BDA_Project

2022-11-29

1.Introduction

Motivation

Except skin cancer, breast cancer is the most common cancer. Even though, deaths from breast cancer have declined over time, breast cancer remains the second leading cause of cancer death among women overall. Each year in the United States, about 264,000 cases of breast cancer are diagnosed in women and about 2,400 in men. Breast cancer death rates have been decreasing steadily since 1989, for an overall decline of 43% through 2020. The decrease in death rates is believed to be the result of finding breast cancer earlier through screening and increased awareness, as well as better treatments. However, the decline has slowed slightly in recent years. Therefore, early detection of breast cancer is very significant in the fight against it.

The problem

Medical screenings are usually performed by one professional. Therefore, the results could be exposed to bias and false assessment. Therefore, obtaining results from multiple professionals provides more reliable diagnosis. Even though, it could be more trustworthy, implementing such process could be impractical and expensive as there might be a deficit of professionals with necessary experience and skills. Therefore, a machine learning application that generates a diagnosis of breast cancer could be a competent solution.

Modeling idea

Bayesian Logistic Regression and Gaussian Naive Bayes Classifier are implemented in order to classify the fine-needle aspirate of breast mass as benign or malignant. In other words, this classification problem could be considered as binary classification problem. Two mentioned models are selected as desired models as they are have been proven to be efficient and simple to implement.

2. Description of the data and the analysis problem.

Data Set

This project conducts the Bayesian analysis on the `Breast Cancer Wisconsin` data set with 30 predictors and 568 observations, available at UCI machine learning repository. The predictors are various characteristics of a cell nucleus, measured from a digitized image of a fine needle aspirate of breast mass. There are three cell nuclei with ten real-valued features recorded, implying thirty features in total. The target is a categorical variable taking either "M", a shorthand for malignant and B a shorthand for benign. Here, we deal with an unbalanced data set where 37.5% of patients are diagnosed with malignant tumors and 62.85% of patients are diagnosed with benign tumors. For the sake of convenience, we treat the data set as balanced and do not perform additional processing on it.

To simplify the matters, we substitute the "M" elements of target vector with 0 and the "B" elements of target vector with 1. We proceed by renaming the column names of the data with their associated feature names of the data.

```
# loading the libraries
```

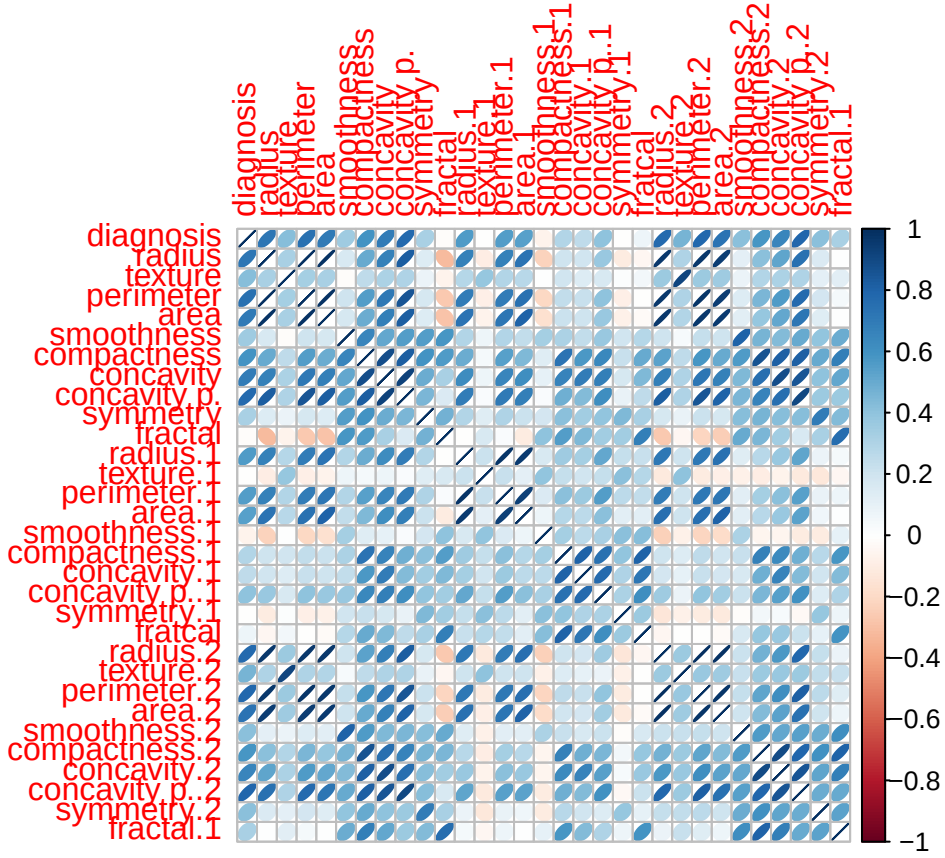
```
library(aaltobda)
library(rstan)
library(corrplot)
library(bayesplot)
library(loo)
library(MLmetrics)
library(ggplot2)
library(gridExtra)

# loading the data
BCW_data<-read.csv("wdbc.data")
seed_index<-1724

# renaming the column names to the feature names specified by "Data Set Information"

colnames(BCW_data)<-c("id", "diagnosis", "radius", "texture", "perimeter", "area",
                      "smoothness", "compactness","concavity", "concavity p.",
                      "symmetry", "fractal", "radius", "texture", "perimeter",
                      "area", "smoothness", "compactness", "concavity", "concavity p.",
                      "symmetry", "fratcal", "radius", "texture", "perimeter",
                      "area", "smoothness", "compactness","concavity",  "concavity p.",
                      "symmetry", "fractal" )

# renaming the labels associated with individuals from categorical to numerical

BCW_data[BCW_data == "M"] <- 1
BCW_data[BCW_data == "B"] <- 0
BCW_data$diagnosis<-as.numeric(BCW_data$diagnosis)
```

Feature Selection via Correlation Matrix

Let us consider the Breast Cancer Wisconsin Data Set with 30 predictors (features) and 568 observations. An issue may arise from redundant features, i.e., there might be several features with strongest effects on predictions whereas the remaining predictors may exhibit zero or marginal effects on predictions. Therefore, classification accuracy may be improved by conducting feature selection in advance. Many proposals have been made toward developing feature selection algorithms such as forward stage-wise regression , lasso , knock-off filters . We will not use the above-mentioned feature selection methods, as they are not within the scope of the bayesian data analysis course. We compute the pearson correlation matrix of the data to explore the correlation between features and the diagnosis and the pairwise correlation between features. We emphasize that there are more sophisticated correlation measures such as *Kendall-*τ and Spearman correlations, which are robust against heavy-tailed distributions.

We observe from the correlation matrix shown in Figure. 1 that features *fractal dimesnional* of cell nuclei 1, *texture* of cell nuclei 2, *symmetry* of cell nuclei 2 have negligible correlation with diagnosis. On the other hand, features *perimeter*, area and radius are highly correlated across all cell nucleons, and features *concavity*, *concavity points* and *compactness* across all cell nucleons. Hence, one may reduce the number of features by eliminating the redundant features having negligible correlation with diagnosis and selecting one feature within the group of highly correlated features. We choose the features *concavity* and *perimeter* from the correlated features.

```
# Plotting the correlation matrix

corrplot(cor(BCW_data[,2:32]), method='ellipse')
```




```
# Data after feature selection
BCW_data_reduced <- BCW_data[, -c(3,6,8,10,12,13,14,16,18,20,21,23,26,28,30) ]
```

Herein, the number of observations is much larger than that of features, implying that feature selection may not significantly boost the classification accuracy. However, we found out that the convergence of logistic regression improves upon removal of irrelevant features and hence we proceed with dimensionally reduced model.

3. Model Description

Bayesian Logistic Regression

We now present the Bayesian logistic regression (Bishop and Nasrabadi 2006), a linear method for classification. Herein, we denote by $\mathcal{D} = \{(y_i, \mathbf{x}_i)\}_{i=1}^n$, the Breast Cancer Data of $n = 568$ observation with binary labels $y_i \in \{0, 1\}$ and associated feature vectors $\mathbf{x}_i \in \mathbb{R}^p$. The logistic regression models the posterior probabilities of binary classes using a logistic sigmoid acting on a linear function of features such that

$$\mathbb{P}(y = 1|\mathbf{x}) = \frac{\exp(\alpha + \mathbf{x}^T\boldsymbol{\beta})}{1 + \exp(\alpha + \mathbf{x}^T\boldsymbol{\beta})} \text{ and } \mathbb{P}(y = 0|\mathbf{x}) = 1 - \mathbb{P}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(\alpha + \mathbf{x}^T\boldsymbol{\beta})}$$

The logistic regression assigns a new instance $\mathbf{x}$ to the class with the largest value for its discriminant function. Here, the discriminant function is the monotone logit transformation of $\mathbb{P}(\mathcal{C}_1|\mathbf{x})$ also known as *log-odds*,

$$\text{logit}(\mathbb{P}(y = 1|\mathbf{x})) = \log\big(\frac{\mathbb{P}(y = 1|\mathbf{x})}{1 - \mathbb{P}(y = 1|\mathbf{x})}\big) = \log\big(\frac{\mathbb{P}(y = 1|\mathbf{x})}{\mathbb{P}(y = 0|\mathbf{x})}\big) = \alpha + \mathbf{x}^T\boldsymbol{\beta}$$

To make a decision on a new instance $\mathbf{x}$, the binary outcome y takes either one or zero. Hence, one can interpret the outcomes y_i are Bernoulli -distributed given $\mathbf{x}_i$ as follows:

$$y_i|\mathbf{x}_i \sim \text{Bernoulli}(p_i)$$

where $p_i = \mathbb{P}(y_i = 1|\mathbf{x}) = \text{logit}^{-1}(\alpha + \mathbf{x}_i^T\beta)$. Consequently, one can write the likelihood function for the entire set of outcomes

$$\mathbb{P}(\mathbf{y}|\mathbf{X}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \tag{1}$$

$$\tag{2}$$

In the context of Bayesian logistic regression, we introduce priors for the intercept α and the coefficient vector β. In this project, we considered a weakly informative prior for the intercept and various priors for the coefficient vector such as Gaussian (Figueiredo 2001), Laplacian (Park and Casella 2008) and Horseshoe (Carvalho, Polson, and Scott 2010). We have additionally studied the Bayesian logistic regression compared to logistic regression with uniform priors. In what follows, we first introduce the priors chosen for this project and then proceed with simulation results.

Naive Bayes Classifier

Naive Bayes Classifier is defined as follows:

$C_{predict} = argmax(P(C_k|x_1, ..., x_n)) = argmax(\frac{P(C_k)P(x_1,...,x_n|C_k)}{P(x_1,...,x_n)})$

Where C_k is a class of k classes and $x = (x_1, ..., x_n)$ is an instance with n feature variables.

