Generalized Linear Models from first principles. Advanced Machine Learning (MDS)

Joan Acero Mateja Zatezalo Pawarit Jamjod

November 2025

Abstract

Generalized Linear Models (GLMs) extend classical linear regression to a broad class of response types by combining exponential family distributions with link functions. In this project, we derive GLMs from first principles, emphasizing their probabilistic foundations, the role of canonical links, and the likelihood structure. Building on this theory, we analyze the iterative algorithms used for maximum likelihood estimation: Newton–Raphson, Fisher Scoring, and Iteratively Reweighted Least Squares. We then implement logistic regression from scratch using the IRLS algorithm, incorporating essential inference tools, including standard errors, Wald tests, confidence intervals, and odds-ratio interpretation. The implementation also includes automatic stepwise variable selection and numerical safeguards for reliable convergence. Finally, we validate the estimator on simulated and biomedical datasets, demonstrating that its performance aligns with that of standard statistical software while providing complete transparency into the underlying computations.

1 Introduction

GLMs provide a unified framework for modeling diverse types of response variables, from binary outcomes to counts and continuous data with non-constant variance. Although GLMs are widely used across statistics and machine learning, their practical implementation is often treated as a black box, with the underlying likelihood structure and estimation algorithms usually obscured by pre-implemented packages. This project addresses that gap by reconstructing GLMs from the ground up and examining how their probabilistic structure leads to the iterative algorithms used in practice.

The main challenge lies in bridging the theoretical formulation of GLMs (rooted in exponential family distributions and link functions) with the numerical procedures required for maximum likelihood estimation. Unlike classical linear regression, GLM likelihood equations rarely admit closed-form solutions; instead, they must be solved using iterative second-order methods such as Newton–Raphson, Fisher Scoring, or Iteratively Reweighted Least Squares. Understanding these algorithms is important for interpreting model behaviour, diagnosing convergence issues, and extending the methodology to non-standard settings.

In this work, drawing on the material from Part I of the course and additional sources, we develop a complete theoretical foundation for GLMs and then implement logistic regression from scratch. Our implementation includes inference tools (standard errors, confidence intervals, Wald tests, and odds ratios) as well as an automated stepwise selection routine inspired by the R step() method. Beyond reproducing standard results, our goal is to make each computational step transparent and theoretically grounded.

The remainder of the report is organized as follows. Section 2 introduces the exponential family and the general GLM framework. Section 3 presents common GLMs and their link functions. Section 4 describes the custom logistic regression estimator and its numerical considerations. Section 5 validates the method on simulated and biomedical datasets. Appendices contain pseudocode and supplementary results.

2 Background

The contents of this section are derived from [1], [2], and Advanced Machine Learning lecture notes.

We first introduce the exponential family of distributions. Subsequently, we define the Classical Linear Model that GLMs generalize. Finally, we introduce GLMs.

2.1 The exponential family

A family of distributions $\{P_\theta : \theta \in \Theta\}$ is said to be a k-parameter exponential family on $\mathbb{R}^q$ if its density function $p_\theta(x)$ can be written as:

$$p_\theta(x) = \exp\left[\sum_{i=1}^{k} \eta_i(\theta) T_i(x) - B(\theta)\right] h(x)$$

Where $\theta \in \Theta \subset \mathbb{R}^k$ is the parameter indexing the distribution, $\eta_1, \eta_2, \ldots, \eta_k$ and B are real-valued functions of the parameter θ, and $T_1, T_2, \ldots, T_k$ and h are real-valued functions of the response variable $x \in \mathbb{R}^q$.

The intuition is that the exponential family structure restricts how the parameter (θ) and the variable (x) can interact. The interaction must be of the form of an inner product between a function of θ and a function of x (the T_i terms) within the exponent.

Let's take a closer look at each component:

- Interaction Term: $\sum_{i=1}^{k} \eta_i(\theta) T_i(x)$. This is the natural generalization of the simple product $x \times \theta$ to higher dimensions, where η and T map the variables into a space where their inner product can be calculated.
- Base Measure Function: $h(x)$. This function depends only on the variable x (or y), and allows the framework to incorporate both discrete and continuous distributions.
- Cumulant Generating Function: $B(\theta)$ (or $b(\theta)$). This term depends only on the parameters θ. It serves as a normalizing constant, ensuring the density integrates or sums to one. $B(\theta)$ captures everything that is characteristic to a particular distribution. Crucially, the derivatives of $B(\theta)$ provide the mean and variance:

  $$\mathbb{E}(Y) = \mu = b'(\theta) \quad \text{and} \quad \text{var}(Y) = b''(\theta)\phi$$

  The fact that the variance must be positive, $\text{var}(Y) > 0$, implies $b''(\theta) > 0$, ensuring concavity of the log-likelihood in θ. Under the canonical link, this concavity transfers to β, guaranteeing a unique MLE.

2.2 One-Parameter Canonical Exponential Family

In GLM analysis, the one-parameter canonical form (assuming $k = 1$ and $q = 1$) is often used, where $T(y) = y$. The density function can be written as:

$$f_\theta(y) = \exp\left[\frac{y\theta - b(\theta)}{\phi} + c(y, \phi)\right]$$

Where θ is the canonical parameter, and ϕ is the dispersion parameter. If known, this is a one-parameter exponential family with θ being the canonical parameter.

Table 1 shows three examples of this family.

2.3 Classical linear models

Let, Y be the $n \times 1$ vector of observations $(Y_1, \ldots, Y_n)^\top$, X be the $n \times p$ model matrix where the i-th row $X_i^\top$ is the vector of p covariate values for the i-th observation, β be the $p \times 1$ vector of parameters $(\beta_1, \ldots, \beta_p)^\top$, and μ be the $n \times 1$ vector of expected means $(\mu_1, \ldots, \mu_n)^\top$.

