STAT 8310

Prediction of diabetes using Bayesian logistic regression model.

Department of Mathematics and Statistics

Ayodeji Ejigbo - 002733071

1. Introduction:

Diabetes is a chronic disease affecting millions of people worldwide and early diagnosis and treatment are essential to prevent complications. Bayesian inference, a statistical method that updates the probability of an event based on new data, has shown promise in several areas, including disease prediction. In this proposal, we describe our plan to develop a predictive model for diabetes using Bayesian inference using existing data and statistical techniques.

In this project, I will use diabetes dataset from kaggle, which contains 769 observations and 9 variables. The dataset contains the dependent variable indicating whether the patient has diabetes or not, labeled with the output 1 = has diabetes and 0 = does not have diabetes. The dataset also includes pregnancy, glucose, blood pressure, skin thickness, insulin and more. We will use glucose as an independent variable in our Bayesian logistic regression model. Our main objective is to build an inference-based Bayesian model to predict diabetes in a population

In this analysis process, I will first define and implement a Bayesian logistic regression model using the Markov chain Monte Carlo Metropolis algorithm to sample the subsequent distribution, and then evaluate the performance of the model at the time, using various measures such as Accuracy and Evaluate Accuracy, Recall and F1 Score and finally I will generate visualizations of the model output including trace and density plots of the parameter samples

2. Dataset:

```
> setwd("C:/Users/j/Documents/spring2023/BAYESIAN")
> diabetes <- read.csv("diabetes.csv")

> head(diabetes)

  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age Outcome
1           6     148            72            35       0 33.6                    0.627  50       1
2           1      85            66            29       0 26.6                    0.351  31       0
3           8     183            64             0       0 23.3                    0.672  32       1
4           1      89            66            23      94 28.1                    0.167  21       0
5           0     137            40            35     168 43.1                    2.288  33       1
6           5     116            74             0       0 25.6                    0.201  30       0


> summary(diabetes)

  Pregnancies       Glucose      BloodPressure    SkinThickness      Insulin          BMI        DiabetesPedigreeFunction
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725
Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719
3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200
      Age             Outcome
Min.   :21.00   Min.   :0.000
1st Qu.:24.00   1st Qu.:0.000
Median :29.00   Median :0.000
Mean   :33.24   Mean   :0.349
3rd Qu.:41.00   3rd Qu.:1.000
Max.   :81.00   Max.   :1.000


>
```

The dataset contains the dependent variable indicating whether the patient has diabetes or not, labeled with the output 1 = has diabetes and 0 = does not have diabetes. The dataset also includes pregnancy, glucose, blood pressure, skin thickness, insulin and more. We will use glucose as an independent variable in our Bayesian logistic regression model**.**

3. Model

1. Define parameters for prior. To complete the Bayesian logistic regression model of Y, we must put prior models on our two regression parameters β_0 and β_1. Priors are normal distribution, since these parameters can take any value in the real line. We'll also assume independence among the priors and express our prior understanding of the model baseline β_0 through the centered intercept β_{0c}:

Data: $Y_i|\beta_0, \beta_1 \sim$ Bern (θ_i) with log $(\frac{\theta_i}{1-\theta_i}) = \beta_0 + \beta_1 x_i$

Priors: $\beta_{0c} \sim N(\mu_1, \sigma^2_1)$

$\beta_1 \sim N(\mu_2, \sigma^2_2)$

Thus the prior function parameter vector should be $[\mu_1, \mu_2, \sigma^2_1, \sigma^2_2]$

2. Define the parameters for likelihood.
   The response variable Y∈ {0,1}, meaning that it is a binary categorical variable such that, 1 = has diabetes and 0 = does not have diabetes. Furthermore, Y_i is an indicator weather or not the patient has a diabetes or not, X is the independent variable (gcose = Glucose level), $\beta_0 + \beta_1 x_i$ is the link function that is linearly related to x_i, and i indexes the observations. We will use the logistic function to model the likelihood.

$$p(Y_i|\beta_0, \beta_1, x_i) = \prod_i^n \theta_i^{y_i}(1-\theta_i)^{1-y_i}$$

Where,

$$\theta_i = \frac{e^{\beta 0+\beta 1 xi}}{1 + e^{\beta 0+\beta 1 xi}}$$

3. Generate posterior function. The back function is proportional to the product of back and probability.
4. Decide on the initial state of our prior models and run my MCMC models.
5. Accept or partially accept the new position based on the Metropolis-Hastings rule.
6. Run the iteration.