$P(x_1, ..., x_n)$ is a constant for an instance, so it has no effect on the argmax function. We can also use logarithmic scale for the argmax function since we are dealing with positive probabilities. Thus the function can be simplified as follows:

$C_{predict} = argmax(log(P(C_k)) + log(P(x_1, ..., x_n|C_k)))$

Naive Bayes classifier makes a strong assumption of mutual independence between the feature variables. Thus the function can be further simplified as:

$C_{predict} = argmax(log(P(C_k)) + \sum_{i=1}^{n} log(P(x_i|C_k)))$

4. Prior justification

Bayesian Logistic Regression

- First, we consider the logistic regression with a Gaussian prior having a zero mean and covariance $\sigma^2\mathbf{I}$. The zero-mean Gaussian prior implies it is very unlikely to observe large coefficients β_j. Indeed, a logistic regression with normally-distributed prior is closely related to the ridge logistic regression and unlike the ridge logistic regression σ does not require any hyper-parameter tuning.

$$\beta|\sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \tag{3}$$
$$\sigma \sim \mathcal{N}(100, 1000) \tag{4}$$

For the sake of simplicity, we refer to the logistic regression with Gaussian prior by "**ridge logistic regression**".

- We now consider the logistic regression with a Laplace prior having a scale parameter σ. The logistic regression with a Laplace prior can be interpreted as Bayesian analog of logistic regression penalized by ℓ_1-norm. In particular, the logistic regression with a Laplace prior shrinks the weakly related coefficients β_j parameters to zero without any need for tuning of hyper-parameters, as is the case with Lasso.

$$\mathbb{P}(\beta|\sigma) = \prod_{j=1}^{p} \frac{1}{2\sigma} \exp(-|\beta_j|/\sigma), \tag{5}$$

$$\sigma \sim \mathcal{N}(100, 1000) \tag{6}$$

For the sake of simplicity, we refer to the logistic regression with Laplace prior by "**lasso logistic regression**".

We then consider the logistic regression with the regularized horseshoe prior (Piironen and Vehtari 2017) for dealing with sparse classification and regression problems. The regularized horseshoe prior is free of user-chosen hyper-parameters plus the regularization of coefficients away from zero is allowed. Furthermore, the regularized horseshoe suggests a rigorous framework for selection of the global shrinkage hyperparameter, based on the effective number of non-zero coefficients.

$$\beta_j|\lambda_j, \tau, c \sim \mathcal{N}(0, \tau^2\tilde{\lambda}_j^2), \tag{7}$$

$$\tilde{\lambda}_j^2 = \frac{c^2\lambda_j^2}{c^2 + \tau^2\lambda_j^2}, \tag{8}$$

$$\lambda_j \sim C^+(0, 1), \tag{9}$$

$$c^2 \sim \text{Inv-Gamma}(\nu/2, \nu s^2/2), \tag{10}$$

$$\tau|\gamma \sim C^+(0, \gamma). \tag{11}$$

For the sake of simplicity, we refer to the logistic regression with regularized horseshoe prior by "**regularized hoseshoe logistic regression**".

Naive Bayes Classifier

For this dataset, we chose Gaussian Naive Bayes Classifier (GNBC) as a model. GNBC is a variant of Naive Bayes Classifier (Bishop and Nasrabadi 2006) where the likelihood distributions $P(x_i|C_k)$ are assumed to be Gaussian. The model type makes sense since all data in the dataset is numeric.

Furthermore, since the problem is a binary classification problem, the prior for the classifier can be modeled as a Bernoulli distribution. The parameter p for the bernoulli distribution was selected according to the distribution of labels in the raw data.

The Gaussian likelihood distributions need to be calculated with stan before using them for classification. For the stan calculations, we chose to normalize the feature variables to have a mean of 0 and use a weakly informative Gaussian prior of $N(0, 10)$ for each of the features. The full likelihood function was then be calculated from the data and the prior.

For classification, we then used the approximated Gaussian likelihood function combined with the prior Bernoulli distribution of labels.

5. Stan code

```
n<-nrow(BCW_data)
split_ratio<-0.8
idx_train <-sample(n,split_ratio*n)
```

```
BCW_train <-BCW_data[idx_train,2:32]
BCW_test <-BCW_data[-idx_train,2:32]

x_train<-BCW_train[,-1]
x_test<-BCW_test[,-1]

#data after feature selection
BCW_train_reduced <-BCW_data_reduced[idx_train,2:17]
BCW_test_reduced <-BCW_data_reduced[-idx_train,2:17]

x_train_reduced<-BCW_train_reduced[,-1]
x_test_reduced<-BCW_test_reduced[,-1]
```

Bayesian Logistic Regression

Ridge logistic regression

```
# Fitting the model using logistic regression with Gaussian prior

file_ridge <- file.path( "ridge_logistic_regression_model.stan")
data_ridge_reduced<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                          'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                          'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                          'y_test'=BCW_test_reduced[,1],'scale_alpha'=1000,
                          'mean_hypbeta'=100,'scale_hypbeta'=1000)

# increasing the target acceptance rate and decreasing the step size for better performance

fit_ridge <- stan(file=file_ridge,data=data_ridge_reduced, core=4,
                  control=list(adapt_delta=0.999, stepsize = 0.008),
                  seed=seed_index, iter=3000, verbose=FALSE)
```

```
## Trying to compile a simple C file

## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG      -fpic  -g -O2 -fdebug-prefix-map=/build/r-bas
## clang -flto=thin -shared -L/usr/lib/R/lib -Wl,-Bsymbolic-functions -Wl,-z,relro -o foo.so foo.o -L/us
## /usr/bin/ld: /usr/lib/llvm-10/bin/../lib/LLVMgold.so: error loading plugin: /usr/lib/llvm-10/bin/../l
## clang: error: linker command failed with exit code 1 (use -v to see invocation)
## make: *** [/usr/share/R/share/make/shlib.mk:10: foo.so] Error 1

## Warning: There were 2034 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: There were 3 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## https://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

Lasso logistic regression

```
# Fitting the model using logistic regression with Laplace prior

file_lasso <- file.path( "lasso_logistic_regression_model.stan")
data_lasso_reduced<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                          'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                          'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                          'y_test'=BCW_test_reduced[,1])

# increasing the target acceptance rate and decreasing the step size for better performance

fit_lasso <- stan(file=file_lasso,data=data_lasso_reduced, core=4,
                  control=list(adapt_delta=0.999, stepsize = 0.008),
                  seed=seed_index, iter=3000, verbose=FALSE)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
```

```
## Warning: There were 4863 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## https://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

Regularized horseshoe logistic regression

```
# Fitting the model using logistic regression with regularized horseshoe prior

file_rh <- file.path( "regularized_horseshoe_logistic_regression_model.stan")
data_rh_reduced<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                       'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                       'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                       'y_test'=BCW_test_reduced[,1],'scale_global'=100,
                       'nu_local'=1,'nu_global'=1,'slab_scale'=100,'slab_df'=20)

# increasing the target acceptance rate and decreasing the step size for better performance

fit_rh <- stan(file=file_rh,data=data_rh_reduced,core=5,
               control=list(adapt_delta=0.999,stepsize = 0.008),
               iter=3000, verbose=FALSE)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
```

```
## Warning: There were 15 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: There were 5984 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: The largest R-hat is NA, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

Naive Bayes Classifier

```
# renaming the labels associated with individuals from categorical to numerical

cancer_data = subset(BCW_data, select = -c(id))

# Randomize data order
rows <- sample(nrow(cancer_data))
cancer_data <- cancer_data[rows, ]

# Split into training and testing data sets
training_size = round(0.8 * nrow(cancer_data),0)
training_data = head(cancer_data, training_size)
test_data = tail(cancer_data, nrow(cancer_data) - training_size)

# Standardize data in order to use standardized normal prior
training_data$diagnosis<-as.character(training_data$diagnosis)
ind <- sapply(training_data, is.numeric)
training_data[ind] <- lapply(training_data[ind], scale)
training_data$diagnosis<-as.numeric(training_data$diagnosis)

test_data$diagnosis<-as.character(test_data$diagnosis)
ind <- sapply(test_data, is.numeric)
test_data[ind] <- lapply(test_data[ind], scale)
test_data$diagnosis<-as.numeric(test_data$diagnosis)

# Split training data into malignant and benign data sets
malignant_training_samples = subset(training_data, diagnosis == 1)
malignant_training_samples = subset(malignant_training_samples, select = -c(diagnosis))
benign_training_samples = subset(training_data, diagnosis == 0)
benign_training_samples = subset(benign_training_samples, select = -c(diagnosis))

# Extract labels from test data
test_labels = test_data[['diagnosis']]
test_samples = subset(test_data, select = -c(diagnosis))

# Prepare the data for stan model
N_benign = nrow(benign_training_samples)
N_malignant = nrow(malignant_training_samples)
prior_p = N_malignant / (N_benign + N_malignant)

naivebayes_model_data <- list(
  N_benign=N_benign,
  N_malignant=N_malignant,
  N_test=nrow(test_samples),
  J=ncol(test_samples),
  samples_benign=benign_training_samples,
  samples_malignant=malignant_training_samples,
  samples_test=test_samples,
  y_test=test_labels,
```

```
  prior_p=prior_p
)

