Comparative Analysis of Logistic Lasso, Ridge, and Bayesian Logistic Regression Models for Pumpkin Seed Classification

MSDS 567: Stat Modeling and Computation Final Project

Jiechun Lin, Yifei Chen, Sitong Liu

Abstract

- **Introduction**: Pumpkin seeds are often consumed as confectionery worldwide because of their adequate protein, fat, carbohydrate and mineral content. This study was conducted on two of the most important premium pumpkin seeds "Ürgüp Sivrisi" and "Çerçevelik", which are usually grown in the Ürgüp and Karacaören regions of Turkey. In this project, 2500 pumpkin seeds of these two varieties were classified using Lasso, Ridge and Bayesian Logistic Regression models and the model effects were compared.
- **Objective**: The primary objective of this project was to **explore Bayesian logistic regression models and compare their performance to classical logistic Lasso and Ridge models**.
- **Results**: Our analysis demonstrated that the various Bayesian models tested did not outperform the classical logistic Lasso and Ridge regression models. However, this experiment provided valuable insights into the role of different predictors and the potential of Bayesian modeling in this context.

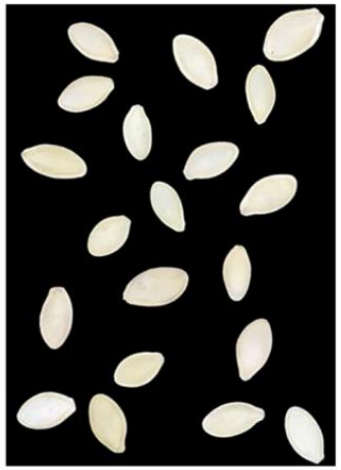
Methodology

- **Data Preprocessing**: Normalize the data, divide the dataset into training and test sets (8:2). Analyze the training set both with and without outliers. Ensure that the test set is isolated for unbiased performance evaluation.
- **Logistic Lasso and Ridge Regression Models**: Fit the logistic Lasso and Ridge regression models to the training set. Investigate the effects of the presence or absence of outliers on model performance.
- **Model Performance Evaluation**: Evaluate the performance of the Lasso and Ridge logistic regression models on the isolated test set. Use metrics such as accuracy, precision, recall, specificity, and F1-score to compare model performance.
- **Bayesian Logistic Regression 1(Non-informative predictor selection process):** Fit a Bayesian logistic regression model using RStan. Set the prior for the intercept as N(0, 10) and the prior for coefficients as Laplace(0, 10). Perform convergence diagnostics on the Bayesian logistic regression model to ensure that the MCMC sampling has converged. Examine the density plots of the posterior distributions for the predictors. Analyze the correlations among the predictors to determine if any should be removed or kept in the model based on their relevance.
- **Bayesian Logistic Regression 2(Informative predictor selection process):** Use the predictors selected from the logistic LASSO regression model and perform MCMC sampling. Compare the coefficients and model performance. Analyze the differences to draw insights.
- **Model Comparison and Threshold Adjustment**: Compare the performance of the Lasso, Ridge, and Bayesian logistic regression models using the evaluation metrics.

Dataset Description

- The dataset consists of 2,500 pumpkin seeds from two classes, with 12 features collected by processing the pixel information of their images. These features represent the morphological characteristics of the seeds for each class.

(a) Ürgüp Sivrisi



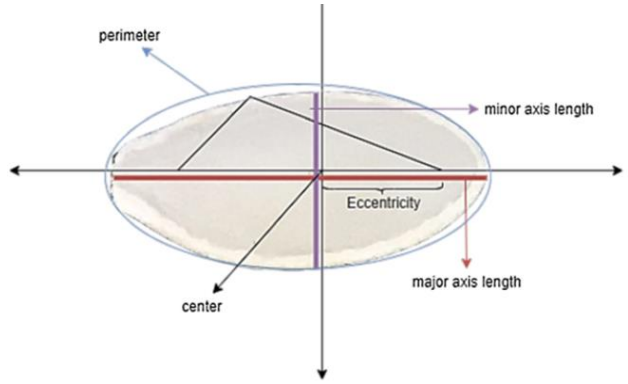
(b) Çerçevelik

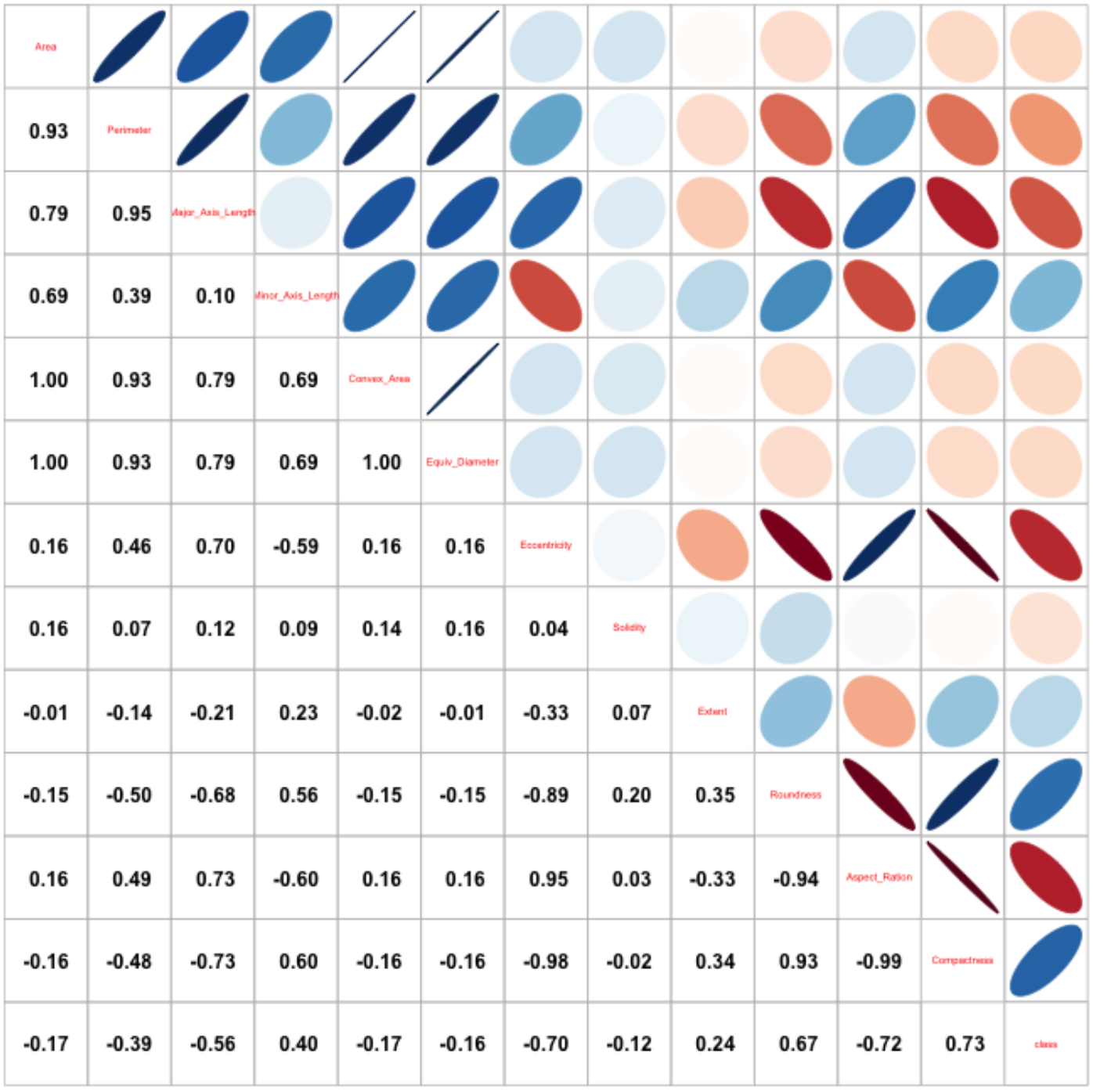
Ad	Piece
Çerçevelik	1300
Ürgüp Sivrisi	1200
Total	**2500**