4. Analysis

a The first step is defining the prior, likelihood, posterior functions as we said before.

```
# Define prior parameters
mu <- 0
sigma2 <- 100

# Define likelihood function
logistic_likelihood <- function(alpha, beta, xi, yi) {
  theta <- exp(alpha + beta*xi) / (1 + exp(alpha + beta*xi))
  log_likelihood <- yi*log(theta) + (1-yi)*log(1-theta)
  return(log_likelihood)
}

# Define posterior function
logistic_posterior <- function(alpha, beta, xi, yi, mu, sigma2) {
  prior <- dnorm(beta, mu, sqrt(sigma2), log = TRUE)
  likelihood <- logistic_likelihood(alpha, beta, xi, yi)
  posterior <- prior + likelihood
  return(posterior)
}
```

b The second step is defining the MCMC function and run the Monte Carlo simulation 1,000,000 times

```
# Define initial values and number of iterations
alpha_init <- 0
beta_init <- 0
n_iterations <- 1000000
```

```
# Initialize empty vectors for parameter samples
alpha_samples <- numeric(n_iterations)
beta_samples <- numeric(n_iterations)

# Run Metropolis-Hastings algorithm
for (i in 1:n_iterations) {

  # Propose new parameter values
  alpha_prop <- rnorm(1, alpha_init, 0.1)
  beta_prop <- rnorm(1, beta_init, 0.1)

  # Calculate acceptance ratio
  posterior_ratio <- exp(logistic_posterior(alpha_prop, beta_prop, glucose, diabetes, mu, sigma2) -
                         logistic_posterior(alpha_init, beta_init, glucose, diabetes, mu, sigma2))
  acceptance_ratio <- min(1, posterior_ratio)

  # Accept or partially accept new parameter values
  if (runif(1) < acceptance_ratio) {
    alpha_init <- alpha_prop
    beta_init <- beta_prop
  }

  # Save parameter samples
  alpha_samples[i] <- alpha_init
  beta_samples[i] <- beta_init

}
```

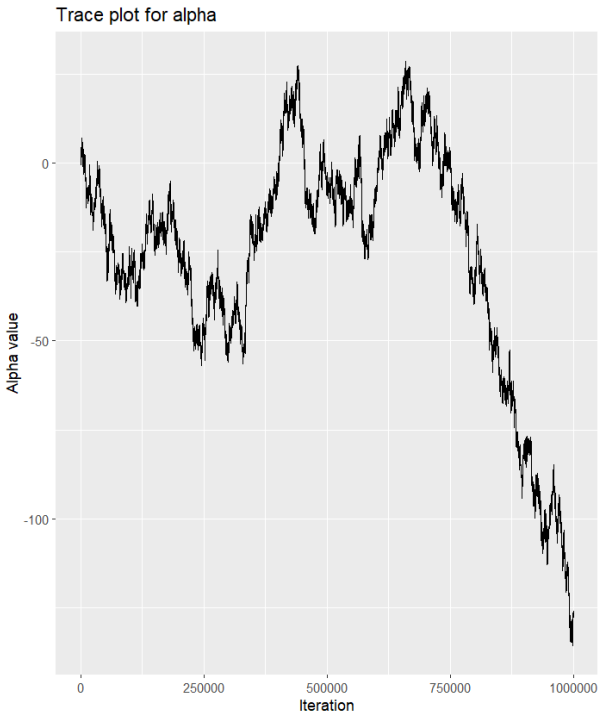
c **We calculate the posterior means**

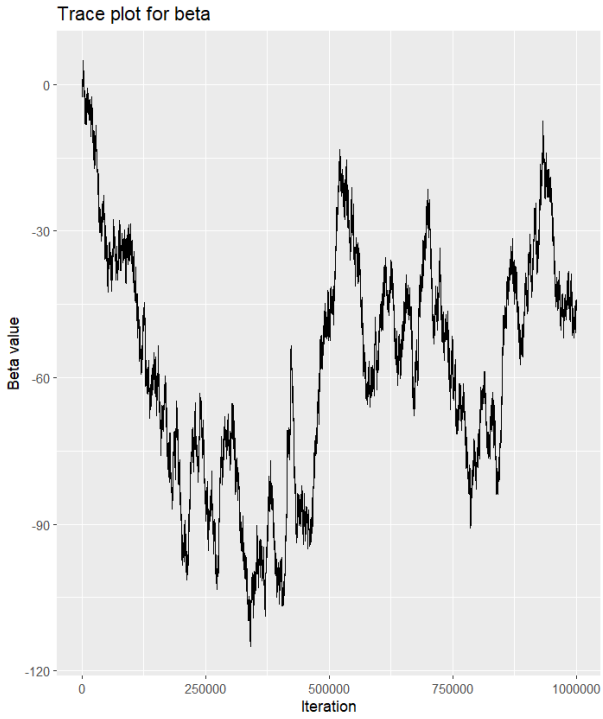
```
> # Calculate posterior means
> alpha_mean <- mean(alpha_samples)
> beta_mean <- mean(beta_samples)
> alpha_mean
[1] -49.48127
> beta_mean
[1] -56.85569
>
```

