems that exceed RAM capacity. The R equivalent to distributed computing with future package exists but Dask ML provides superior integration with the Python scientific stack.