- https://www.muratkoklu.com/datasets/vtdhnd05.php
- https://doi.org/10.1007/s10722-021-01226-0

Data Exploration

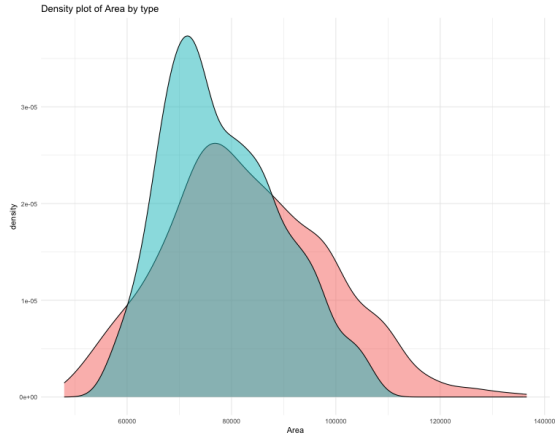
No	Name	Explanation
1	Area (A)	It gave the number of pixels within the borders of a pumpkin seed
2	Perimeter (p)	It gave the circumference in pixels of a pumpkin seed
3	Major Axis Length (Maj.AL)	It gave the circumference in pixels of a pumpkin seed
4	Minor Axis Length (Min.AL)	It gave the small axis distance of a pumpkin seed
5	Eccentricity (e)	It gave the eccentricity of a pumpkin seed
6	Convex Area (CA)	It gave the number of pixels of the smallest convex shell at the region formed by the pumpkin seed
7	Extent (E)	It returned the ratio of a pumpkin seed area to the bounding box pixels
8	Equiv Diameter (ED)	It was formed by multiplying the area of the pumpkin seed by four and dividing by the number pi, and taking the square root
9	Compactness (C)	It proportioned the area of the pumpkin seed relative to the area of the circle with the same circumference
10	Solidity (s)	It considered the convex and convex condition of the pumpkin seeds
11	Roundness (r)	It measured the ovality of pumpkin seeds without considering its distortion of the edges
12	Aspect Ratio (AR)	It gave the aspect ratio of the pumpkin seeds

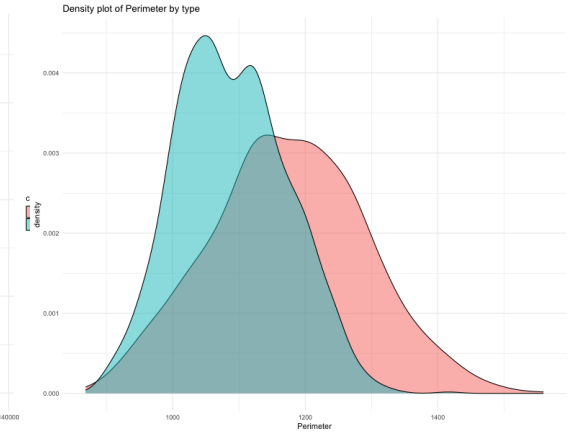


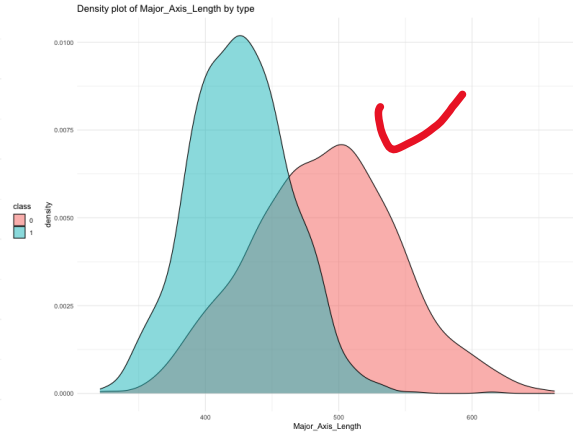


- The correlation plots show that the data has severe multicollinearity.

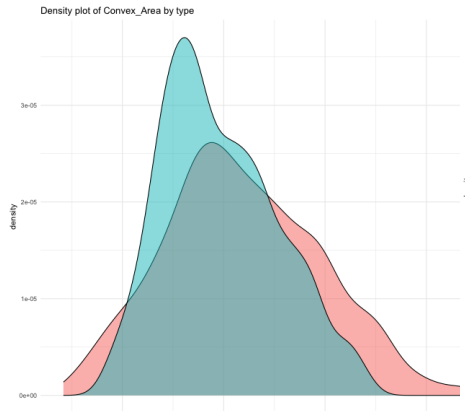
- These features - *Major_Axis_Length, Minor_Axis_Length, Eccentricity, Roundness, Aspect_Ratio,* and *Compactness* - show the most distinct differences between classes in the feature density plots(next slide), indicating that they may have a significant influence on the model.