A classical linear model consists of two key components.

Table 1: One-Parameter Canonical Exponential Family Examples

	Normal	Poisson	Bernoulli
Notation	$\mathcal{N}(\mu, \sigma^2)$	$\mathcal{P}(\mu)$	$\mathcal{B}(p)$
Range of y	$(-\infty, \infty)$	$[0, \infty)$	$\{0, 1\}$
ϕ	σ^2	1	1
$b(\theta)$	$\frac{\theta^2}{2}$	e^θ	$\log(1 + e^\theta)$
$c(y, \phi)$	$-\frac{1}{2}\left(\frac{y^2}{\phi} + \log(2\pi\phi)\right)$	$-\log y!$	1

1. Random component: The response variable $Y|X$ is continuous and normally distributed with a constant variance σ^2 and mean $\mu = \mu(X)$:

$$Y|X \sim \mathcal{N}(\mu(X), \sigma^2 I) \tag{1}$$

   where I is the $n \times n$ identity matrix.

2. Systematic component (link): The expected value of the response variable given the covariates is a linear combination of the parameters and covariates:

$$E(Y|X) = \mu(X) = X\beta \tag{2}$$

   This is equivalent to $E(Y_i|X_i) = \mu_i = X_i^\top \beta$ for each observation i.

These two components of the CLM are restrictive and often not followed by real-world data.

On one hand, the assumption of a continuous, Normal distribution with constant variance is unsuitable for many types of data. Responses can be discrete counts, binary outcomes, or continuous outcomes that are not Normal and have varying variance with the mean .

Moreover, the identity link $\mu = X\beta$ is also a major constraint. The linear predictor $X\beta$ can produce any real value from $-\infty$ to $+\infty$. However, the mean μ of the response is often restricted. For example, a mean for a binary outcome must be a probability in $(0, 1)$, and a mean for a count must be positive ($\mu > 0$). The model can thus predict values that are physically impossible for the response.

2.4 Generalized Linear Models

GLMs generalize normal linear regression models to avoid the limitations above in the following way:

1. Random component: GLMs relax the assumption that the response variable, conditional on the covariates, must be continuous and normally distributed. Instead, GLMs require that the response variable $Y|X$ comes from the exponential family of distributions. This generalization is useful because the exponential family is a broad class that includes the Normal, Poisson, Bernoulli, Gamma, and many other distributions, allowing the model to match the true nature of the data.

2. Systematic component (link): Because the expected value $\mu(X) = \mathbb{E}(Y \mid X)$ may be restricted by the distribution of Y, GLMs introduce a link function $g(\cdot)$ such that

$$g\big(\mu(X)\big) = X\beta.$$

   The link function ensures that the transformed mean can take any real value, making it compatible with the unrestricted linear predictor $X\beta$. This allows GLMs to model different types of response variables while maintaining a linear relationship in the parameters.

   Note that the link function must be monotone and differentiable, since β is the parameter of interest and must appear in the likelihood function for maximum likelihood estimation.

$$\mu = g^{-1}(X\beta)$$

2.4.1 The Canonical Link Function

In some cases, many link functions can be used for a given exponential family distribution. However, for each distribution in the exponential family, there is one special link function called the canonical link. This is the link function $g(\cdot)$ that makes the linear predictor $X\beta$ equal to the canonical parameter θ mentioned in Section 2.2.

$$g(\mu) = \theta$$

Since we know that $\mu = b'(\theta)$, the canonical link is mathematically defined as the inverse of the derivative of the cumulant generating function: $g(\mu) = (b')^{-1}(\mu)$.

Using the canonical link provides the model with desirable statistical properties, such as ensuring that the log-likelihood function is strictly concave, thereby guaranteeing that the maximum likelihood estimate is unique. Moreover, the formulas for model fitting are simplified. Specifically, the Newton-Raphson and Fisher Scoring algorithms for parameter estimation become identical.

While canonical links have these nice properties, non-canonical links are sometimes preferred for practical or interpretability reasons. For example, for data following a binomial distribution (binary response data), the canonical link is the logit function, but the probit and complementary log-log link functions are also commonly used, depending on the data structure and modeling assumptions (see Section 3).

The following table shows the canonical links for common distributions.

Table 2: Canonical Link Functions for Common Distributions

Distribution	Canonical Link $g(\mu)$
Normal	Identity: μ
Poisson	Log: $\log(\mu)$
Bernoulli	Logit: $\log(\mu/(1-\mu))$
Gamma	Reciprocal: $-1/\mu$

2.4.2 Generalized Linear Models Fitting

The parameters β of a GLM are estimated using maximum likelihood estimation (MLE). Each independent response Y_i follows an exponential family distribution:

$$f(y_i; \theta_i, \phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right\}, \tag{3}$$

where θ_i is the canonical parameter, ϕ is the dispersion parameter, and $a(\cdot)$, $b(\cdot)$, $c(\cdot)$ are known functions. Key exponential family properties are: $\mu_i = \mathbb{E}[Y_i] = b'(\theta_i)$ and $\text{Var}(Y_i) = a(\phi)b''(\theta_i)$.

The linear predictor $\eta_i = x_i^\top\beta$ relates to the mean through the link function $g(\mu_i) = \eta_i$, which implies $\mu_i = g^{-1}(\eta_i)$. Since $\mu_i = b'(\theta_i)$, we have $\theta_i = (b')^{-1}(\mu_i) = (b')^{-1}(g^{-1}(x_i^\top\beta))$, establishing θ_i as a function of β.