# Load and fit the stan model
naivebayes_file <- file.path("naivebayes.stan")
naivebayes_fit <- stan(file=naivebayes_file,data=naivebayes_model_data, core=4,
                  control=list(adapt_delta=0.999, stepsize = 0.008),
                  seed=seed_index, iter=3000, verbose=FALSE)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
```

6. Explanation on how the Stan code was run

Every model were run with 4 chains and 3000 iterations, 1500 of which are warm-up iterations and 1500 are sampling iterations.

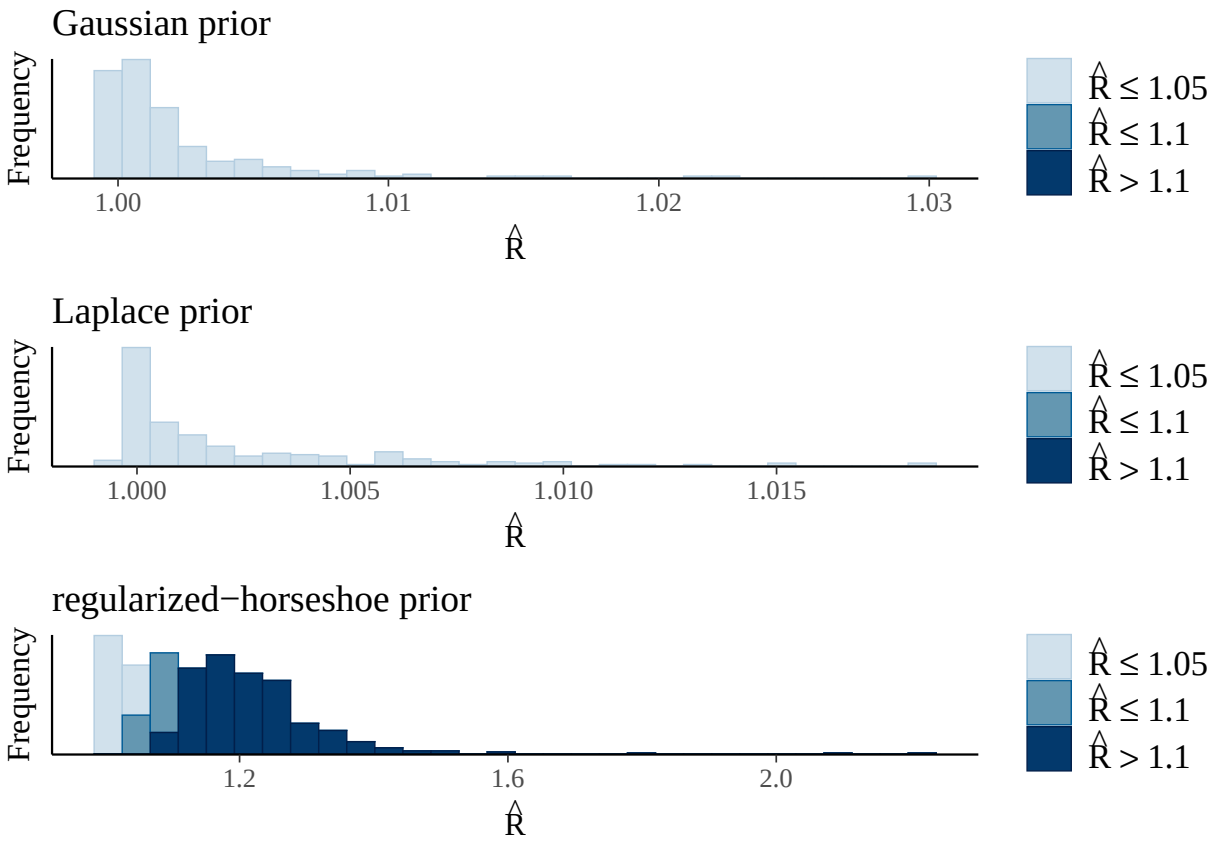
7. Convergence Diagnostics

Bayesian Logistic Regression

We now discuss the convergence issues of HMC and report different metrics of convergence. Firstly, we analyze the R-hat diagnostics that compares between and within-chains means and variances. Generally, estimates are fully reliable if R-hat values below 1.01 whereas estimates associated with R-hat values above 1.1 are not reliable. The R-package `bayesplot` adopts a slightly more relaxed convention by considering estimates associated with R-hat values above 1.05 as fully reliable. It can be observed from the following figure, the the logistic regression estimates using Laplace prior are fully reliable. Certain number of estimates using regularized horseshoe and Gaussian priors are not reliable as their corresponding R-hat values are above 1.1.

```
# computing r-hat values for the three models
rhats_ridge<-rhat(fit_ridge)
rhats_lasso<-rhat(fit_lasso)
```

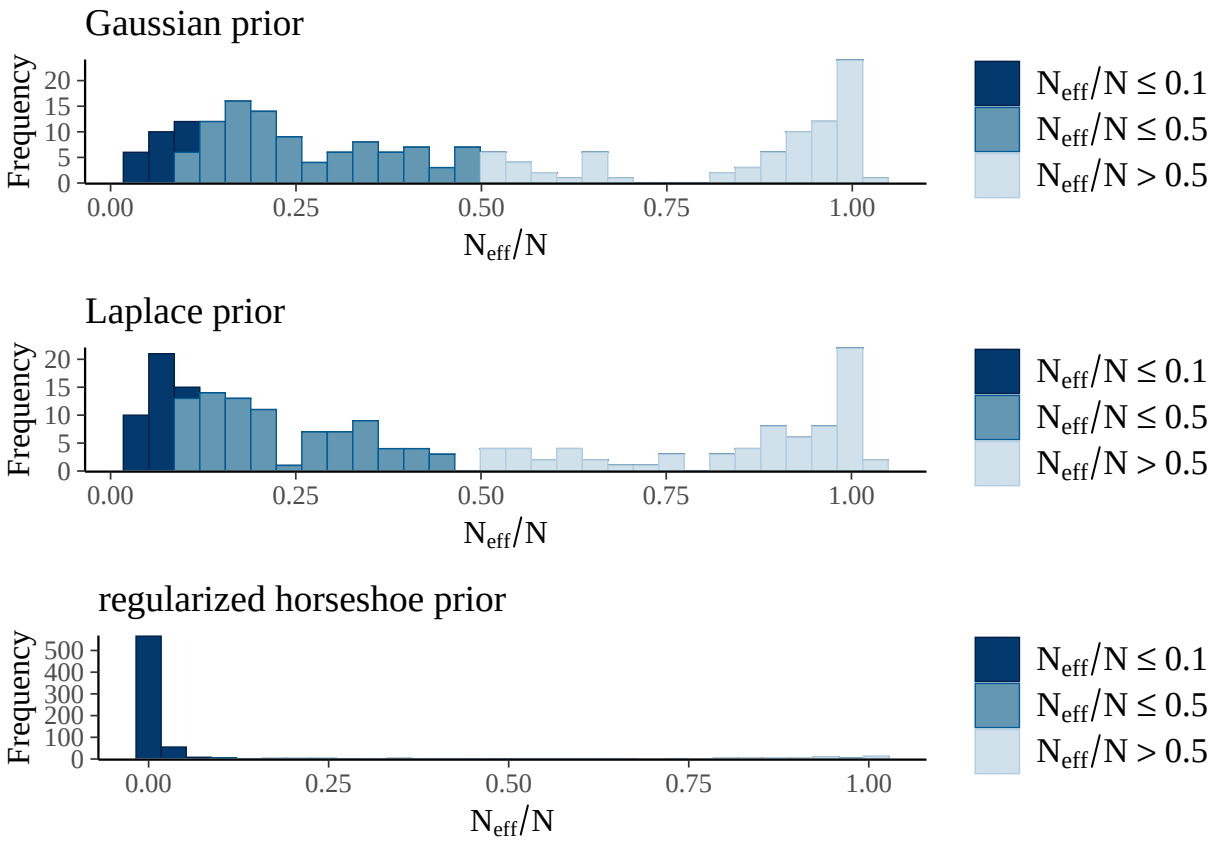
```
rhats_rh<-rhat(fit_rh)


# plotting the r-hat histogram

p1<-mcmc_rhat_hist(rhats_ridge)+ ggtitle("Gaussian prior")+
  ylab("Frequency")
p2<-mcmc_rhat_hist(rhats_lasso)+ ggtitle("Laplace prior")+
  ylab("Frequency")
p3<-mcmc_rhat_hist(rhats_rh)+ ggtitle("regularized-horseshoe prior")+
  ylab("Frequency")
grid.arrange(p1, p2, p3,nrow = 3)
```



We then turn our attention to the effective sample size, which measures the how many independent random samples provide the same accuracy as the number of dependent samples via MCMC. The following figures show that the bulk of estimates obtained by logistic regression using regularized horseshoe have very low effective sample size, that is not desired. As expected, the logistic regression using Laplace prior shows a better performance in term of effective sample size.

```
# computing the ratios of effective sample size to total sample size
ratios_ridge <- neff_ratio(fit_ridge)
ratios_lasso <- neff_ratio(fit_lasso)
ratios_rh <- neff_ratio(fit_rh)

# plotting the ratios of effective sample size to total sample size histogram
p1<-mcmc_neff_hist(ratios_ridge, size = 2)+ ggtitle("Gaussian prior")+
```

```
  ylab("Frequency")
p2<-mcmc_neff_hist(ratios_lasso, size = 2)+ ggtitle("Laplace prior")+
  ylab("Frequency")
p3<-mcmc_neff_hist(ratios_rh, size = 2)+ ggtitle("regularized horseshoe prior")+
  ylab("Frequency")