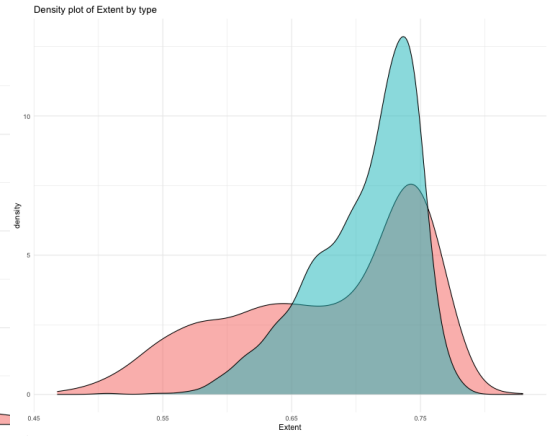

Density plot of Area by type
density
Area
class
0
1


Density plot of Perimeter by type
density
Perimeter
class
0
1

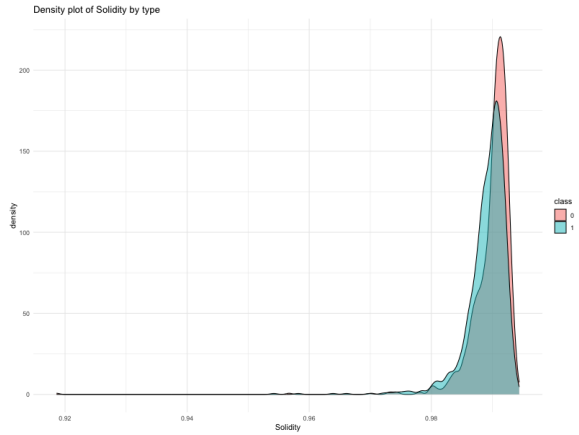

Density plot of Major_Axis_Length by type
density
Major_Axis_Length
class
0
1

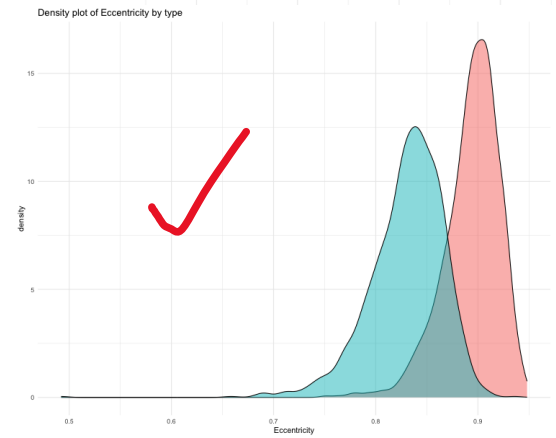

Density plot of Minor_Axis_Length by type
density
Minor_Axis_Length
class
0
1

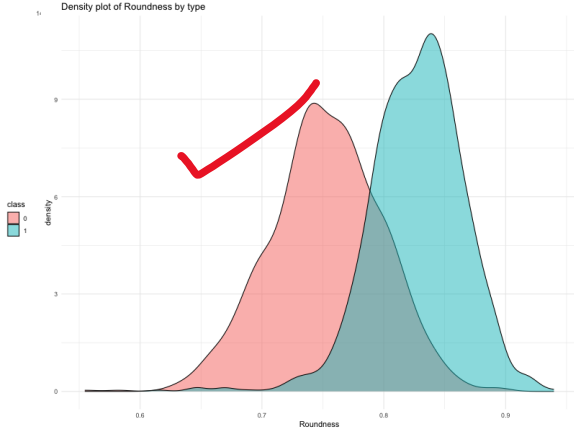

Density plot of Convex_Area by type
density
class
0
1

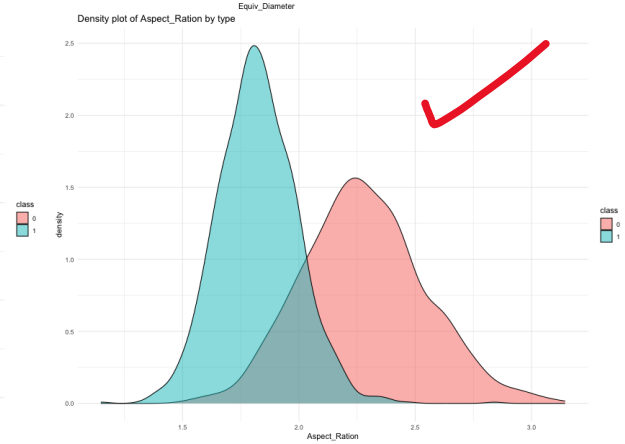

Density plot of Extent by type
density
Extent
class
0
1

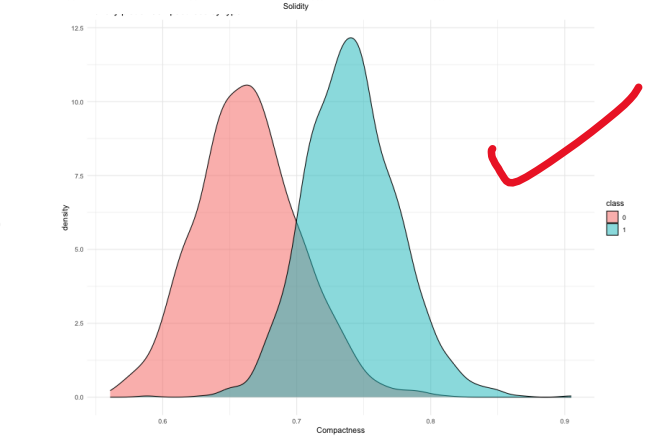

Density plot of Equiv_Diameter by type
density
Equiv_Diameter
class
0
1


Density plot of Solidity by type
density
Solidity
class
0
1


Density plot of Eccentricity by type
density
Eccentricity
class
0
1


Density plot of Roundness by type
density
Roundness
class
0
1


Density plot of Aspect_Ration by type
density
Aspect_Ration
class
0
1


density
Compactness
class
0
1

Data Pre-Processing

- Make sure no NAs
- Set 'Çerçevelik' as 1, 'Urgup Sivrisi' as 0
- Normalize the predictors: center and scale
- Stratified Train-Test Split

Not removing the outliers

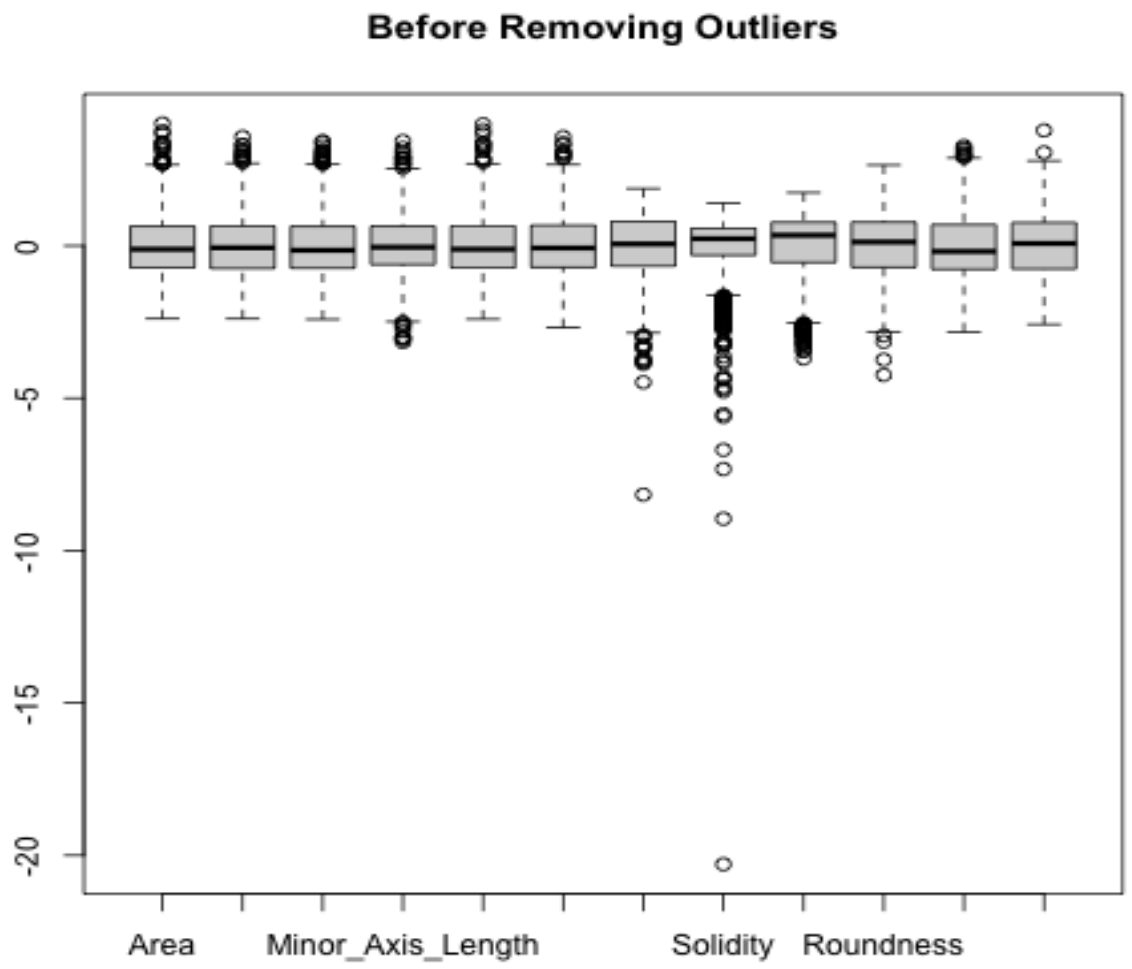
```
train_set:
   0    1
 955 1045
test_set:
  0   1
245 255
```

Removing the outliers

```
train_set_sub:
  0   1
842 988
test_set_sub:
  0   1
219 238
```

170 rows removed.