For n independent observations, the log-likelihood is defined as:

$$\ell(\beta) = \sum_{i=1}^{n} \frac{y_i\theta_i(\beta) - b(\theta_i(\beta))}{a(\phi)} + c(y_i, \phi). \tag{4}$$

If we take the gradient of the log-likelihood with respect to the parameters (score vector), by using the chain rule $\frac{\partial \ell}{\partial \beta_j} = \sum_i \frac{\partial \ell}{\partial \theta_i}\frac{\partial \theta_i}{\partial \mu_i}\frac{\partial \mu_i}{\partial \eta_i}\frac{\partial \eta_i}{\partial \beta_j}$:

$$\frac{\partial \ell}{\partial \theta_i} = \frac{y_i - b'(\theta_i)}{a(\phi)} = \frac{y_i - \mu_i}{a(\phi)}, \quad \frac{\partial \theta_i}{\partial \mu_i} = \frac{1}{b''(\theta_i)}, \quad \frac{\partial \mu_i}{\partial \eta_i} = \frac{d\mu_i}{d\eta_i} = (g')^{-1}(\eta_i), \quad \frac{\partial \eta_i}{\partial \beta_j} = x_{ij}. \tag{5}$$

Combining terms yields the compact matrix form:

$$s(\beta) = \frac{\partial \ell}{\partial \beta} = X^\top W(y - \mu), \tag{6}$$

where $W = \text{diag}(w_i)$ with weights:

$$w_i = \frac{1}{a(\phi)b''(\theta_i)} \left(\frac{d\mu_i}{d\eta_i}\right)^2 = \frac{1}{\text{Var}(Y_i)} \left(\frac{d\mu_i}{d\eta_i}\right)^2. \tag{7}$$

The observed Hessian matrix is:

$$H(\beta) = \frac{\partial^2 \ell}{\partial \beta \partial \beta^\top}. \tag{8}$$

Differentiating equation (6) yields:

$$H(\beta) = -X^\top W X + X^\top \frac{\partial W}{\partial \beta}(y - \mu) - X^\top W \frac{\partial \mu}{\partial \beta}. \tag{9}$$

Moreover, the expected Hessian, also referred to as Fisher Information Matrix is:

$$I(\beta) = -\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \beta \partial \beta^\top}\right] = X^\top W X. \tag{10}$$

The MLE $\hat{\beta}$ satisfies $s(\beta) = 0$. Since closed-form solutions rarely exist, we employ iterative numerical methods. Three closely related second-order algorithms are standard for GLM fitting:

1. Newton-Raphson Method

   Newton-Raphson is a general-purpose optimization algorithm that uses a second-order Taylor approximation of the log-likelihood around the current estimate $\beta^{(t)}$:

   $$\ell(\beta) \approx \ell(\beta^{(t)}) + s(\beta^{(t)})^\top (\beta - \beta^{(t)}) + \frac{1}{2}(\beta - \beta^{(t)})^\top H(\beta^{(t)})(\beta - \beta^{(t)}). \tag{11}$$

   Setting the derivative to zero and solving for β yields the update:

   $$\beta^{(t+1)} = \beta^{(t)} - H^{-1}(\beta^{(t)}) s(\beta^{(t)}). \tag{12}$$

   Main advantages include a quadratic convergence rate near the optimum and the use of actual observed information (exact second derivatives). However, this comes with some disadvantages, including the need to compute the full observed Hessian $H(\beta)$, which is computationally expensive.Moreover, it may fail if $H(\beta^{(t)})$ is not negative definite (non-concave regions). Additionally, it can be numerically unstable for non-canonical links or poor starting values.

2. Fisher Scoring

   Fisher Scoring modifies Newton-Raphson by replacing the observed Hessian with the expected Hessian (Fisher Information):

   $$\beta^{(t+1)} = \beta^{(t)} + I^{-1}(\beta^{(t)}) s(\beta^{(t)}) = \beta^{(t)} + (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} (y - \mu^{(t)}). \tag{13}$$

   Main advantages over Newton-Raphson include that $I(\beta)$ is always positive semi-definite by construction (guaranteed invertibility), and it requires only first derivatives and variance functions. Moreover, it is more globally stable: it works well even far from the optimum. It is important to note that for canonical links: $H(\beta) = I(\beta)$, so Fisher Scoring and Newton-Raphson are identical in this case.

3. Iteratively Reweighted Least Squares (IRLS)

   Fisher Scoring has an elegant interpretation as a weighted least-squares algorithm. Rewriting the Fisher Scoring update:

   $$\begin{aligned} \beta^{(t+1)} &= \beta^{(t)} + (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} (y - \mu^{(t)}) \\ &= (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} \left[X\beta^{(t)} + W^{(t)^{-1}} (y - \mu^{(t)})\right] \\ &= (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} z^{(t)}, \end{aligned} \tag{14}$$

where $z^{(t)}$ is the *working response* (adjusted dependent variable):

$$z_i^{(t)} = \eta_i^{(t)} + (y_i - \mu_i^{(t)}) \left(\frac{d\eta_i}{d\mu_i} \right)_{\mu_i^{(t)}}. \tag{15}$$

For canonical links, where $\eta_i = g(\mu_i)$ and $d\eta_i/d\mu_i = g'(\mu_i)$, this simplifies to

$$z_i^{(t)} = g(\mu_i^{(t)}) + (y_i - \mu_i^{(t)}) g'(\mu_i^{(t)}).$$

The update $\beta^{(t+1)} = (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} z^{(t)}$ is precisely the solution to the weighted least-squares problem.

At each iteration, IRLS constructs pseudo-responses $z^{(t)}$ that linearize the model around the current fit, then computes weights $W^{(t)}$ based on the variance structure and link function and finaly solves a standard weighted least-squares regression of $z^{(t)}$ on X. This provides an intuitive connection between GLMs and ordinary least squares, treating GLM fitting as the iterative solution of weighted linear regressions. Pseudocode for the algorithm is available in Appendix A.