grid.arrange(p1, p2, p3, nrow = 3)
```



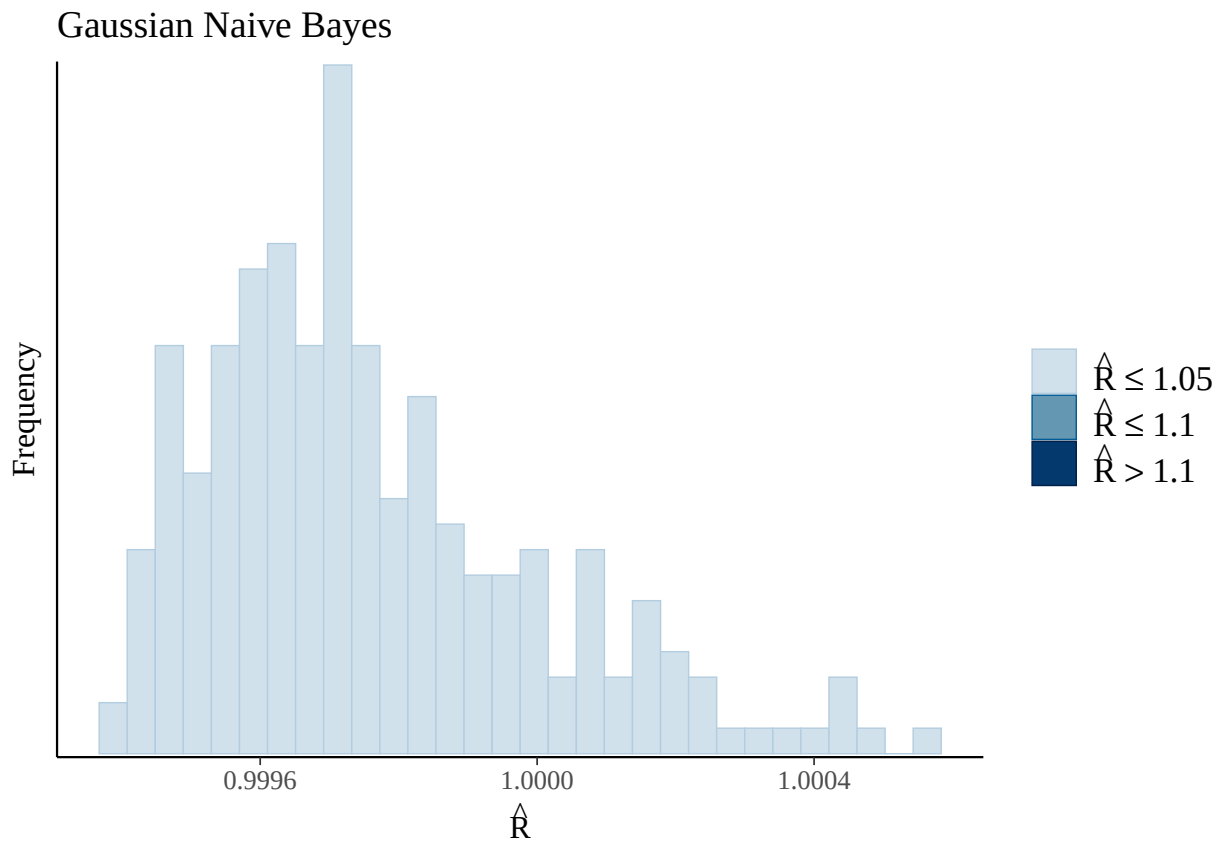
In addition to the above issues, we notice from the warning messages while fitting the regularized horseshoe logistic regression that there are divergent transitions and maximum treedepth is hit. Regarding the divergent transitions, we could mitigate the issue substantially by increasing the target acceptance rate to 0.999 and reducing the step size to 0.008. However, there were still few divergent transitions and these issues often occur due to highly varying nature of posterior function. Furthermore, we have observed some R-hat values are "NA", which we believe it is not an important as discussed by the lecturer of the course on `Github`. This phenomenon could be observed for a discrete quantity where one state has high probability and chain does not visit the other state often, which matches with our problem where labels are discrete and data is unbalanced.

Naive Bayes Classifier

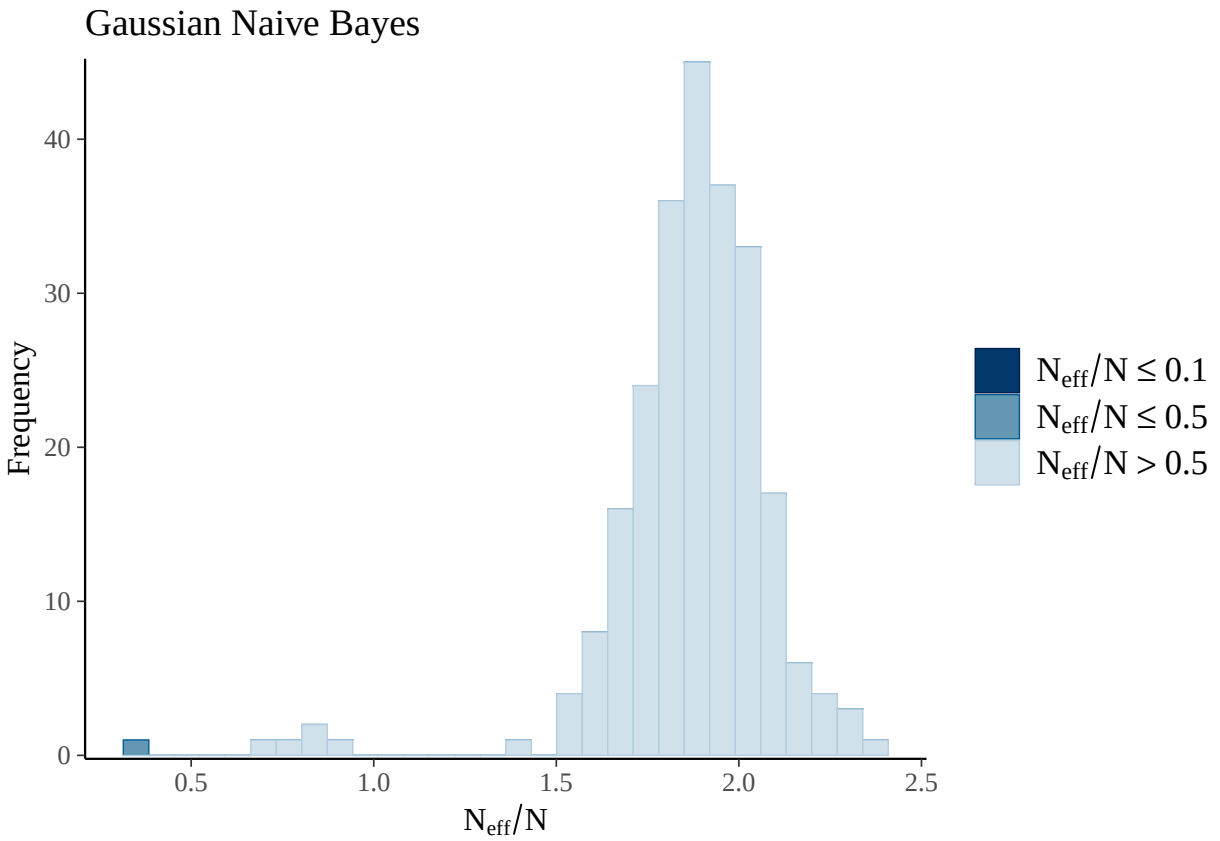
As can be observed from the R-hat histogram, all R-hat values are below 1.01, implying that all parameter estimates via naive bayes classifier are fully reliable whereas there were some R-hat values of logistic regression classifier were larger than 1.1, meaning that they are not reliable estimates.

```
# computing r-hat values for the three models
rhats_naivebayes<-rhat(naivebayes_fit)

# plotting the r-hat histogram
```

```
p1<-mcmc_rhat_hist(rhats_naivebayes)+ ggtitle("Gaussian Naive Bayes")+ylab("Frequency")
grid.arrange(p1)
```




```
# computing the ratios of effective sample size to total sample size
ratios_naivebayes <- neff_ratio(naivebayes_fit)

# plotting the ratios of effective sample size to total sample size histogram
p1<-mcmc_neff_hist(ratios_naivebayes, size = 2)+ ggtitle("Gaussian Naive Bayes")+
  ylab("Frequency")

grid.arrange(p1)
```



From the figure above, we can see that the bulk of estimates obtained by the Naive Bayes Classifier has very high effective sample size.

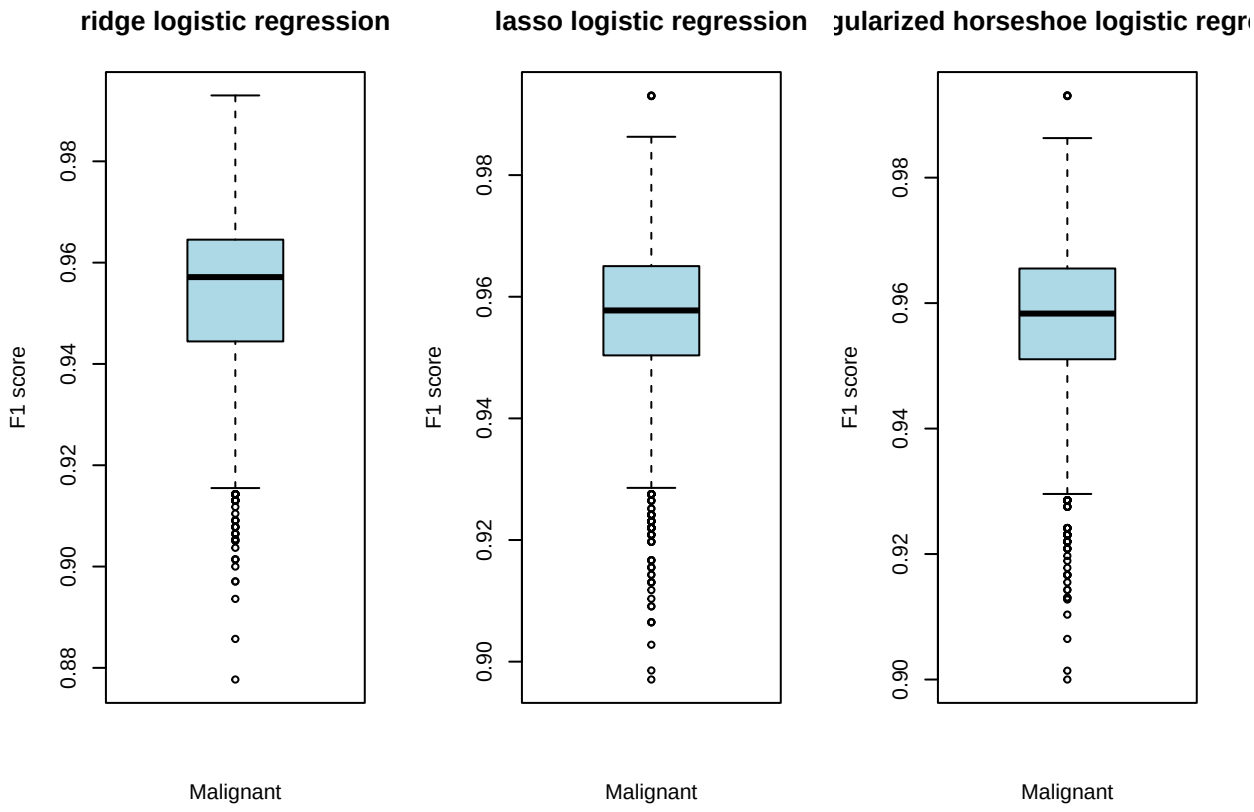
#8 Posterior Predictive Performance

Bayesian Logistic Regression

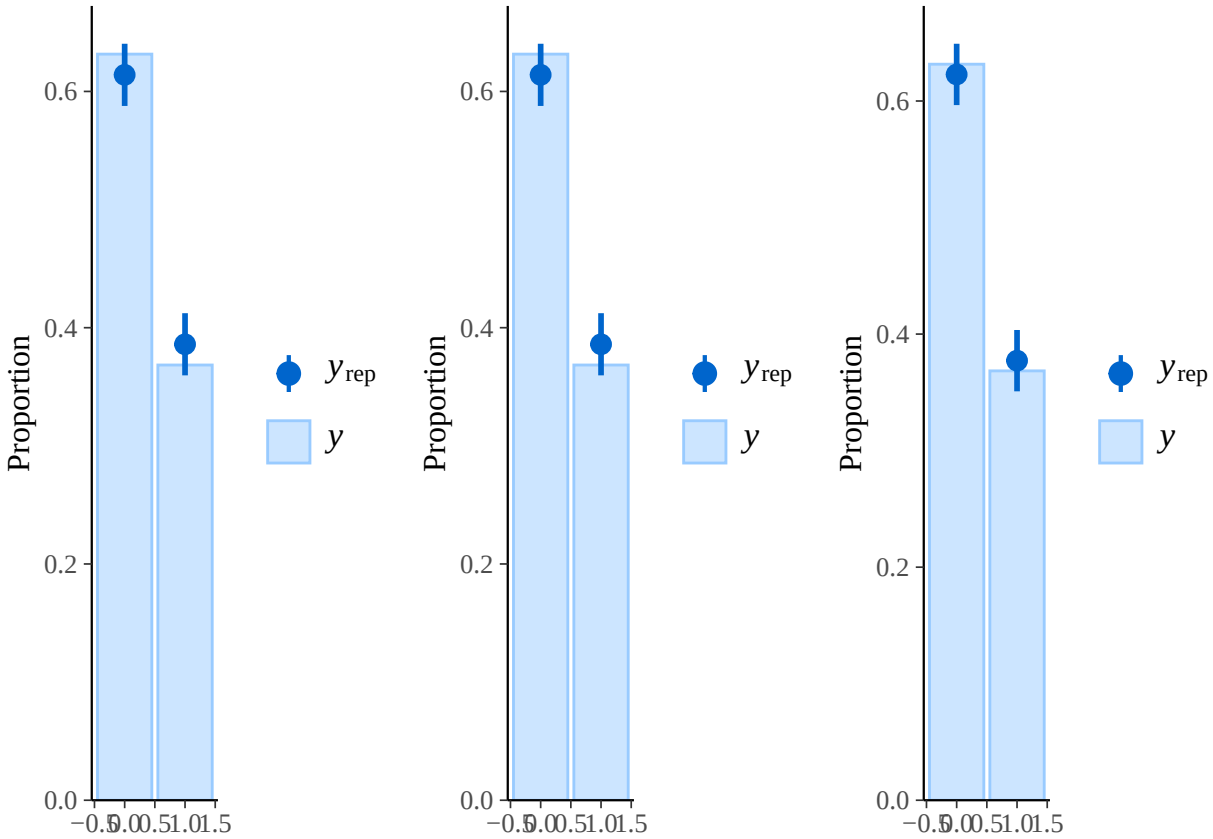
Herein, we will demonstrate the posterior predictive performance of the Bayesian logistic regression for Gaussian prior, Laplace prior and regularized horseshoe prior. We compare the effect of priors on posterior predictive performance using the celebrated F1-score (Rijsbergen 1979) and credible intervals. More specifically, we simultaneously illustrate the bar plot of the true labels and plot the credible intervals of posterior predictions for each label. As can be seen from the following figures, the logistic exhibit reliable performance in classification of cancer patients where the bulk of predictions are close to true labels and credible intervals are very narrow, implying reliable prediction of labels. Hence, it would be difficult to decide which model performs better, leading us to choose an alternative method for selection of the best model.