Outliers Detection



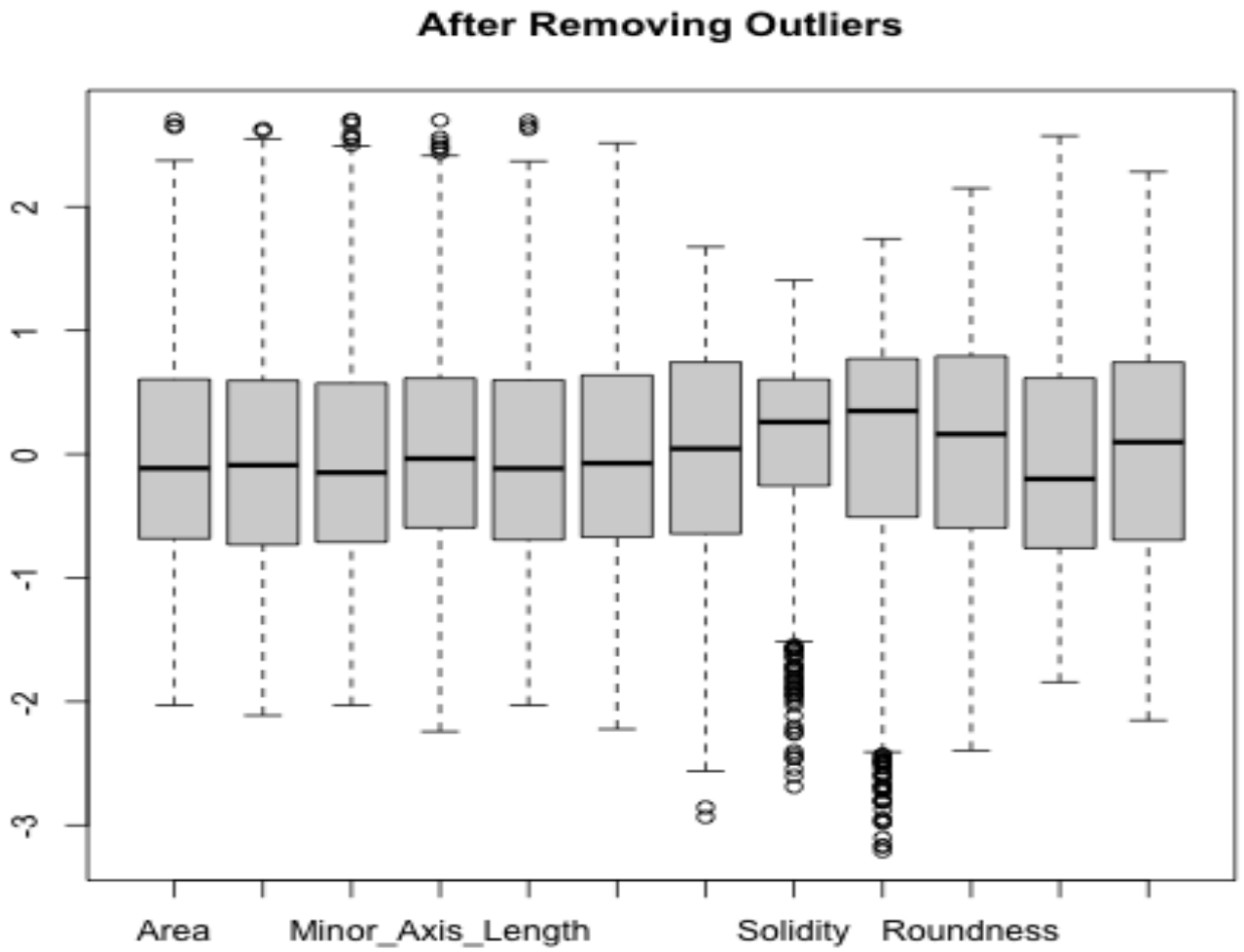
- The Mahalanobis distance :

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

- μ is the mean of the training data, and $\mathbf{S}^{-1}$ is the inverse covariance matrix of the training data.

- This distance metric takes into account the correlations between variables, which makes it a useful tool for handling multivariate data.

Removing Outliers



170 rows removed.

Logistic Regression

- Logistic regression is a statistical method for analyzing a dataset where the response variable is binary. It is used to model the probability of a certain class or event occurring based on one or more predictor variables.

$$\Pr(Y = 1|X_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip})}}$$

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}$$

- p: probability of the event $(Y = 1|X_i)$ occurring
- β_0: intercept (log-odds of the event when all predictors are zero)
- $\beta_1, \ldots, \beta_p$: coefficients representing the effect of the predictors $(X_1, \ldots, X_p)$ on the log-odds of the event.

Dealing with Multicollinearity

- Multicollinearity occurs when the predictor variables in a logistic regression model are highly correlated with each other, leading to unstable and imprecise coefficient estimates. This can make it difficult to determine the individual effect of each predictor variable on the outcome.
- Lasso and Ridge regression are two commonly used methods for handling multicollinearity in logistic regression.

Logistic Lasso Regression

- In Lasso, the penalty term is the sum of the absolute values of the coefficients(L1 penalty), which has the effect of **shrinking some** coefficients towards zero and **setting some coefficients to zero** entirely. This leads to variable selection and automatic feature selection, effectively reducing the number of predictor variables in the model.

$$\hat{\beta} = \text{argmin}_{\beta} \quad L(\beta) = -\left[\sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i)\right] + \lambda \sum_{j=1}^{p} |\beta_j|$$

Logistic Ridge Regression

- In Ridge regression, the penalty term is the sum of the squares of the coefficients(L2 penalty), which has the effect of **shrinking all** the coefficients towards zero, but **not setting any coefficients to zero**. This leads to shrinkage of the coefficients, reducing the impact of correlated predictor variables, but does not result in variable selection.

$$\hat{\beta} = \text{argmin}_{\beta} \quad L(\beta) = -\left[\sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2$$

The Performance Metrics

- **Accuracy = (TP + TN) / (TP + TN + FP + FN)**
  - The proportion of correctly classified instances out of the total instances.
- **Precision= TP / (TP + FP)**
  - The proportion of true positives out of the total predicted positives. It indicates the accuracy of the positive predictions made by the model.
- **Recall= TP / (TP + FN)**
  - The proportion of true positives out of the total actual positives. It indicates the ability of the model to correctly identify positive instances.
- **Specificity= TN / (TN + FP)**
  - The proportion of true negatives out of the total actual negatives. It indicates the ability of the model to correctly identify negative instances.
- **F1-score=2 * (Precision * Recall) / (Precision + Recall)**
  - The harmonic mean of precision and recall. It provides a single metric that balances both precision and recall. It is particularly useful when dealing with imbalanced datasets, where the accuracy metric can be misleading.