3 Common GLMs and their Link Functions

We illustrate how GLMs are applied to various response data types by selecting appropriate distributions and link functions.

3.1 Continuous Data: Normal Linear Regression

For continuous response variables, the normal distribution with identity link $g(\mu) = \mu$ is standard. This reduces to classical linear regression:

$$\mu_i = x_i^\top \beta$$

or equivalently

$$Y_i = x_i^\top \beta + \varepsilon_i$$

with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.

The coefficient β_j represents the expected change in Y for a one-unit increase in predictor X_j, holding other variables constant. The relationship is additive and linear, making interpretation straightforward. This model is appropriate when the response is continuous and unbounded, residuals are approximately normal, and the relationship between predictors and response is linear.

3.2 Count Data: Poisson Regression

For non-negative count data, the Poisson distribution with log link $g(\mu) = \log(\mu)$ is used. The model specification is $\log(\mu_i) = x_i^\top \beta$, ensuring $\mu_i = \exp(x_i^\top \beta) > 0$.

The coefficient β_j represents the change in log-mean per unit increase in X_j. More intuitively, on the original scale, a one-unit increase in X_j multiplies the expected count by $\exp(\beta_j)$, known as the incidence rate ratio (IRR). For example, if $\beta_j = 0.05$, each unit increase in X_j increases the expected count by approximately 5%. This multiplicative interpretation is essential: the effect is proportional to the current level rather than additive.

3.3 Binary Data: Logistic Regression

For binary responses, the Bernoulli distribution is used with mean $p_i \in (0, 1)$ representing the probability of success. Several link functions map this probability to the real line. The logit link $g(p) = \log(p/(1-p))$ is canonical and most common, yielding the model

$$\log(p_i/(1-p_i)) = x_i^\top \beta$$

The coefficient β_j represents the change in log-odds per unit increase in X_j. Exponentiating gives the odds ratio: $\exp(\beta_j)$ is the multiplicative change in odds of success. For instance, if

$\exp(\beta_j) = 1.5$, a one-unit increase in X_j increases the odds by 50%. Values greater than 1 indicate positive association, less than 1 indicate negative association, and equal to 1 indicate no effect.

The probit link $g(p) = \Phi^{-1}(p)$ is based on an underlying latent normal variable model, where $p_i = \Phi(x_i^\top \beta)$ and Φ is the standard normal CDF. Here β_j represents the change in the z-score of the latent variable, but lacks a simple odds ratio interpretation. This link is preferred when theory suggests an underlying continuous normal process, such as in psychometric or toxicological studies.

The complementary log-log link $g(p) = \log(-\log(1-p))$ gives $p_i = 1 - \exp(-\exp(x_i^\top \beta))$ and is asymmetric, approaching zero faster than one. This makes it suitable for rare events or grouped survival data, where $\exp(\beta_j)$ can be interpreted as a hazard ratio. It arises naturally when modeling time-to-event data discretized into intervals or when the underlying process is a Poisson arrival.

All three links produce similar fitted probabilities for moderate p-values but differ in tail behavior. The logit is generally preferred for its interpretability through odds ratios, the probit for latent variable frameworks, and the complementary log-log for rare events. Model selection criteria, such as AIC, can guide the choice when theory does not clearly indicate a single option.

4 Custom Logistic Regression Implementation

To validate the theoretical framework and demonstrate practical GLM application, we developed a custom logistic regression implementation in Python using NumPy for numerical operations and SciPy for statistical distributions. The implementation follows Algorithm 1 and extends it with comprehensive inference capabilities and automated variable selection.

4.1 Core Features

The `CustomLogisticRegression` class implements the IRLS algorithm with several key features. First, numerical stability is ensured through sigmoid input clipping to $[-500, 500]$ to prevent overflow, and ridge regularization ($10^{-8}I$) is added to the Hessian matrix during the weighted least-squares step to handle near-singular matrices. Convergence is determined when the absolute change in log-likelihood falls below $\epsilon = 10^{-6}$ (by default) or after a maximum of a certain number of iterations.

The implementation computes the variance-covariance matrix as $\text{Var}(\beta) = (X^\top W X)^{-1}$, where W contains the final iteration weights. From this, standard errors are derived as $\text{SE}(\beta_j) = \sqrt{\text{Var}(\beta_j)}$, enabling Wald z-statistics $z_j = \beta_j / \text{SE}(\beta_j)$ and two-tailed p-values for testing $H_0 : \beta_j = 0$. Confidence intervals are constructed as $\beta_j \pm 1.96 \cdot \text{SE}(\beta_j)$, and odds ratios $\exp(\beta_j)$ with their confidence intervals are reported for interpretability.

Model fit is assessed through multiple statistics: log-likelihood, deviance $\mathcal{D} = -2\ell(\beta)$, AIC $= 2k - 2\ell(\beta)$, BIC $= k\log(n) - 2\ell(\beta)$, and McFadden's pseudo-$R^2 = 1 - \ell(\beta)/\ell_{\text{null}}$. A likelihood ratio test comparing the fitted model to an intercept-only null model evaluates global significance.

4.2 Categorical Variable Support

Our approach handles categorical variables through a `feature_groups` parameter. This dictionary maps conceptual variables (e.g., `'Region'`) to their constituent dummy columns (e.g., `['Region_B', 'Region_C']`), following R's GLM convention where the first category serves as the reference level. This structure is crucial for proper variable selection, ensuring that all dummy variables belonging to a categorical feature are retained or removed together, maintaining the interpretability of the categorical predictor.

