GLM Investigation

Generalised linear model (GLM) is an essential statistical tool used in many fields to simulate the relationships between a response variable and one or more predictor variables. GLMs have been implemented using a variety of statistical and machine learning frameworks, each of which uses a unique optimisation algorithm suited to its design principles and performance objectives. The optimisation techniques used by some well-known frameworks and packages for estimating GLM parameters are summarised in this paper. It also identifies instances in which these implementations might perform better than the glm function in base R or its Python counterparts.

Module/Framework/Package	**Name and Brief Description of the Algorithm**	**Example of Superior Performance Compared to Base R or Equivalent in Python**
a. Base R (stats package)	**Iteratively Reweighted Least Squares (IRLS)**: The glm function in base R uses IRLS to estimate parameters for GLMs, iteratively updating weights to find maximum likelihood estimates.	It is suitable for small to moderately sized datasets. For larger datasets, specialized packages like biglm may offer better performance.
b. Big Data Versions of R (biglm package)	**Modified IRLS with Chunk Processing**: The bigglm function in the biglm package processes data in chunks, allowing GLM fitting on datasets larger than available memory.	Efficiently handles datasets that exceed available memory by processing data in chunks, outperforming base R's glm function in such scenarios.
c. Dask-ML	**Scalable Algorithms with Various Optimizers**: Dask-ML implements scalable algorithms for GLMs that can operate on large datasets distributed across clusters, supporting optimization	Ideal for large-scale data that cannot fit into a single machine's memory, leveraging distributed computing for efficient model training. In Python, Dask-ML provides better scalability compared to scikit-learn's

	methods like Gradient Descent and L-BFGS.	LogisticRegression for big data.
d. SparkR	**Distributed GLM Fitting with Regularization**: The spark.glm function fits GLMs on large datasets using Spark's distributed computing framework, supporting elastic-net regularization for improved model performance.	Suitable for massive datasets distributed across a cluster, processing data in parallel across multiple nodes, offering significant performance improvements over base R's glm when handling big data.
e. Spark MLlib	**Stochastic Gradient Descent (SGD) and L-BFGS**: Spark MLlib offers scalable machine learning algorithms, including GLMs, optimized for distributed computing using optimization techniques like SGD and L-BFGS for large-scale data processing.	Efficiently scales computations across a cluster for extensive datasets requiring distributed processing, outperforming single-machine approaches like base R's glm. In Python, Spark MLlib can handle larger datasets than scikit-learn's LogisticRegression.
f. Scikit-learn	**Coordinate Descent and Stochastic Gradient Descent**: Scikit-learn implements GLMs using optimization algorithms such as coordinate descent for Lasso regression and stochastic gradient descent for large-scale linear models.	For moderately large datasets that fit into memory, scikit-learn's optimized algorithms can provide faster computation times compared to base R's glm. However, for extremely large datasets, distributed frameworks like Dask-ML or Spark may be more appropriate.

In summary, while base R's glm function is suitable for smaller datasets, specialized packages and frameworks like biglm, Dask-ML, SparkR, Spark MLlib, and scikit-learn offer optimized algorithms and distributed computing capabilities that provide superior performance for larger datasets or big data scenarios.

References:

1. **Base R (stats package)**:
   - **Optimization Method**: Iteratively Reweighted Least Squares (IRLS).
   - **Reference**: glm function - Fitting Generalized Linear Models
2. **Big Data Versions of R (biglm package)**:
   - **Optimization Method**: Modified IRLS algorithm that processes data in chunks to reduce memory usage.
   - **Reference**: CRAN Task View: High-Performance and Parallel Computing with R
3. **Dask-ML**:
   - **Optimization Methods**: Supports various optimization algorithms and regularizers, following the scikit-learn estimator API.
   - **Reference**: Generalized Linear Models — Dask-ML Documentation
4. **SparkR**:
   - **Optimization Methods**: Utilizes optimization techniques suitable for large-scale data processing.
   - **Reference**: Generalized Linear Models — SparkR Documentation
5. **Spark MLlib**:
   - **Optimization Methods**: Employs convex optimization methods such as Stochastic Gradient Descent (SGD) and Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS).
   - **Reference**: Classification and regression - Spark 3.5.4 Documentation
6. **Scikit-learn**:
   - **Optimization Methods**: Implements algorithms like coordinate descent for Lasso regression and stochastic gradient descent for large-scale linear models.
   - **Reference**: Linear Models — scikit-learn Documentation