LogisticLasso Regression

- A 10-fold CV based on the accuracy of the model is performed. Setting the threshold as 0.5, the test set accuracy before removing outliers is 0.868, and after removing outliers is 0.864.

Before	After
Accuracy: 0.868	Accuracy: 0.864
Precision: 0.8487085	Precision: 0.8351254
Recall: 0.9019608	Recall: 0.9137255
Specificity: 0.8326531	Specificity: 0.8122449
F1-score: 0.8745247	F1-score: 0.8726592

- The two models have quite similar performance. Model 1 (with outliers) has a slightly higher accuracy, precision, specificity, and F1-score, whereas Model 2 (without outliers) has a higher recall.

Logistic Lasso Regression

```
$lambda
[1] 0.01

$intercept
      s0
0.520097

$beta
12 x 1 sparse Matrix of class "dgCMatrix"
                           s0
Area                .
Perimeter           .
Major_Axis_Length -0.014201437
Minor_Axis_Length   .
Convex_Area         .
Equiv_Diameter      .
Eccentricity        .
Solidity           -0.058495715
Extent              .
Roundness           0.063653658
Aspect_Ration      -0.003888242
Compactness         0.276571762
```

```
$best_model

$lambda
[1] 0.05

$intercept
       s0
0.5326153

$beta
12 x 1 sparse Matrix of class "dgCMatrix"
                           s0
Area                .
Perimeter           .
Major_Axis_Length -0.00414198
Minor_Axis_Length   .
Convex_Area         .
Equiv_Diameter      .
Eccentricity        .
Solidity           -0.05165421
Extent              .
Roundness           .
Aspect_Ration      -0.10354786
Compactness         0.24040622
```

- Model 2 has a slightly simpler model (using one less variable).
- If the future data also contains outliers, the model trained with outliers may perform better.

Lasso Model Interpretation

- The density plots of each feature show that *Major_Axis_Length*, *Minor_Axis_Length*, *Eccentricity*, *Roundness*, *Aspect_Ration*, and *Compactness* have the most distinguishable differences between the two classes.
- The beta coefficients for each variable in both models show that:
  - *Major_Axis_Length* and A*spect_Ration* have negative coefficients for class 1, meaning that as these variables decrease, the likelihood of the observation being in class 1 increases.
  - *Roundness*(only kept in model 1) and compactness have positive coefficients for class 1, meaning that as these variables increase, the likelihood of the observation being in class 1 increases.
  - *Minor_Axis_Length* and *Eccentricity* were removed from both models due to their high covariance with *aspect _ration* and *compactness*.
- Therefore, the feature selection output supports the observation that class 1 has a shorter major axis, higher roundness, lower aspect ration, and higher compactness.

Logistic Ridge Regression

- Using the same setting, only fits to the data after removing outliers:

Before

```
$best_model

$lambda
[1] 0.05

$intercept
      s0
0.520206

$beta
12 x 1 sparse Matrix of class "dgCMatrix"
                          s0
Area              -0.024404165
Perimeter         -0.009424144
Major_Axis_Length -0.043103168
Minor_Axis_Length  0.065299592
Convex_Area       -0.019947680
Equiv_Diameter     0.014161433
Eccentricity      -0.029540934
Solidity          -0.063464101
Extent             0.003557799
Roundness          0.078151975
Aspect_Ration     -0.076969261
Compactness        0.091395578
```

After

```
$best_model

$lambda
[1] 0.07272727

$intercept
       s0
0.5369842

$beta
12 x 1 sparse Matrix of class "dgCMatrix"
                          s0
Area              -0.018009135
Perimeter         -0.010252939
Major_Axis_Length -0.054546683
Minor_Axis_Length  0.066815742
Convex_Area       -0.015815234
Equiv_Diameter     0.005176806
Eccentricity      -0.068036459
Solidity          -0.113444473
Extent             0.001418848
Roundness          0.045712241
Aspect_Ration     -0.092368266
Compactness        0.093296440
```

Before	After
Accuracy: 0.866	Accuracy: 0.862
Precision: 0.8405797	Precision: 0.8369565
Recall: 0.9098039	Recall: 0.9058824
Specificity: 0.8204082	Specificity: 0.8163265
F1-score: 0.873823	F1-score: 0.8700565

- The performance of the two models are close.

Ridge Model Interpretation

- The coefficients have been shrunk towards zero due to the regularization.
- Both models have similar coefficient patterns, with some differences in magnitudes.
- For both models, the coefficients with the largest magnitude are *Aspect_Ration* and *Compactness*. These features seem to have the most significant impact on the target variable. This again supports the data observation.
- The signs of the coefficients are consistent between the two models, which indicates that the direction of the relationship between the features and the response is the same for both models.

Between Lasso and Ridge

- The performance of both Ridge and Lasso regression models with the data is close, choosing between them depends on the specific problem.
- Lasso regression can set some coefficients to exactly zero, effectively performing feature selection. If a simpler model is favored, Lasso might be more appropriate.
- Ridge regression can be more stable in the presence of noise compared to Lasso regression. If the stability of the model is a concern, Ridge might be a better choice.

Trained with or without Outliers

- Since both the lasso and ridge models have similar performance, feature selection, and coefficient magnitudes with or without outliers, it is reasonable to believe that the presence of outliers in the training dataset may not have had a significant impact on the model's ability to generalize. Alternatively, there could be outliers present in the test set(and we suggest this is the case in the pumkin seeds industry).
- Therefore, we choose to **keep the outliers in the dataset** and proceed with training the Bayesian models.

Bayesian Logistic Model

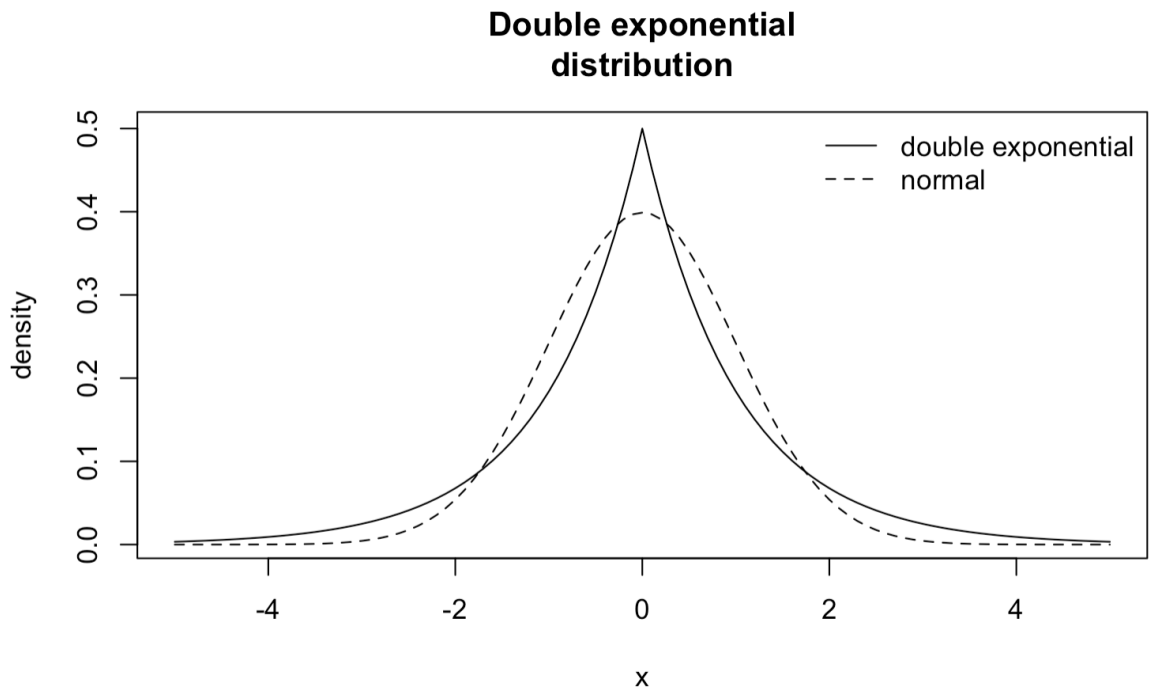
- Bayesian logistic regression is an approach to logistic regression that incorporates Bayesian inference, allowing us to model uncertainties in the estimated parameters.
- In Bayesian logistic regression, we do not obtain a single set of coefficients as in frequentist logistic regression; instead, we estimate a posterior distribution of the coefficients given the data.
- The posterior distribution provides a measure of uncertainty in the estimated coefficients and can be used to make predictions or draw inferences about the relationship between the predictor variables and the binary outcome.