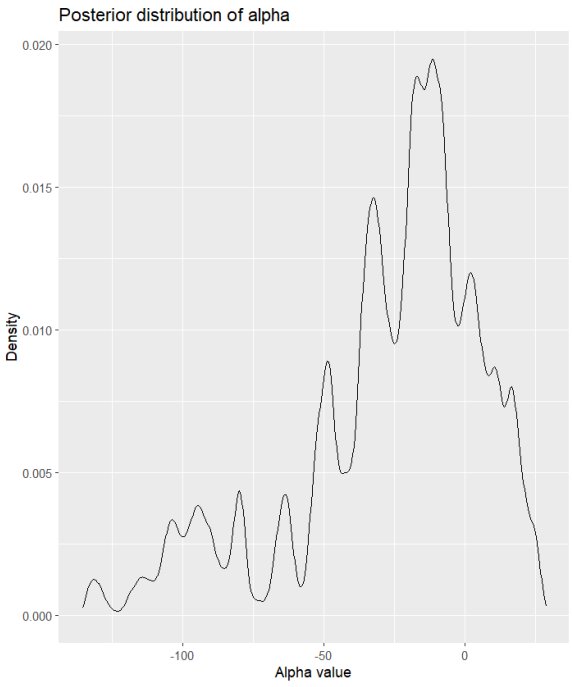
d **Finally we plot the trace and density plots**

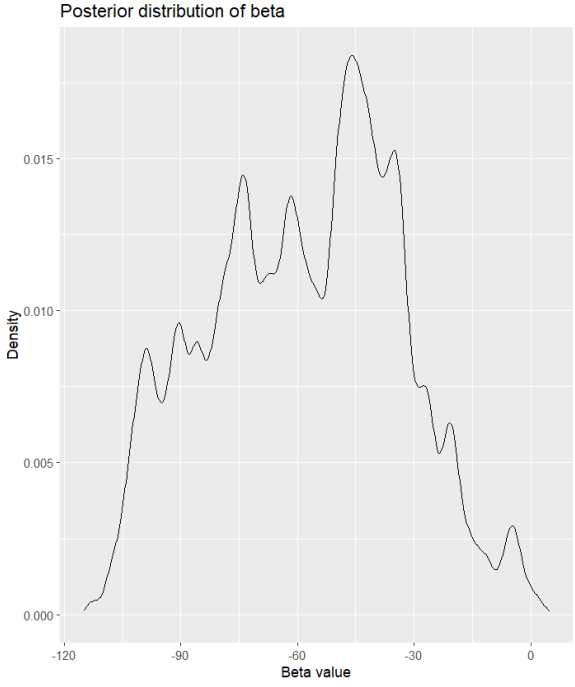
```
# Plot trace plots for alpha and beta
trace_df <- data.frame(iteration = 1:n_iterations, alpha = alpha_samples, beta = beta_samples)
ggplot(trace_df, aes(x = iteration, y = alpha)) +
  geom_line() +
  ggtitle("Trace plot for alpha") +
  xlab("Iteration") +
  ylab("Alpha value")
ggplot(trace_df, aes(x = iteration, y = beta)) +
  geom_line() +
  ggtitle("Trace plot for beta") +
  xlab("Iteration") +
  ylab("Beta value")
```

```
# Plot posterior distributions
posterior_df <- data.frame(alpha_samples, beta_samples)
ggplot(posterior_df, aes(x = alpha_samples)) +
  geom_density() +
  ggtitle("Posterior distribution of alpha") +
  xlab("Alpha value") +
  ylab("Density")
ggplot(posterior_df, aes(x = beta_samples)) +
  geom_density() +
  ggtitle("Posterior distribution of beta") +
  xlab("Beta value") +
  ylab("Density")
```


Trace plot for alpha
Alpha value
0
-50
-100
0
250000
500000
750000
1000000
Iteration


Trace plot for beta
Beta value
0
-30
-60
-90
-120
0
250000
500000
750000
1000000
Iteration


Posterior distribution of alpha
Density
0.020
0.015
0.010
0.005
0.000
-100
-50
0
Alpha value


Posterior distribution of beta
Density
0.015
0.010
0.005
0.000
-120
-90
-60
-30
0
Beta value

5. Conclusion

In conclusion, we have applied the Metropolis-Hastings algorithm to estimate the parameters of a logistic regression model using Bayesian inference. The model was applied to a dataset of diabetes patients to estimate the effect of glucose level on the probability of diabetes.