```
# Extracting posterior predicted labels

y_pred_ridge<-as.matrix(fit_ridge,pars="y_pred")
y_pred_lasso<-as.matrix(fit_lasso,pars="y_pred")
y_pred_rh<-as.matrix(fit_rh,pars="y_pred")

# computing F1-score for measuring the posterior predictive accuracy

f1_ridge<-rep(0,nrow(y_pred_ridge))
for (i in 1:nrow(y_pred_ridge)){
  f1_ridge[i]=F1_Score(y_pred_ridge[i,],BCW_test[,1])
}
```

```
f1_lasso<-rep(0,nrow(y_pred_lasso))
for (i in 1:nrow(y_pred_lasso)){
  f1_lasso[i]=F1_Score(y_pred_lasso[i,],BCW_test[,1])
}

f1_rh<-rep(0,nrow(y_pred_rh))
for (i in 1:nrow(y_pred_rh)){
  f1_rh[i]=F1_Score(y_pred_rh[i,],BCW_test[,1])
}

# box-plots of f1-scores

par( mfrow= c(1,3) )
boxplot(f1_ridge,xlab="Malignant",ylab="F1 score",col="lightblue",main="ridge logistic regression")
boxplot(f1_lasso,xlab="Malignant",ylab="F1 score",col="lightblue",main="lasso logistic regression")
boxplot(f1_rh,xlab="Malignant",ylab="F1 score",col="lightblue",main="regularized horseshoe logistic regr
```

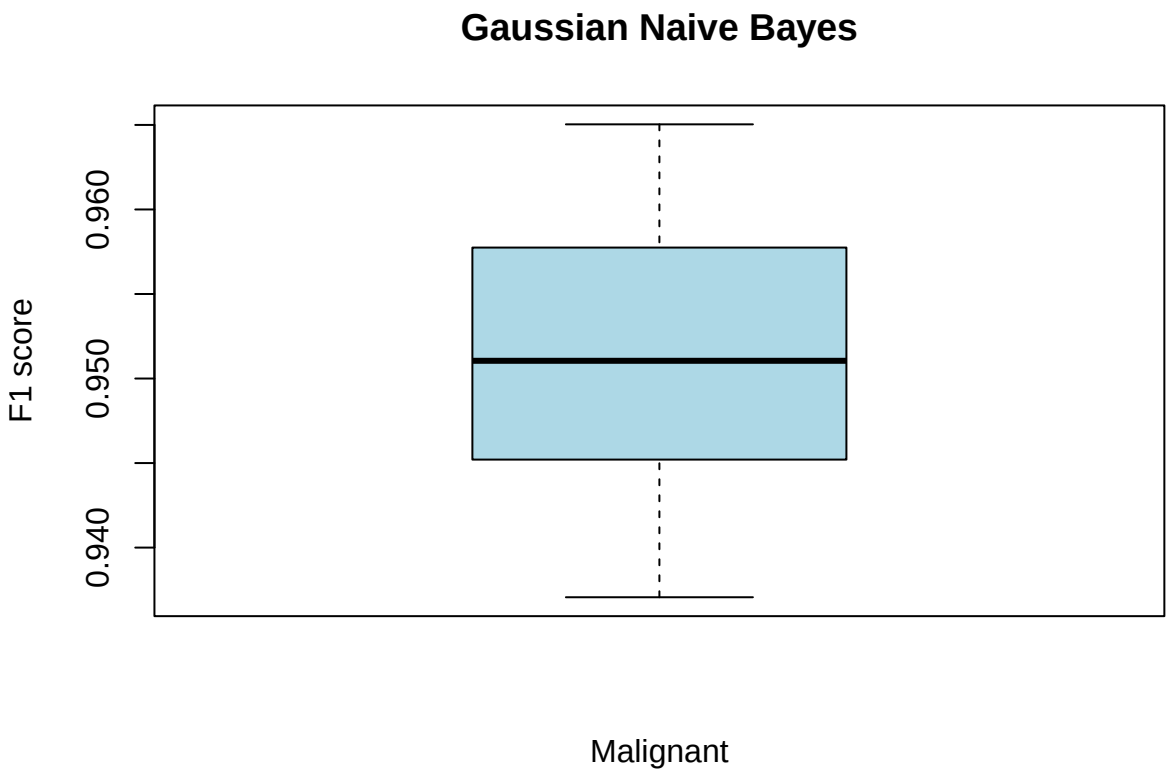



```
# barplot of malignant and benign cases of the test plus credible intervals

color_scheme_set("brightblue")
p1<-ppc_bars(y = BCW_test[,1],yrep = y_pred_ridge, freq=FALSE)
p2<-ppc_bars(y = BCW_test[,1],yrep = y_pred_lasso, freq=FALSE)
p3<-ppc_bars(y = BCW_test[,1],yrep = y_pred_rh, freq=FALSE)
grid.arrange(p1, p2, p3, nrow = 1)
```

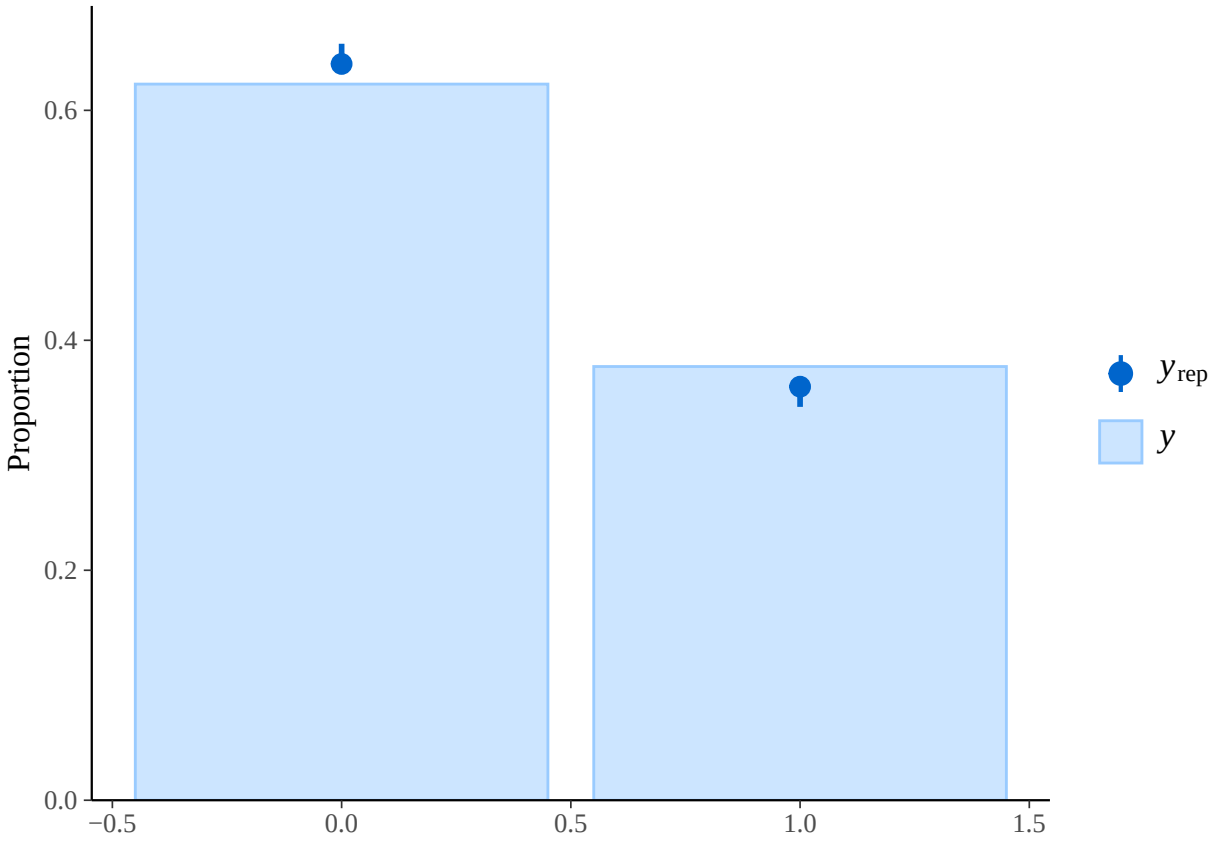


Naive Bayes Classifier

```
# Extracting posterior predicted labels

y_pred_naivebayes<-as.matrix(naivebayes_fit,pars="y_pred")

# computing F1-score for measuring the posterior predictive accuracy

f1_naivebayes<-rep(0,nrow(y_pred_naivebayes))
for (i in 1:nrow(y_pred_naivebayes)){
  f1_naivebayes[i]=F1_Score(y_pred_naivebayes[i,],test_labels)
}

# box-plots of f1-scores

boxplot(f1_naivebayes,xlab="Malignant",ylab="F1 score",col="lightblue",main="Gaussian Naive Bayes")
```