Weakly Informative Priors

- To perform Bayesian logistic regression, we need to specify prior distributions for the model parameters. The most common choices for priors are non-informative or weakly informative priors.
- **Non-informative priors** are used when there is limited prior knowledge about the parameters or when the aim is to **let the data drive the inference.**
- Weakly informative priors include uniform priors (e.g.,U(-100, 100)), normal distribution priors with mean 0 and a large variance (e.g.,N(0, σ^2)),Laplace (double exponential) priors mean 0 and a large scale.

Laplace Prior

- **Laplace priors, also known as double exponential priors**, is less informative than the normal prior and can lead to **sparser solutions** by placing more probability **mass on the tails** of the distribution.
- The Laplace prior **allows** for the possibility of some **large coefficient values** (representing strong effects), while still encouraging small or zero values for less important variables. This prior acts as a form of regularization, leading to more interpretable models with fewer non-zero coefficients.



Bayesian Lasso Logistic Model

The Bernoulli likelihood for a set of points $\{(x_i, y_i)\}$ in a Bayesian logistic regression model is given by:

$$P(Y_i = y_i | X_i = x_i, \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})}}^{y_i} \left(1 - \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})}}\right)^{(1-y_i)}$$

Laplace priors are assigned to the coefficients:

$$\beta_j \sim \text{Laplace}(\mu_j, b_j) \quad \text{for} \quad j = 1, \ldots, p$$

A normal prior is assigned to the intercept:

$$\beta_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$$

The posterior distribution for the set of points $\{(x_i, y_i)\}$ in the Bayesian logistic regression model is:

$$P(\beta_0, \beta_1, \ldots, \beta_p | \{Y_i, X_i\}_{i=1}^N) \propto \prod_{i=1}^{N} P(Y_i = y_i | X_i = x_i, \beta_0, \beta_1, \ldots, \beta_p) \times P(\beta_0) \times P(\beta_1) \times \ldots \times P(\beta_p)$$

- This model employs a **Laplace prior for the coefficients** and a **normal prior for the intercept** of the log odds. These are non-conjugate priors
- The scale parameter (tau) for the Laplace prior can have its own hyperprior, such as the Cauchy distribution. This is the hierarchical setting, where a hyperprior is placed on the parameter of the prior distribution(not covred in this project).

JAGS in R

- The rjags package provides an interface from R to the JAGS library for Bayesian data analysis. JAGS uses Markov Chain Monte Carlo (MCMC) to generate a sequence of dependent samples from the posterior distribution of the parameters.
- General JAGS process:
- **Define the model in JAGS syntax**: Write the Bayesian model in JAGS syntax using a separate text file or a character string within R.
- **Prepare the data**: Organize the data in a list format in R, with each variable being an element of the list.
- **Set initial values**: Provide initial values for the parameters of the model.
- **Create the JAGS model**: Use the jags.model() function to create a JAGS model object. Provide the model file or character string, data, initial values, and any other necessary arguments.

JAGS with Non-informative Priors

- In the non-informative setting, randomly set
- int ~ N(0, 5^2) (in JAGS is dnorm(mean, 1/variance))
- b ~ ddexp(0, sqrt(2))

```
mod1_string = " model {
    for (i in 1:length(y)) {
        y[i] ~ dbern(p[i])
        logit(p[i]) = int + b[1]*Area[i] + b[2]*Perimeter[i] +
b[3]*Major_Axis_Length[i] + b[4]*Minor_Axis_Length[i] + b[5]*Convex_Area[i] +
        b[6]*Equiv_Diameter[i]+b[7]*Eccentricity[i]+b[8]*Solidity[i]+b[9]*Extent[i]+b[1
0]*Roundness[i]+b[11]*Aspect_Ration[i]+
        b[12]*Compactness[i]
    }
    int ~ dnorm(0.0, 1.0/25.0)
    for (j in 1:12) {
        b[j] ~ ddexp(0.0, sqrt(2.0))
    }
}"
```

Define the model string

```
data_jags = list(y=train_set$class,
                 Area=X_train['Area'],
                 Perimeter=X_train['Perimeter'],
                 Major_Axis_Length=X_train['Major_Axis_Length'],
                 Minor_Axis_Length=X_train['Minor_Axis_Length'],
                 Convex_Area=X_train['Convex_Area'],
                 Equiv_Diameter=X_train['Equiv_Diameter'],
                 Eccentricity=X_train['Eccentricity'],
                 Solidity=X_train['Solidity'],
                  Extent=X_train['Extent'],
                 Roundness=X_train['Roundness'],
                 Aspect_Ration=X_train['Aspect_Ration'],
                 Compactness=X_train['Compactness'])

params = c("int", "b")
```

Prepare the data

```
mod1 = jags.model(textConnection(mod1_string), data=data_jags, n.chains=3)
update(mod1, 2e3)
mod1_sim = coda.samples(model=mod1,
                        variable.names=params,
                        n.iter=1e4)
mod1_csim = as.mcmc(do.call(rbind, mod1_sim))
```