4.3 Stepwise Variable Selection

The `step()` method implements backward elimination using AIC as the selection criterion, mirroring R's `step()` function behavior. The algorithm iteratively evaluates removing each feature group (not individual columns) and selects the group whose removal yields the lowest AIC. For a model with k parameters and log-likelihood ℓ, AIC is computed as $2k - 2\ell$. The process terminates when no further removal improves AIC, returning a new fitted model instance with the selected features. This approach differs from typical wrapper methods by: (1) operating on feature groups

rather than individual coefficients, enabling proper handling of categorical variables with multiple dummy columns, and (2) returning a new fitted model object rather than modifying the original, following R's functional programming style.

A `summary()` method formats all inference statistics in tabular form with significance stars, mimicking standard statistical software output.

5 Experimentation

5.1 Data description

The dataset used in this study is the UCI Heart Disease dataset [3], which contains clinical data from four sources: Cleveland, Hungary, Switzerland, and the VA Long Beach. However, we only included the Cleveland source, since it is the one that has been used in other research studies, and we can test our implementation against the previous works. The used dataset comprises 303 patient records with 14 commonly used attributes, selected from an original set of 76. Each record contains a mixture of numerical and categorical features, including demographic information, clinical measurements, and exercise test results. The target variable indicates the presence of heart disease on a scale from 0 (no disease) to 4 (presence of disease), which is simplified to a binary classification task: 0 = absence, 1 = presence of heart disease.

Table 3: Dataset characteristics

Property	**Value**
Number of observations	303
Number of features	14
Feature types	6 continuous, 8 categorical
Number of classes	2
Class distribution	Slightly Imbalanced
Missing values	Yes

5.2 Preprocessing

A thorough data preprocessing process is included in the Preprocessing.ipynb notebook (*/Experiments/CustomLogRegValidation/Preprocessing.ipynb*). The detailed justification of all decisions is explained in the notebook, but we summarize it here also:

First, **missing values**, represented by '?' in the raw data, were identified in the `ca` (4 instances) and `thal` (2 instances) columns. To handle these, a cluster-based imputation was chosen over simple global imputation for its robustness. A K-Means clustering algorithm ($k = 3$) was fit on the features (excluding the target) to group similar samples. Each missing value was then imputed using the mode of its feature within its assigned cluster, resulting in the `ca` values being imputed to 0.0 and `thal` values to 3.0.

Second, **outlier treatment** was performed to reduce the skewing effect of extreme measurements. Outliers were detected using the $1.5 \times$ IQR rule. To manage skewness, a Box-Cox transformation was applied to the `thalach` column (resolving its single outlier), and a log transformation was applied to `trestbps` and `chol`. As the transformations did not resolve all outliers in `trestbps` and `chol`, the remaining outlier samples (9 and 5, respectively) were removed. Outliers in binary features like `cp` and `fbs` were intentionally retained, as they represent valid categorical distinctions rather than measurement errors.

Third, **no feature scaling** was performed. This decision preserves the clinical interpretability of the model's coefficients, since keeping predictors in their original units allows odds ratios to be interpreted directly. Although scaling can improve numerical conditioning, it does not change the fitted probabilities of a logistic regression model; it only changes the magnitudes of the estimated coefficients, so avoiding standardization is appropriate when interpretability is prioritized.

Fourth, **categorical variables** were encoded to provide numerical inputs for the model. Two separate datasets were created for this purpose: one for the custom Python implementation, using

k-1 dummy variables via one-hot encoding (`pd.get_dummies` with `drop_first=True`) to avoid multicollinearity; and a second dataset for R's `glm()`, where columns were simply converted to the 'category' type, allowing R to handle the factor encoding internally.

Finally, **no feature selection** or extraction was performed during this preprocessing stage. This step was intentionally deferred, as the objective is to perform selection later using the custom-implemented `step()` (stepwise selection) method on the full set of prepared predictors.

5.3 Custom Implementation Validation

The initial step in evaluating our custom logistic regression implementation is to validate it against R's built-in glm() function. To ensure a fair comparison, both methods were applied to the same preprocessed dataset. The results of this comparison are presented below, demonstrating the correctness and consistency of our implementation.

We fit the "full" model on the cleaned dataset. As a result, the validation was successful, demonstrating a complete match in results:

- **Full Model**: Our custom logistic regression implementation produced an AIC of **224.8201**, which is the identical score to R's AIC of **224.82**.
- **Stepwise Selection**: We then applied the `step()` method (which mirrors R's backward elimination). Our algorithm perfectly replicated R's selection process of features, dropping `restecg`, `age`, and `fbs` in the same order to yield the same final model.
- **Final Model**: Our final selected model yielded an AIC of **219.1728**, matching R's final AIC of **219.17**.
- **Coefficients**: The final coefficient array from our Python model was numerically identical to the `coef(step_model)` output from R, confirming the correctness of our IRLS fitting procedure.

Full screenshots of the validation (R and Python outputs) can be found in Appendix B.

This one-to-one correspondence confirms that our from-scratch implementation of GLM fitting and selection algorithms is as precise and reliable as the industry-standard routines used in R.

5.4 Experimentation

The next step in our experimentation involves applying the validated logistic regression implementation to the dataset described earlier.

This section includes performing stepwise variable selection to identify the most relevant predictors and interpreting the results obtained from the resulting optimal model. After validating our implementation, we applied our step() method (configured for backward elimination as described in Section 4.3) to the whole model. The algorithm iteratively removed the least significant predictors (`restecg`, `age`, and `fbs`, in that order) to arrive at a final, optimized model with an AIC of **219.17**.

This final model allows for a clear interpretation of the most significant risk factors for heart disease in the Cleveland dataset.

Before interpreting individual predictors, it is useful to recall the relationship between the logistic regression coefficients and the odds of the outcome. In logistic regression, the log-odds of the event (here, heart disease) is modeled as a linear combination of the predictors:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k,$$

where p is the probability of heart disease, β_i are the estimated coefficients, and X_i are the predictors.