```
color_scheme_set("brightblue")
p1<-ppc_bars(y = test_labels,yrep = y_pred_naivebayes, freq=FALSE)
grid.arrange(p1)
```



As can be seen from the above figures, the Naive Bayes Classifier exhibits reliable performance in classification of cancer patients where the bulk of predictions are close to true labels and credible intervals are very narrow, implying reliable prediction of labels.

9. Sensitivity analysis of priors

Bayesian Logistic Regression

In this section, we examine how the choice of prior could effect the estimated parameters via Bayesian logistic regression. There is a common practice in Bayesian statistics to choose non-informative or weakly informative. In this sense, one prefers to regularize the posterior distribution with minimal expert knowledge and hence often seeks weakly informative for Bayesian analysis. Taking into account computational constraints, we will conduct a sensitivity analysis of priors only with the following priors:

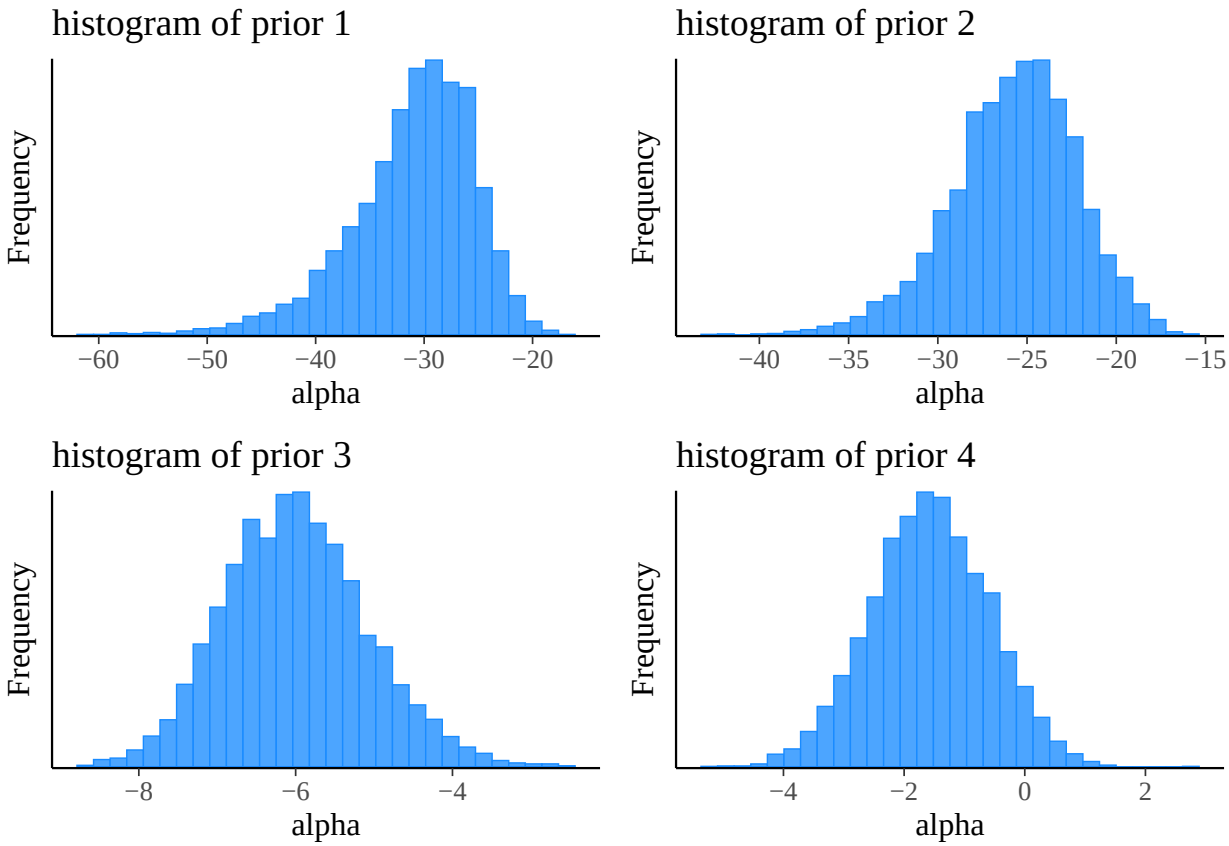
- Prior 1: $\alpha \sim \mathcal{N}(0, 1000)$, $\beta|\sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\sigma \sim \mathcal{N}(100, 1000)$
- Prior 2: $\alpha \sim \mathcal{N}(0, 10)$, $\beta|\sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\sigma \sim \mathcal{N}(0, 10)$
- Prior 3: $\alpha \sim \mathcal{N}(0, 1)$, $\beta|\sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\sigma \sim \mathcal{N}(0, 1)$
- Prior 4: $\alpha \sim \mathcal{N}(0, 1)$, $\beta|\sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\sigma \sim \mathcal{N}(100, 1000)$

```
# Fitting the logistic regression with Gaussian prior under Prior 2

data_ridge_reduced_2<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                            'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                            'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                            'y_test'=BCW_test_reduced[,1],'scale_alpha'=10,
                            'mean_hypbeta'=0,'scale_hypbeta'=10)
```

```
fit_ridge_2 <- stan(file=file_ridge,data=data_ridge_reduced_2, core=4,
                    control=list(adapt_delta=0.999, stepsize = 0.008),
                    seed=seed_index, iter=3000, verbose=FALSE)

# Fitting the logistic regression with Gaussian prior under Prior 3

data_ridge_reduced_3<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                            'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                            'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                            'y_test'=BCW_test_reduced[,1],'scale_alpha'=1,
                            'mean_hypbeta'=0,'scale_hypbeta'=1)

fit_ridge_3 <- stan(file=file_ridge,data=data_ridge_reduced_3,core=4,
                    control=list(adapt_delta=0.999,stepsize = 0.008),
                    seed=seed_index,iter=3000,verbose=FALSE)

# Fitting the logistic regression with Gaussian prior under Prior 4

data_ridge_reduced_4<- list('N_train'=nrow(x_train_reduced),'P'=ncol(x_train_reduced),
                            'X_train'=x_train_reduced,'y_train'=BCW_train_reduced[,1],
                            'N_test'=nrow(x_test_reduced),'X_test'=x_test_reduced,
                            'y_test'=BCW_test_reduced[,1],'scale_alpha'=1,
                            'mean_hypbeta'=100,'scale_hypbeta'=1000)

fit_ridge_4 <- stan(file=file_ridge,data=data_ridge_reduced_4, core=4,
                    control=list(adapt_delta=0.999, stepsize = 0.008),
                    seed=seed_index, iter=3000, verbose=FALSE)
```

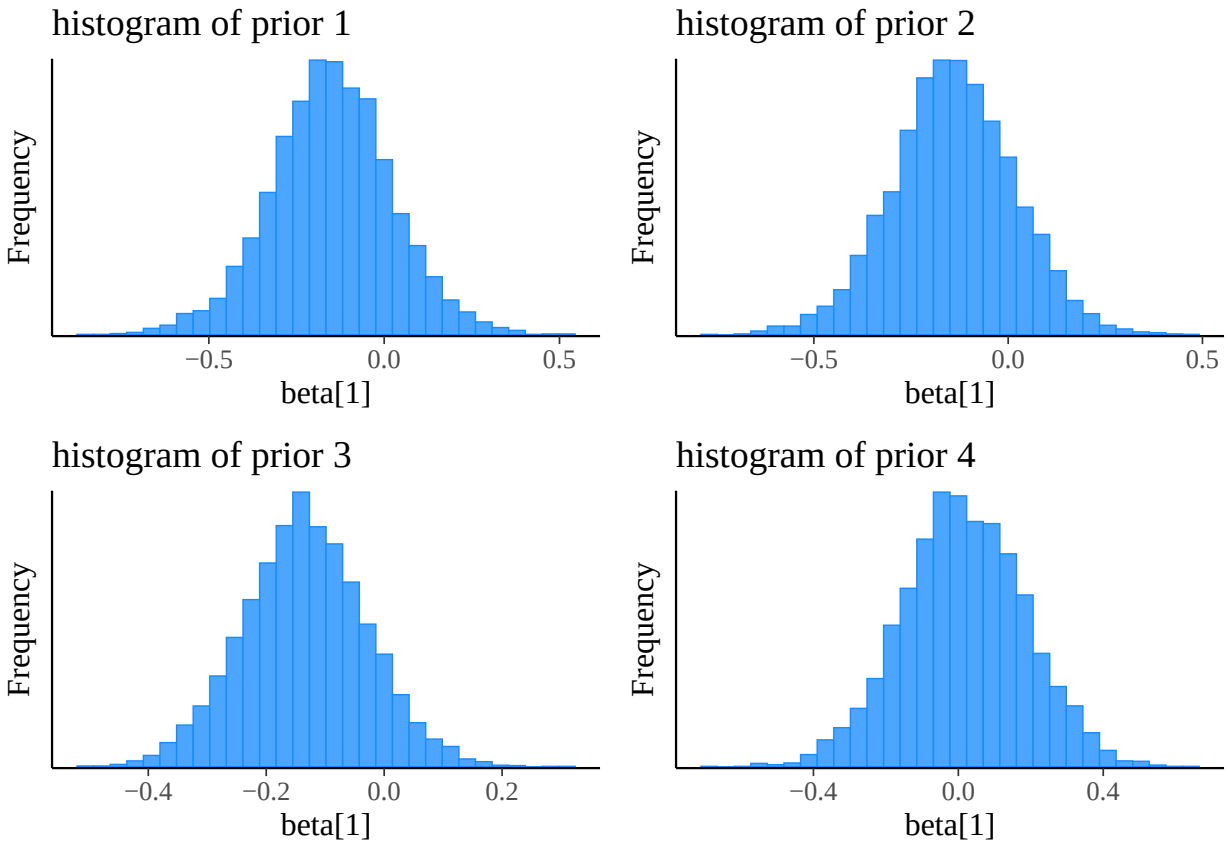
As it can be seen from the histogram plots of the following figures, the histogram of β_1 and β_2 estimates are not significantly affected by the choice of priors and the mass is concentrated around the same values. However, the histograms of α estimates are inflated with choice of priors more than those of β_i estimates.

```
# Histograms of \alpha estimates

color_scheme_set("brightblue")
p1<-mcmc_hist(fit_ridge, pars = c("alpha"))+
  ggtitle("histogram of prior 1")+
  ylab("Frequency")
p2<-mcmc_hist(fit_ridge_2, pars = c("alpha"))+
  ggtitle("histogram of prior 2")+
  ylab("Frequency")
p3<-mcmc_hist(fit_ridge_3, pars = c("alpha"))+
  ggtitle("histogram of prior 3")+
  ylab("Frequency")
p4<-mcmc_hist(fit_ridge_4, pars = c("alpha"))+
  ggtitle("histogram of prior 4")+
  ylab("Frequency")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