Create the model and run the MCMC

Try Several Simulations

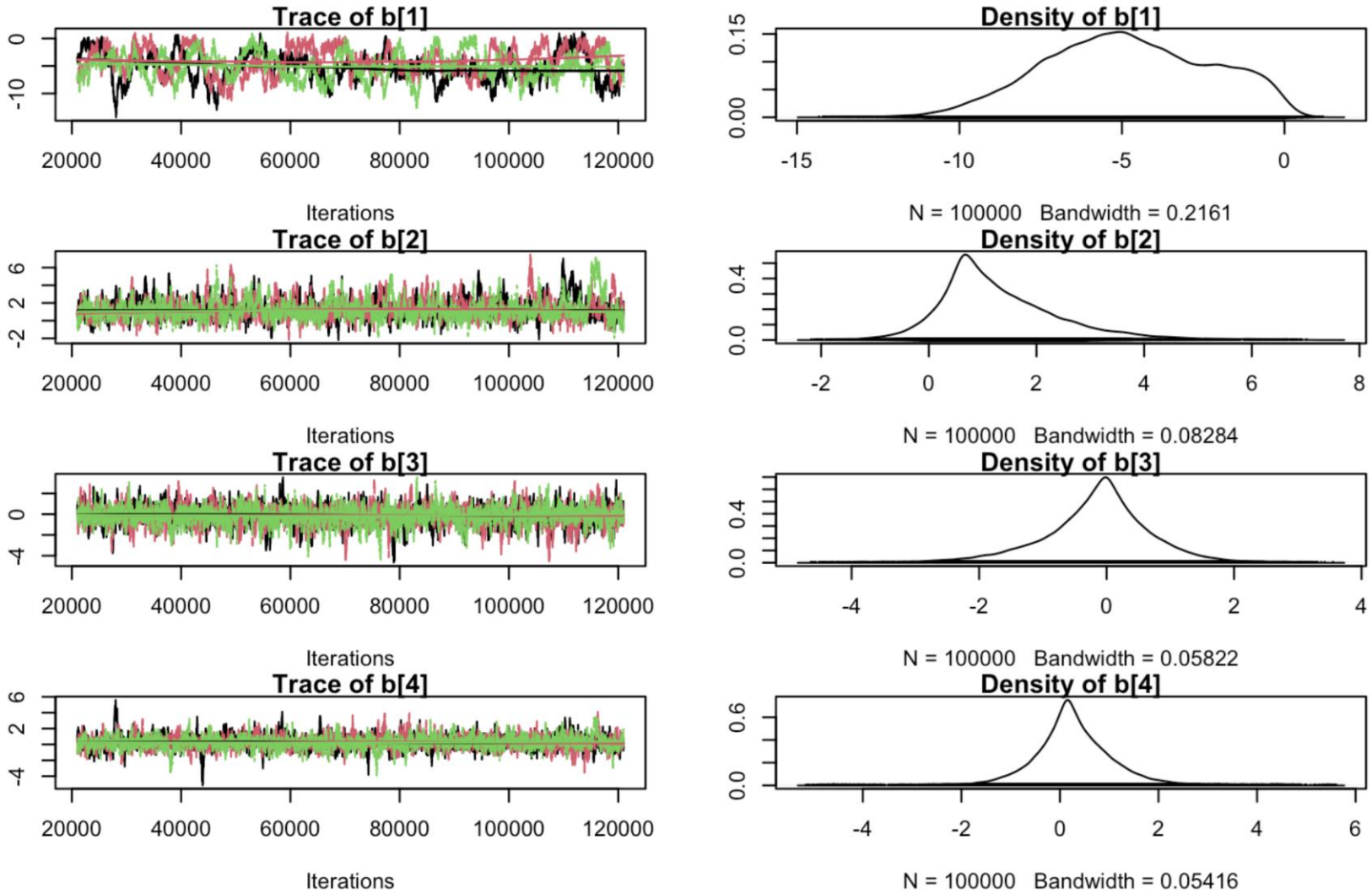
- Mod1(non-infromative, chain=3, burin-in=4000, iter=20000)
- Mod2(informative, chain=3,burn-in=6000,iter=30000)
- Mod3(informative, chian=3, burin-in=20000,iter=100000)
- However, the **mixing** in this model is **unsatisfactory**. Convergence diagnostics, including Gelman-Rubin, effective sample size, autocorrelation diagnostics, and trace plots, indicate that increasing the number of iterations and the burn-in period does not significantly improve convergence. Furthermore, the computational process is **highly time-consuming**, requiring hours to complete a single run.