To make these coefficients more interpretable in a clinical context, we exponentiate them to obtain the odds ratios:

$$\text{Odds Ratio for } X_i = e^{\beta_i}.$$

An odds ratio greater than 1 indicates that an increase in the predictor increases the odds of heart disease, while an odds ratio less than 1 indicates a protective effect. This approach allows for a direct and clinically meaningful interpretation of the impact of each predictor on the probability of the outcome.

This interpretability is a primary strength of GLMs in a clinical context. We interpret the predictors with the highest statistical significance ($p < 0.01$) by examining their coefficients and corresponding odds ratios:

- `ca` (Number of major vessels): This was the most significant predictor. Its coefficient of 1.177 ($p < 0.001$) corresponds to an odds ratio of **3.24** ($exp(1.177)$). This implies that for each additional vessel colored during fluoroscopy, a patient's odds of having heart disease more than triple, holding other factors constant.
- `thal` (Thallium stress test): A result of 'reversible defect' (`thal7`) was highly significant. Its coefficient of 1.352 ($p = 0.0013$) yields an odds ratio of **3.87**. This suggests a patient with a reversible defect has nearly 3.9 times the odds of having heart disease compared to a patient with a baseline test result.
- `cp` (Chest Pain Type): The 'asymptomatic' category (`cp4`) was a strong predictor with a coefficient of 1.949 ($p = 0.0027$) and an odds ratio of **7.02**. This indicates that, compared to the baseline 'typical angina' (`cp1`), patients presenting as asymptomatic (for chest pain type) had 7 times the odds of having heart disease.
- `sex` (Gender): The `sex1` (male) predictor has a coefficient of 1.506 ($p = 0.0034$), for an odds ratio of **4.51**. In this dataset, males had over 4.5 times the odds of heart disease as females (the baseline category), all else being equal.

This analysis demonstrates that logistic regression not only fits the data accurately but also provides the actionable, interpretable results that are highly valued in biomedical applications.

6 Conclusions

This project contains the complete derivation of Generalized Linear Models from first principles, culminating in a robust and fully functional CustomLogisticRegression implementation in Python. By deriving the model from the theoretical foundations of the exponential family and implementing the Iteratively Reweighted Least Squares algorithm, we developed a tool with comprehensive statistical inference capabilities, including odds ratios, p-values, and AIC.

A key part of the project was the empirical validation of our implementation against R's glm() and step() functions. Our model perfectly replicated R's full-model AIC, stepwise selection trace, final model AIC, and final coefficients, confirming the correctness of our fitting and selection algorithms.

Applying this validated model to the UCI Heart Disease dataset further demonstrated its practical utility. The final model identified and quantified significant clinical predictors (such as number of major vessels, thalium stress, chest pain type, and sex) by interpreting their odds ratios. This work successfully bridges the gap between GLM theory and practical application, resulting in an accurate, interpretable, and fully validated tool for real-world data analysis.

Overall, we saw how GLMs are theoretically built and how useful GLMs are by generalizing the more limited classical linear models, allowing modeling of different types of data, and providing highly interpretable results, which is crucial in fields like medicine.

References

[1] Peter McCullagh. *Generalized linear models*. Routledge, 2019.

[2] Philippe Rigollet. Statistics for applications (18.650). MIT OpenCourseWare, 2016.

[3] UCI Machine Learning Repository. Heart disease data set. https://archive.ics.uci.edu/dataset/45/heart+disease, 1988. Accessed: 2025-11-16.

A IRLS

Algorithm 1 Iteratively Reweighted Least Squares (IRLS) for GLM Fitting

Require: $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$, link function $g(\cdot)$, variance function $V(\mu)$, tolerance $\epsilon > 0$, maximum iterations $T_{\max}$
Ensure: Estimated coefficients $\hat{\beta} \in \mathbb{R}^p$

Initialize: Set $\beta^{(0)}$
Compute initial linear predictors: $\eta^{(0)} \leftarrow X\beta^{(0)}$
Compute initial fitted means: $\mu^{(0)} \leftarrow g^{-1}(\eta^{(0)})$
Set iteration counter: $t \leftarrow 0$
repeat
  // Step 1: Compute weights
  for $i = 1$ to n **do**
    $w_i^{(t)} \leftarrow \frac{1}{V(\mu_i^{(t)})} \left(\frac{d\mu_i}{d\eta_i}\Big|_{\mu_i^{(t)}} \right)^2 = \frac{1}{V(\mu_i^{(t)}) \cdot [g'(\mu_i^{(t)})]^2}$
  end for
  $W^{(t)} \leftarrow \mathrm{diag}(w_1^{(t)}, \ldots, w_n^{(t)})$
  // Step 2: Compute working response
  for $i = 1$ to n **do**
    $z_i^{(t)} \leftarrow \eta_i^{(t)} + (y_i - \mu_i^{(t)}) \cdot g'(\mu_i^{(t)})$
  end for
  // Step 3: Solve weighted least squares
  $\beta^{(t+1)} \leftarrow (X^\top W^{(t)} X)^{-1} X^\top W^{(t)} z^{(t)}$
  // Step 4: Update fitted values
  $\eta^{(t+1)} \leftarrow X\beta^{(t+1)}$
  $\mu^{(t+1)} \leftarrow g^{-1}(\eta^{(t+1)})$
  // Step 5: Check convergence
  $\Delta \leftarrow \|\beta^{(t+1)} - \beta^{(t)}\|_2$
  $t \leftarrow t + 1$
until $\Delta < \epsilon$ or $t \geq T_{\max}$
if $t \geq T_{\max}$ **then**
  Warning: Algorithm did not converge
end if
return $\hat{\beta} \leftarrow \beta^{(t)}$

Convergence criteria. Common stopping rules include:

- Parameter change: $\|\beta^{(t+1)} - \beta^{(t)}\|_2 < \epsilon$ (typically $\epsilon = 10^{-6}$).
- Deviance change: $|\mathcal{D}(\beta^{(t+1)}) - \mathcal{D}(\beta^{(t)})| < \epsilon$, where $\mathcal{D}(\beta) = -2\ell(\beta)$.
- Gradient norm: $\|s(\beta^{(t+1)})\|_2 < \epsilon$.