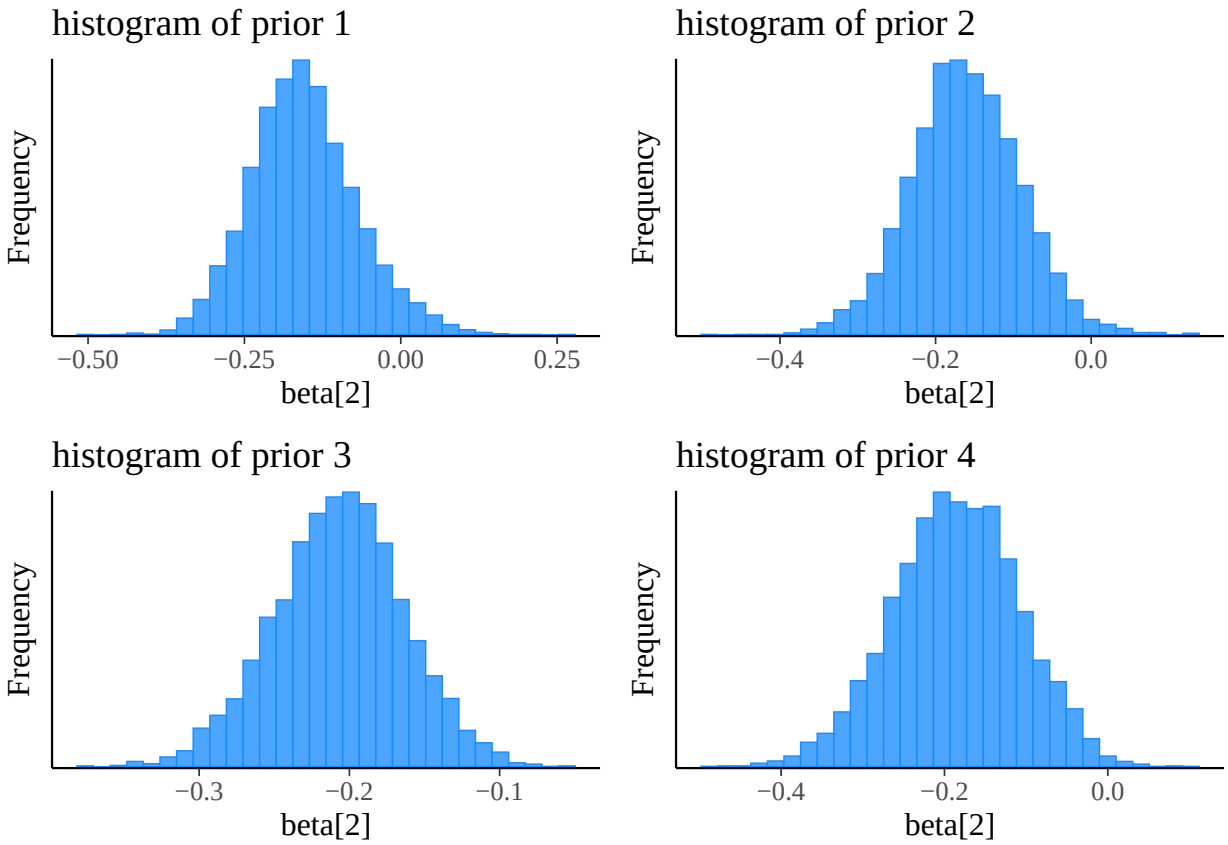

```
# Histograms of \beta_1 estimates

p1<-mcmc_hist(fit_ridge, pars = c("beta[1]"))+
  ggtitle("histogram of prior 1")+
  ylab("Frequency")
p2<-mcmc_hist(fit_ridge_2, pars = c("beta[1]"))+
  ggtitle("histogram of prior 2")+
  ylab("Frequency")
p3<-mcmc_hist(fit_ridge_3, pars = c("beta[1]"))+
  ggtitle("histogram of prior 3")+
ylab("Frequency")
p4<-mcmc_hist(fit_ridge_4, pars = c("beta[1]"))+
  ggtitle("histogram of prior 4")+
  ylab("Frequency")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Histograms of \beta_2 estimates

p1<-mcmc_hist(fit_ridge, pars = c("beta[2]"))+
  ggtitle("histogram of prior 1")+
  ylab("Frequency")
p2<-mcmc_hist(fit_ridge_2, pars = c("beta[2]"))+
  ggtitle("histogram of prior 2")+
  ylab("Frequency")
p3<-mcmc_hist(fit_ridge_3, pars = c("beta[2]"))+
  ggtitle("histogram of prior 3")+
  ylab("Frequency")
p4<-mcmc_hist(fit_ridge_4, pars = c("beta[2]"))+
  ggtitle("histogram of prior 4")+
  ylab("Frequency")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



10. Model comparison

Bayesian Logistic Regression

We now aim to select the model with best predictive performance using the method of leave-one-out cross validation (LOO-CV) (Vehtari, Gelman, and Gabry 2017). Herein, we use the expected-log-predictive-density-score, elpd_{loo} as defined in . We observe that the largest elpd_{loo} score and the lowest number of $\hat{k}$-values are associated with the regularized horseshoe logistic shoe logistic regression. This is in contradiction with earlier results, as there are bad $\hat{k}$ in all three models, the PSIS-LOO estimate might be not a reliable metric.

```
# Computing PSIS-LOO convergence diagnostics

psisloo_ridge<-loo(-extract(fit_ridge,pars='log_lik')$log_lik)
psisloo_lasso<-loo(-extract(fit_lasso,pars='log_lik')$log_lik)
psisloo_rh<-loo(-extract(fit_rh,pars='log_lik')$log_lik)


psisloo_ridge
```

```
##
## Computed from 6000 by 114 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -16.5  6.7
## p_loo         3.7  1.9
## looic        33.0 13.3
```

```
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     104   91.2%   1073
##  (0.5, 0.7]   (ok)         7    6.1%   3115
##    (0.7, 1]   (bad)        3    2.6%   6000
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

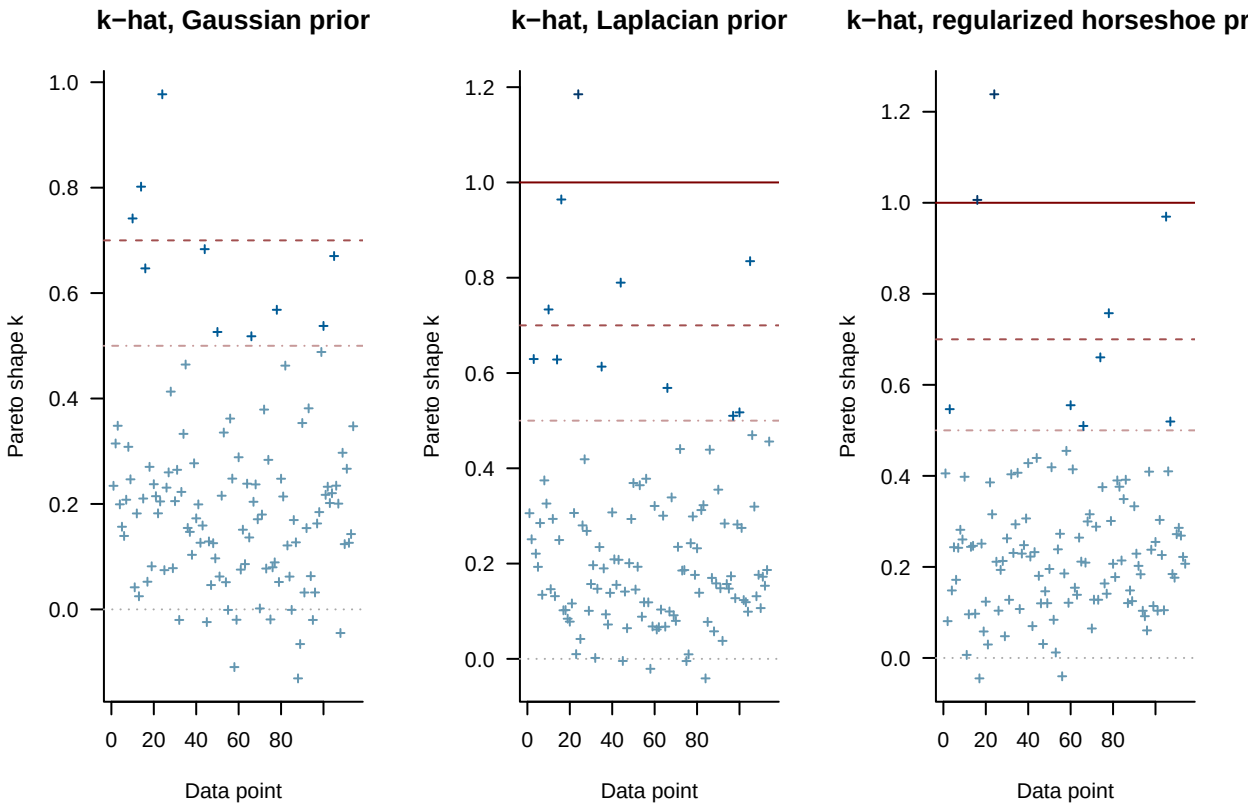
```
psisloo_lasso
```

```
##
## Computed from 6000 by 114 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -15.7  6.4
## p_loo         4.5  2.3
## looic        31.3 12.8
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     103   90.4%   959
##  (0.5, 0.7]   (ok)         6    5.3%   6000
##    (0.7, 1]   (bad)        4    3.5%   6000
##    (1, Inf)   (very bad)   1    0.9%   6000
## See help('pareto-k-diagnostic') for details.
```

```
psisloo_rh
```

```
##
## Computed from 6000 by 114 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -13.6  5.1
## p_loo         3.9  1.9
## looic        27.2 10.3
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     105   92.1%   1384
##  (0.5, 0.7]   (ok)         5    4.4%   6000
##    (0.7, 1]   (bad)        2    1.8%   5068
##    (1, Inf)   (very bad)   2    1.8%   6000
## See help('pareto-k-diagnostic') for details.
```

```
# scatter-plot of k-hat values
par( mfrow= c(1,3))
plot(psisloo_ridge, main="k-hat, Gaussian prior")
plot(psisloo_lasso, main="k-hat, Laplacian prior")
plot(psisloo_rh, main="k-hat, regularized horseshoe prior")
```



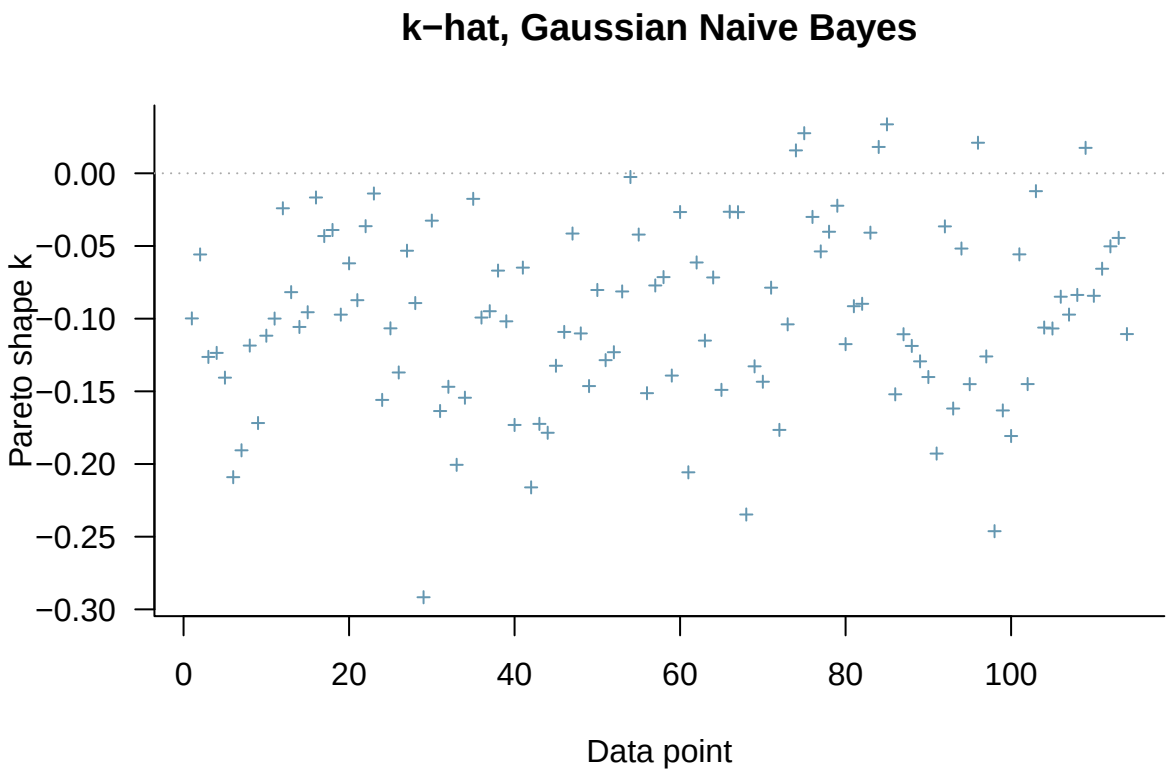
Naive Bayes Classifier

As it can be observed, all k-hat values of naive bayes calssifier are below 0.7, hence the PSIS-LOO estimate is reliable.

```
# Computing PSIS-LOO convergence diagnostics

psisloo_naivebayes<-loo(-extract(naivebayes_fit,pars='log_lik')$log_lik)

psisloo_naivebayes
```

```
##
## Computed from 6000 by 114 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -35.8 1.1
## p_loo         0.0 0.0
## looic        71.6 2.2
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
# scatter-plot of k-hat values

plot(psisloo_naivebayes, main="k-hat, Gaussian Naive Bayes")
```



11. Discussion of issues and potential improvements

Regarding the issues faced during the implementation process, we had difficulties with combining separately generated R and Stan files together. As this project is compiled by three students, it was difficult to combine all the necessary files to one report. In addition, there were errors that took time to debug. This could be improved by utilizing a common version control from the start (e.g. GitHub). Moreover, there were an issue with using different Stan libraries that are "rstan" and "cmdstanr". This caused putting in extra effort to convert all the files to one single report. This could be avoided by defining used libraries and versions better at the start.