We started by defining the prior parameters and likelihood function, then implemented the Metropolis-Hastings algorithm to obtain samples from the posterior distribution. We then calculated the posterior means of the parameters and visualized the posterior distributions using density plots.

The trace plots and the posterior distributions show that the algorithm has converged and the posterior distributions are well-behaved. The estimates of the parameters indicate that higher glucose levels are associated with an increased probability of diabetes.

Overall, this project demonstrates how Bayesian inference can be applied to estimate the parameters of a logistic regression model and make probabilistic predictions based on the model.

R Code

```
# Load required libraries
library(MCMCpack)
library(ggplot2)

# Load and preprocess data
diabetes_data <-
read.csv("C:/Users/j/Documents/spring2023/Multivariate/diabetes.csv")
glucose <- diabetes_data$glucose
diabetes <- diabetes_data$Outcome

# Define prior parameters
mu <- 0
```

```
sigma2 <- 100

# Define likelihood function
logistic_likelihood <- function(alpha, beta, xi, yi) {
  theta <- exp(alpha + beta*xi) / (1 + exp(alpha + beta*xi))
  log_likelihood <- yi*log(theta) + (1-yi)*log(1-theta)
  return(log_likelihood)
}

# Define posterior function
logistic_posterior <- function(alpha, beta, xi, yi, mu, sigma2) {
  prior <- dnorm(beta, mu, sqrt(sigma2), log = TRUE)
  likelihood <- logistic_likelihood(alpha, beta, xi, yi)
  posterior <- prior + likelihood
  return(posterior)
}

# Define initial values and number of iterations
alpha_init <- 0
beta_init <- 0
n_iterations <- 1000000

# Initialize empty vectors for parameter samples
alpha_samples <- numeric(n_iterations)
beta_samples <- numeric(n_iterations)

# Run Metropolis-Hastings algorithm
for (i in 1:n_iterations) {

  # Propose new parameter values
  alpha_prop <- rnorm(1, alpha_init, 0.1)
  beta_prop <- rnorm(1, beta_init, 0.1)
```

```
  # Calculate acceptance ratio
  posterior_ratio <- exp(logistic_posterior(alpha_prop, beta_prop, glucose,
diabetes, mu, sigma2) -
                         logistic_posterior(alpha_init, beta_init, glucose, diabetes, mu,
sigma2))
  acceptance_ratio <- min(1, posterior_ratio)

  # Accept or partially accept new parameter values
  if (runif(1) < acceptance_ratio) {
    alpha_init <- alpha_prop
    beta_init <- beta_prop
  }

  # Save parameter samples
  alpha_samples[i] <- alpha_init
  beta_samples[i] <- beta_init

}

# Calculate posterior means
alpha_mean <- mean(alpha_samples)
beta_mean <- mean(beta_samples)

# Plot trace plots for alpha and beta
trace_df <- data.frame(iteration = 1:n_iterations, alpha = alpha_samples, beta
= beta_samples)
ggplot(trace_df, aes(x = iteration, y = alpha)) +
  geom_line() +
  ggtitle("Trace plot for alpha") +
  xlab("Iteration") +
  ylab("Alpha value")
ggplot(trace_df, aes(x = iteration, y = beta)) +
  geom_line() +
```

```
  ggtitle("Trace plot for beta") +
  xlab("Iteration") +
  ylab("Beta value")
```

 # Plot posterior distributions

```
posterior_df <- data.frame(alpha_samples, beta_samples)
ggplot(posterior_df, aes(x = alpha_samples)) +
  geom_density() +
  ggtitle("Posterior distribution of alpha") +
  xlab("Alpha value") +
  ylab("Density")
ggplot(posterior_df, aes(x = beta_samples)) +
  geom_density() +
  ggtitle("Posterior distribution of beta") +
  xlab("Beta value") +
  ylab("Density")
```