When using the IRLS algorithm, there are some common practical considerations:

- Starting values: Good initialization accelerates convergence. Common choices: $\beta^{(0)} = 0$ or fit from linear regression.
- Numerical stability: Add small ridge λI to $X^\top W X + \lambda I$ if matrix is near-singular.

B Custom Logistic Regression Validation

Following figures show the key points of the Custom Logistic Regression Validation:

```
> summary(full_model)

Call:
glm(formula = target ~ age + sex + cp + trestbps + chol + fbs +
    restecg + thalach + exang + oldpeak + slope + ca + thal,
    family = binomial, data = data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.739e+01  1.017e+01  -2.693  0.00707 **
age         -1.180e-02  2.517e-02  -0.469  0.63923
sex1         1.482e+00  5.285e-01   2.803  0.00506 **
cp2          8.306e-01  7.655e-01   1.085  0.27790
cp3          7.025e-02  6.523e-01   0.108  0.91423
cp4          1.916e+00  6.583e-01   2.910  0.00361 **
trestbps     3.183e+00  1.669e+00   1.906  0.05659 .
chol         1.655e+00  1.197e+00   1.383  0.16680
fbs1        -4.667e-01  6.013e-01  -0.776  0.43764
restecg1     3.812e-01  2.932e+00   0.130  0.89654
restecg2     4.746e-01  3.857e-01   1.230  0.21852
thalach     -4.024e-05  2.394e-05  -1.680  0.09288 .
exang1       7.283e-01  4.528e-01   1.609  0.10769
oldpeak      4.385e-01  2.392e-01   1.833  0.06684 .
slope2       9.589e-01  4.689e-01   2.045  0.04084 *
slope3       8.509e-01  9.115e-01   0.933  0.35058
ca           1.228e+00  2.783e-01   4.413 1.02e-05 ***
thal6       -1.099e-01  7.947e-01  -0.138  0.88998
thal7        1.408e+00  4.339e-01   3.244  0.00118 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 395.69  on 286  degrees of freedom
Residual deviance: 186.82  on 268  degrees of freedom
AIC: 224.82

Number of Fisher Scoring iterations: 6
```

(a) Full model summary - R

```
Coefficients (with 95% Confidence Intervals):
================================================================================================
Variable              Coef    Std.Err        z      P>|z|     [0.025     0.975]
------------------------------------------------------------------------------------------------
Intercept          -26.7479     9.7965   -2.730     0.0063   -45.9486    -7.5472  **
trestbps             2.8936     1.5842    1.827     0.0678    -0.2113     5.9985   .
chol                 1.7062     1.1540    1.479     0.1393    -0.5555     3.9680
thalach             -0.0000     0.0000   -1.740     0.0818    -0.0001     0.0000   .
oldpeak              0.4397     0.2318    1.897     0.0578    -0.0145     0.8940   .
ca                   1.1767     0.2606    4.516     0.0000     0.6660     1.6874 ***
cp_2.0               0.7906     0.7589    1.042     0.2975    -0.6967     2.2780
cp_3.0               0.0465     0.6454    0.072     0.9426    -1.2184     1.3113
cp_4.0               1.9493     0.6491    3.003     0.0027     0.6770     3.2215  **
sex_1.0              1.5059     0.5135    2.933     0.0034     0.4995     2.5123  **
exang_1.0            0.7033     0.4470    1.573     0.1157    -0.1729     1.5795
slope_2.0            0.9452     0.4589    2.060     0.0394     0.0457     1.8446   *
slope_3.0            0.7884     0.8860    0.890     0.3736    -0.9482     2.5249
thal_6.0            -0.2374     0.7734   -0.307     0.7588    -1.7533     1.2784
thal_7.0             1.3521     0.4212    3.210     0.0013     0.5266     2.1776  **
------------------------------------------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(b) Full model summary - Custom LogReg Validation

Figure 1: Full Model Comparison R and our (Python) validation.

```
> step_model <- step(full_model, direction = "backward", trace = TRUE)
Start:  AIC=224.82
target ~ age + sex + cp + trestbps + chol + fbs + restecg + thalach +
    exang + oldpeak + slope + ca + thal

           Df Deviance    AIC
- restecg   2   188.35 222.35
- age       1   187.04 223.04
- fbs       1   187.43 223.43
- chol      1   188.77 224.77
<none>          186.82 224.82
- slope     2   191.09 225.09
- exang     1   189.38 225.38
- thalach   1   189.72 225.72
- oldpeak   1   190.35 226.35
- trestbps  1   190.55 226.55
- sex       1   195.31 231.31
- thal      2   199.06 233.06
- cp        3   204.33 236.33
- ca        1   211.12 247.12
```

Step 1 - R

```
Step:  AIC=222.35
target ~ age + sex + cp + trestbps + chol + fbs + thalach + exang +
    oldpeak + slope + ca + thal

           Df Deviance    AIC
- age       1   188.54 220.54
- fbs       1   188.97 220.97
<none>          188.35 222.35
- chol      1   190.86 222.86
- exang     1   190.95 222.95
- slope     2   193.18 223.18
- thalach   1   191.41 223.41
- oldpeak   1   191.57 223.57
- trestbps  1   192.28 224.28
- sex       1   197.72 229.72
- thal      2   199.81 229.81
- cp        3   205.67 233.67
- ca        1   213.24 245.24
```

Step 2 - R