12. Conclusion what was learned from the data analysis

During implementation process of the project, we applied what we have learned during the course into practicality. In addition, we learned how to work with certain R package libraries for visualizations such as "bayesplot", "ggplot2". We also learned how to solve the divergent transitions when the posterior curvature is highly varying and we need to increase the target acceptance rate and step size. Moreover, we gained experience in working as a group on such in-depth project and have understood the importance of leaving enough time for combining individual members' work and importance of clearly defining the libraries and other factors beforehand.

13. Self-reflection of what the group learned while making the project

We gained experience in working as a group on such in-depth project and have understood the importance of leaving enough time for combining individual members' work and importance of clearly defining the libraries and other factors beforehand. In addition, the list of bullet points available in the course repository was very helpful as it suggested us what sections are relevant or not to the project. Without this list, we would have spent significantly more time on writing the report.

14. References

Bishop, Christopher M, and Nasser M Nasrabadi. 2006. *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer.

Carvalho, Carlos M, Nicholas G Polson, and James G Scott. 2010. "The Horseshoe Estimator for Sparse Signals." *Biometrika* 97 (2): 465–80.

Figueiredo, Mário. 2001. "Adaptive Sparseness Using Jeffreys Prior." *Advances in Neural Information Processing Systems* 14.

Park, Trevor, and George Casella. 2008. "The Bayesian Lasso." *Journal of the American Statistical Association* 103 (482): 681–86.

Piironen, Juho, and Aki Vehtari. 2017. "Sparsity Information and Regularization in the Horseshoe and Other Shrinkage Priors." *Electronic Journal of Statistics* 11 (2): 5018–51.

Rijsbergen, CV. 1979. "Information Retrieval 2nd Ed Buttersworth." *London [Google Scholar]*.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2017. "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC." *Statistics and Computing* 27 (5): 1413–32.

15. Appendix

- The stan code for the ridge logistic regression

```
writeLines(readLines(file_ridge))
```

```
## // Ridge logistic regression
## data {
##   int<lower=0> N_train;
##   int<lower=0> P;
##
##   matrix[N_train,P] X_train;
##   int<lower=0,upper=1> y_train[N_train];
##
##   int<lower=0> N_test;
##
##   matrix[N_test,P] X_test;
##   int<lower=0,upper=1> y_test[N_test];
##
##   real<lower=0> scale_alpha;
##
##   real<lower=0> mean_hypbeta;
##   real<lower=0> scale_hypbeta;
## }
##
## parameters {
##
##   real<lower=0> sigma;
##   real alpha;     // intercept
```

```
##   vector[P] beta;
##
## }
##
## model {
##
##  alpha ~ normal(0,scale_alpha);
##
##   for (j in 1:P){
##
##      beta[j] ~ normal(0,sigma) ;
##
##   }
##
##   sigma ~ normal(mean_hypbeta,scale_hypbeta);
##   y_train ~ bernoulli_logit(alpha+X_train*beta);
##
## }
##
## generated quantities {
##
##
##  int<lower=0,upper=1> y_pred[N_test];
##  vector[N_test] log_lik;
##
##  for (i in 1:N_test){
##
##    log_lik[i] = -bernoulli_logit_lpmf(y_test[i] | alpha+X_test[i,]*beta);
##    y_pred[i] = bernoulli_logit_rng(alpha+X_test[i,]*beta);
##
##
##  }
##
##
##
##
## }
```

- The stan code for the lasso logistic regression

```
writeLines(readLines(file_lasso))
```

```
## // Lasso logistic regression
## data {
##   int<lower=0> N_train;
##   int<lower=0> P;
##
##   matrix[N_train,P] X_train;
##   int<lower=0,upper=1> y_train[N_train];
##
##   int<lower=0> N_test;
##
##   matrix[N_test,P] X_test;
##   int<lower=0,upper=1> y_test[N_test];
## }
```

```
##
##
## parameters {
##
##   real<lower=0> sigma;
##   real alpha;     // intercept
##   vector[P] beta;
##
## }
##
## model {
##
##  alpha ~ normal(0,100);
##
##   for (j in 1:P){
##
##      beta[j] ~ double_exponential(0,sigma) ;
##
##   }
##
##   sigma ~ normal(100,1000);
##   y_train ~ bernoulli_logit(alpha+X_train*beta);
##
## }
##
## generated quantities {
##
##
##  int<lower=0,upper=1> y_pred[N_test];
##  vector[N_test] log_lik;
##
##  for (i in 1:N_test){
##
##    log_lik[i] = -bernoulli_logit_lpmf(y_test[i] | alpha+X_test[i,]*beta);
##    y_pred[i] = bernoulli_logit_rng(alpha+X_test[i,]*beta);
##
##  }
##
##
## }
```

- The stan code for the regularized horseshoe logistic regression

```
writeLines(readLines(file_rh))
```

```
## // regularized horseshoe logistic regression
## data {
##   int<lower=0> N_train;
##   int<lower=0> P;
##
##   matrix[N_train,P] X_train;
##   int<lower=0,upper=1> y_train[N_train];
##
##   int<lower=0> N_test;
##
```

```
##   matrix[N_test,P] X_test;
##   int<lower=0,upper=1> y_test[N_test];
##
##   real<lower=0> scale_global;
##
##   real<lower=1> nu_local;
##   real<lower=1> nu_global;
##
##   real<lower=0> slab_scale;
##   real<lower=0> slab_df;
##
## }
##
##
## parameters {
##
##   real<lower=0> tau; // global shrinkage parameter
##   vector<lower=0> [P] lambda; // local shrinkage parameter
##   real alpha;     // intercept
##   vector[P] z;
##   real<lower=0> caux;
## }
##
## transformed parameters {
##
##   vector[P] beta; // regression coefficients
##   vector[N_train] f;
##   vector<lower=0>[P] lambda_tilde;
##   real<lower=0> c;
##
##   c = slab_scale * sqrt(caux);
##   lambda_tilde = sqrt( c^2 * square(lambda) ./ (c^2 + tau^2*square(lambda)) );
##   beta = z .* lambda_tilde*tau;
##   f = alpha + X_train*beta;
## }
##
##
## model {
##
##   alpha ~ normal(0,1000);
##   lambda ~ student_t(nu_local, 0, 1);
##
##   for (j in 1:P){
##
##        z[j] ~ normal(0,1) ;
##   }
##   tau ~ student_t(nu_global, 0, scale_global);
##   caux ~ inv_gamma(0.5*slab_df, 0.5*slab_df);
##   y_train ~ bernoulli_logit(f);
## }
##
## generated quantities {
##
##
```

```
##  int<lower=0,upper=1> y_pred[N_test];
##  vector[N_test] log_lik;
##
##  for (i in 1:N_test){
##
##    log_lik[i] = -bernoulli_logit_lpmf(y_test[i] | alpha+X_test[i,]*beta);
##    y_pred[i] = bernoulli_logit_rng(alpha+X_test[i,]*beta);
##
##
##  }
##
##
##
##
## }
##
##
```

- The stan code for the naive bayes classifier

```
writeLines(readLines(naivebayes_file))
```

```
## data {
##   int<lower=0> N_benign; // Number of benign training data samples
##   int<lower=0> N_malignant; // Number of malignant training data samples
##   int<lower=0> N_test; // Number of test samples
##   int<lower=0> J; // Number of features
##   vector[J] samples_benign[N_benign]; // Benign samples
##   vector[J] samples_malignant[N_malignant]; // Malignant samples
##   vector[J] samples_test[N_test]; // Test samples
##   vector<lower=0, upper=1>[N_test] y_test ; // Test samples' labels; 1 for Malignant, 0 for benign
##   real<lower=0, upper=1> prior_p; // Prior probability for a sample being malignant, equal to N_malig
## }
##
## parameters {
##   vector[J] mu_benign;
##   vector<lower=0>[J] sigma_benign;
##   vector[J] mu_malignant;
##   vector<lower=0>[J] sigma_malignant;
## }
##
## model {
##   for (j in 1:J) {
##      mu_benign[j] ~ normal(0, 10);
##      sigma_benign[j] ~ inv_chi_square(0.1);
##      mu_malignant[j] ~ normal(0, 10);
##      sigma_malignant[j] ~ inv_chi_square(0.1);
##   }
##
##   for (j in 1:J) {
##     samples_benign[, j] ~ normal(mu_benign[j], sigma_benign[j]);
##     samples_malignant[, j] ~ normal(mu_malignant[j], sigma_malignant[j]);
##   }
## }
##
```

```
## generated quantities {
##   int<lower=0,upper=1> y_pred[N_test];
##   real log_lik[N_test];
##
##   for (i in 1:N_test) {
##     real malignant_p = bernoulli_lpmf(1 | prior_p);
##     real benign_p = bernoulli_lpmf(0 | prior_p);
##     real likelihood_malignant_p = 0;
##     real likelihood_benign_p = 0;
##     for (j in 1:J) {
##       likelihood_malignant_p += normal_lpdf(samples_test[i, j] | mu_malignant[j], sigma_malignant[j]]
##       likelihood_benign_p += normal_lpdf(samples_test[i, j] | mu_benign[j], sigma_benign[j]);
##     }
##     malignant_p += likelihood_malignant_p;
##     benign_p += likelihood_benign_p;
##     if (malignant_p > benign_p) {
##       y_pred[i] = 1;
##     } else {
##       y_pred[i] = 0;
##     }
##     if (y_test[i] == 1) {
##       log_lik[i] = likelihood_malignant_p / (likelihood_malignant_p + likelihood_benign_p);
##     } else {
##       log_lik[i] = likelihood_benign_p / (likelihood_malignant_p + likelihood_benign_p);
##     }
##   }
## }
```