```
Step:  AIC=220.55
target ~ sex + cp + trestbps + chol + fbs + thalach + exang +
    oldpeak + slope + ca + thal

           Df Deviance    AIC
- fbs       1   189.17 219.17
<none>          188.54 220.54
- chol      1   190.95 220.95
- exang     1   191.28 221.28
- slope     2   193.29 221.29
- thalach   1   191.47 221.47
- oldpeak   1   191.87 221.87
- trestbps  1   192.28 222.28
- thal      2   200.04 228.04
- sex       1   198.38 228.38
- cp        3   206.06 232.06
- ca        1   214.34 244.34
```

Step 3 - R

```
Step:  AIC=219.17
target ~ sex + cp + trestbps + chol + thalach + exang + oldpeak +
    slope + ca + thal

           Df Deviance    AIC
<none>          189.17 219.17
- chol      1   191.41 219.41
- slope     2   193.50 219.50
- exang     1   191.62 219.62
- thalach   1   192.30 220.30
- trestbps  1   192.59 220.59
- oldpeak   1   192.93 220.93
- sex       1   198.51 226.51
- thal      2   201.58 227.58
- cp        3   208.47 232.47
- ca        1   214.35 242.35
```

Step 4 - R

(a) R backward elimination trace

```
Start:  AIC=224.8201
        Variables: Intercept, cp, sex, fbs, restecg, exang, slope, thal, age, trestbps, chol, thalach, oldpeak, ca
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step:   AIC=222.3511  Dropping Group: restecg
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step:   AIC=220.5454  Dropping Group: age
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step:   AIC=219.1728  Dropping Group: fbs
--------------------------------------------------------------------------------

================================================================================
Stepwise selection finished (no further AIC improvement).
Final Model AIC: 219.1728
Final Variables: Intercept, cp, sex, exang, slope, thal, trestbps, chol, thalach, oldpeak, ca
================================================================================
```

(b) Custom Python backward elimination trace

Figure 2: Comparing the backward elimination trace of R and our (Python) implementation.

```
> summary(step_model)

Call:
glm(formula = target ~ sex + cp + trestbps + chol + thalach +
    exang + oldpeak + slope + ca + thal, family = binomial, data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.675e+01  9.797e+00  -2.730  0.00633 **
sex1         1.506e+00  5.135e-01   2.933  0.00336 **
cp2          7.906e-01  7.589e-01   1.042  0.29748
cp3          4.646e-02  6.454e-01   0.072  0.94261
cp4          1.949e+00  6.491e-01   3.003  0.00267 **
trestbps     2.894e+00  1.584e+00   1.827  0.06777 .
chol         1.706e+00  1.154e+00   1.479  0.13927
thalach     -3.850e-05  2.212e-05  -1.740  0.08185 .
exang1       7.033e-01  4.471e-01   1.573  0.11568
oldpeak      4.397e-01  2.318e-01   1.897  0.05778 .
slope2       9.452e-01  4.589e-01   2.060  0.03944 *
slope3       7.884e-01  8.860e-01   0.890  0.37358
ca           1.177e+00  2.606e-01   4.516 6.31e-06 ***
thal6       -2.374e-01  7.734e-01  -0.307  0.75885
thal7        1.352e+00  4.212e-01   3.210  0.00133 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 395.69  on 286  degrees of freedom
Residual deviance: 189.17  on 272  degrees of freedom
AIC: 219.17

Number of Fisher Scoring iterations: 6

>
> # Extract coefficients for comparison with your Python implementation
> coef(step_model)
  (Intercept)          sex1           cp2           cp3           cp4      trestbps          chol
-2.674796e+01  1.505892e+00  7.906335e-01  4.645639e-02  1.949263e+00  2.893614e+00  1.706227e+00
      thalach        exang1       oldpeak        slope2        slope3            ca         thal6
-3.849603e-05  7.032951e-01  4.397302e-01  9.451510e-01  7.883728e-01  1.176698e+00 -2.374375e-01
        thal7
 1.352073e+00
```

(a) Final model summary and all coefficients - R

```
Coefficients (with 95% Confidence Intervals):
================================================================================================
Variable              Coef    Std.Err        z      P>|z|     [0.025     0.975]
------------------------------------------------------------------------------------------------
Intercept          -26.7479     9.7965   -2.730     0.0063   -45.9486    -7.5472  **
trestbps             2.8936     1.5842    1.827     0.0678    -0.2113     5.9985   .
chol                 1.7062     1.1540    1.479     0.1393    -0.5555     3.9680
thalach             -0.0000     0.0000   -1.740     0.0818    -0.0001     0.0000   .
oldpeak              0.4397     0.2318    1.897     0.0578    -0.0145     0.8940   .
ca                   1.1767     0.2606    4.516     0.0000     0.6660     1.6874 ***
cp_2.0               0.7906     0.7589    1.042     0.2975    -0.6967     2.2780
cp_3.0               0.0465     0.6454    0.072     0.9426    -1.2184     1.3113
cp_4.0               1.9493     0.6491    3.003     0.0027     0.6770     3.2215  **
sex_1.0              1.5059     0.5135    2.933     0.0034     0.4995     2.5123  **
exang_1.0            0.7033     0.4470    1.573     0.1157    -0.1729     1.5795
slope_2.0            0.9452     0.4589    2.060     0.0394     0.0457     1.8446   *
slope_3.0            0.7884     0.8860    0.890     0.3736    -0.9482     2.5249
thal_6.0            -0.2374     0.7734   -0.307     0.7588    -1.7533     1.2784
thal_7.0             1.3521     0.4212    3.210     0.0013     0.5266     2.1776  **
------------------------------------------------------------------------------------------------
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(b) Final model summary and all coefficients - Python LogReg Validation

Figure 3: Final Model and Coefficents Comparison - R and Python