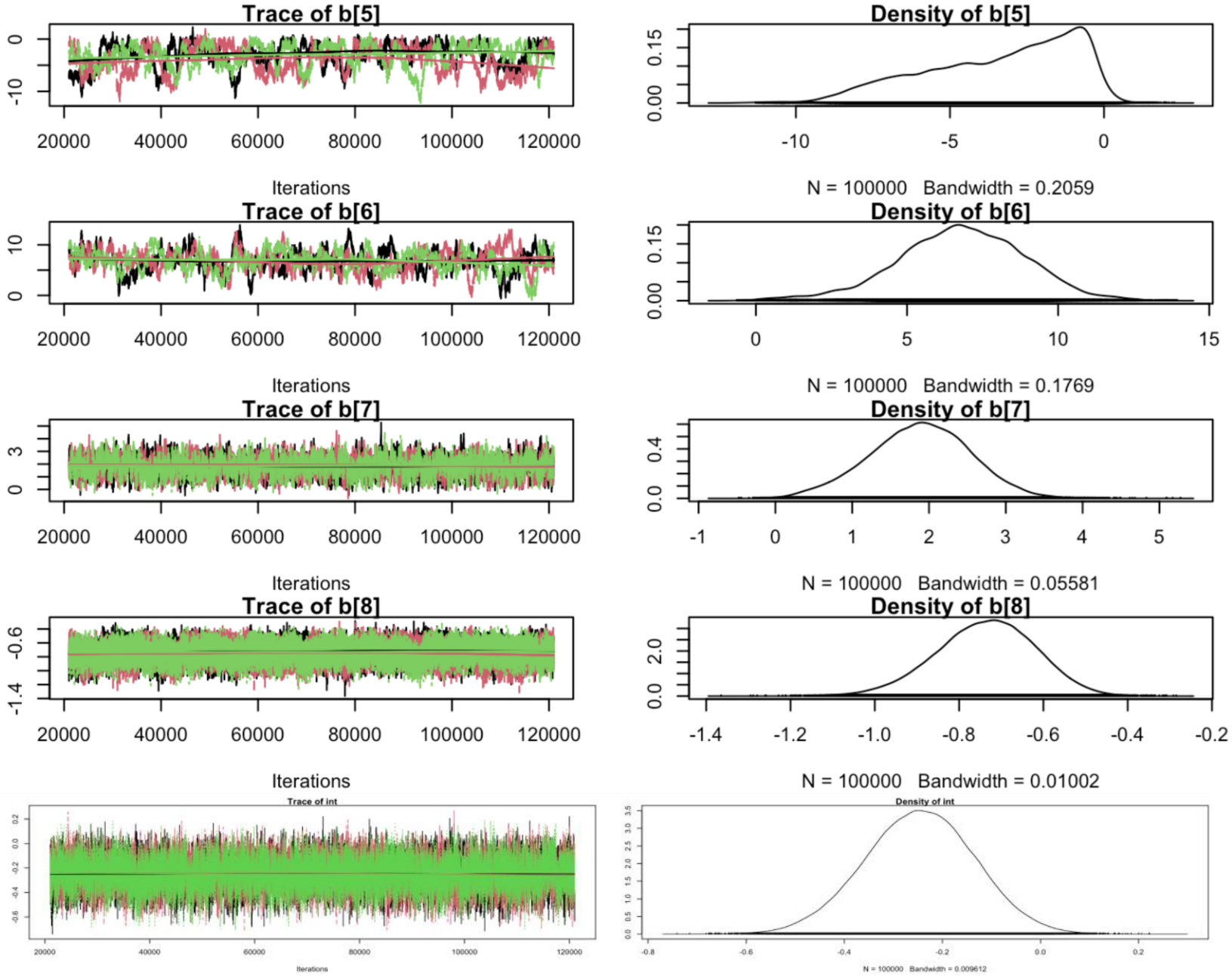
```
Summary:

Iterations = 21001:121000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 1e+05

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean      SD  Naive SE Time-series SE
b[1]  -4.8135 2.54010 0.0046376      0.2310378
b[2]   1.3153 1.13503 0.0020723      0.0451043
b[3]  -0.1515 0.86518 0.0015796      0.0232606
b[4]   0.2713 0.79952 0.0014597      0.0226923
b[5]  -3.3746 2.42035 0.0044189      0.2130626
b[6]   6.7830 2.10189 0.0038375      0.1657304
b[7]   1.8791 0.65592 0.0011975      0.0142655
b[8]  -0.7375 0.11773 0.0002149      0.0014656
b[9]  -0.0162 0.08377 0.0001529      0.0002047
b[10]  1.1110 0.49755 0.0009084      0.0160708
b[11] -1.1651 1.12453 0.0020531      0.0312186
b[12]  3.0301 1.51893 0.0027732      0.0522746
int   -0.2469 0.11296 0.0002062      0.0012588

2. Quantiles for each variable:

         2.5%      25%      50%      75%    97.5%
b[1]  -9.7068 -6.62619 -4.88958 -2.89168 -0.30283
b[2]  -0.4242  0.58102  1.07468  1.88564  4.10446
b[3]  -2.1061 -0.58437 -0.08422  0.33241  1.50328
b[4]  -1.2773 -0.15926  0.22377  0.69366  1.98000
b[5]  -8.4101 -5.13380 -2.87946 -1.33013 -0.11243
b[6]   2.3217  5.43724  6.82271  8.22361 10.70485
b[7]   0.5511  1.44177  1.89391  2.32861  3.13653
b[8]  -0.9796 -0.81465 -0.73307 -0.65542 -0.51940
b[9]  -0.1819 -0.07174 -0.01579  0.03939  0.14876
b[10]  0.2922  0.76737  1.05617  1.39116  2.25242
b[11] -3.5325 -1.94323 -1.05116 -0.28051  0.62970
b[12]  0.5140  1.83103  2.97802  4.10768  6.07372
int   -0.4716 -0.32258 -0.24567 -0.17012 -0.02849
```

```
Gelman Diagnostic:
Potential scale reduction factors:

      Point est. Upper C.I.
b[1]        1.13       1.40
b[2]        1.00       1.01
b[3]        1.00       1.00
b[4]        1.00       1.00
b[5]        1.13       1.40
b[6]        1.00       1.01
b[7]        1.00       1.01
b[8]        1.02       1.06
b[9]        1.00       1.00
b[10]       1.00       1.00
b[11]       1.00       1.00
b[12]       1.00       1.00
int         1.00       1.00

Multivariate psrf

1.1
```

```
Autocorrelation Diagnostic:
           b[1]      b[2]      b[3]      b[4]      b[5]      b[6]      b[7]      b[8]        b[9]
Lag 0  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.000000000
Lag 1  0.9992167 0.9955636 0.9896181 0.9912949 0.9991075 0.9989019 0.9717754 0.6317044 0.270118379
Lag 5  0.9960886 0.9785665 0.9500529 0.9577097 0.9956026 0.9945700 0.8783043 0.3445356 0.007205816
Lag 10 0.9922583 0.9584664 0.9044825 0.9186402 0.9913017 0.9891799 0.7854695 0.2451153 0.002567876
Lag 50 0.9632856 0.8194871 0.6379309 0.6691708 0.9596584 0.9470337 0.4505290 0.1766481 0.003377640
          b[10]     b[11]     b[12]       int
Lag 0  1.0000000 1.0000000 1.0000000 1.0000000
Lag 1  0.9585164 0.9887705 0.9942988 0.6791505
Lag 5  0.8476679 0.9471348 0.9720596 0.4959367
Lag 10 0.7763574 0.9004308 0.9446151 0.4062748
Lag 50 0.6263480 0.6531759 0.7465804 0.1410162

Effective Sample Size:
      b[1]       b[2]       b[3]       b[4]       b[5]       b[6]       b[7]       b[8]
  117.0450   638.1642  1411.9451  1244.8386   130.5084   162.4095  2117.9966  6833.0598
      b[9]      b[10]      b[11]      b[12]        int
167524.9916   972.9658  1299.6888   846.3588  8055.0644
```

- As shown above, in this setting, three chains of 100,000 iterations are run. Nevertheless, the convergence diagnostics indicate that the chains are far from convergence.


Trace of b[1]
Density of b[1]
Iterations
N = 100000 Bandwidth = 0.2161
Trace of b[2]
Density of b[2]
Iterations
N = 100000 Bandwidth = 0.08284
Trace of b[3]
Density of b[3]
Iterations
N = 100000 Bandwidth = 0.05822
Trace of b[4]
Density of b[4]
Iterations
N = 100000 Bandwidth = 0.05416



- A caterpillar shape indicates that the MCMC sampler is not mixing well, and suggests that the chains have not converged to the target distribution.
- In contrast, a trace plot with good mixing should not show any systematic patterns, and the points should be distributed randomly and uniformly across the range of the parameter values.

Hamiltonian Monte Carlo(HMC)

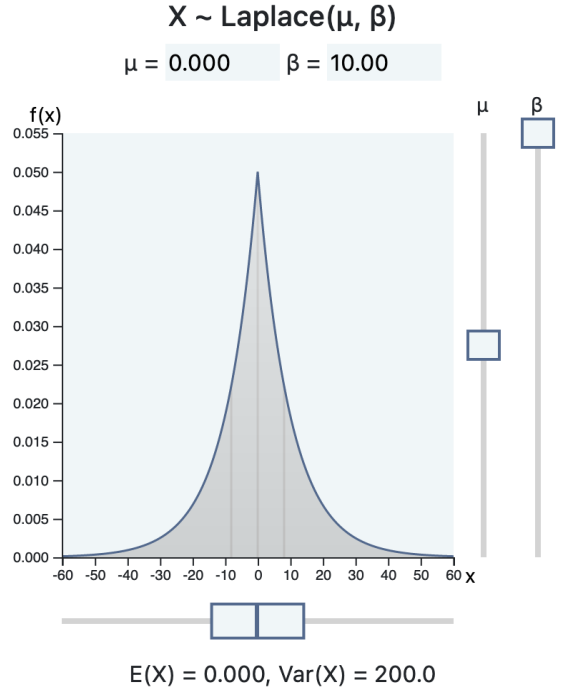
- JAGS samplers are Gibbs samplers, which means they sequentially update one parameter at a time, conditioning on the current values of all other parameters.
- HMC, on the other hand, is a more sophisticated MCMC algorithm that uses the **gradient of the log posterior distribution** to guide the exploration of the parameter space.
- It simulates a Hamiltonian system, a physical system where particles move under the influence of potential energy and kinetic energy.
- By reducing random walk behavior and improving the exploration of high-dimensional spaces, the algorithm can **make longer, more efficient jumps** through the parameter space than Gibbs samplers, which can improve convergence times and lead to better mixing of the chains.
- **JAGS** samplers are typically f**aster** and more straightforward to use for **simple models** with conjugate priors, while **HMC** is more **powerful for complex models**.

Why RStan?

- Stan is a powerful and flexible software for Bayesian statistical modeling that uses HMC. RStan is the R interface to Stan.
- *rstan* package provides a way to **set** the number of working **cores** for running MCMC simulations, by using the cores argument in the stan() function. This further improves sampling efficiency and **speeds up computations**.

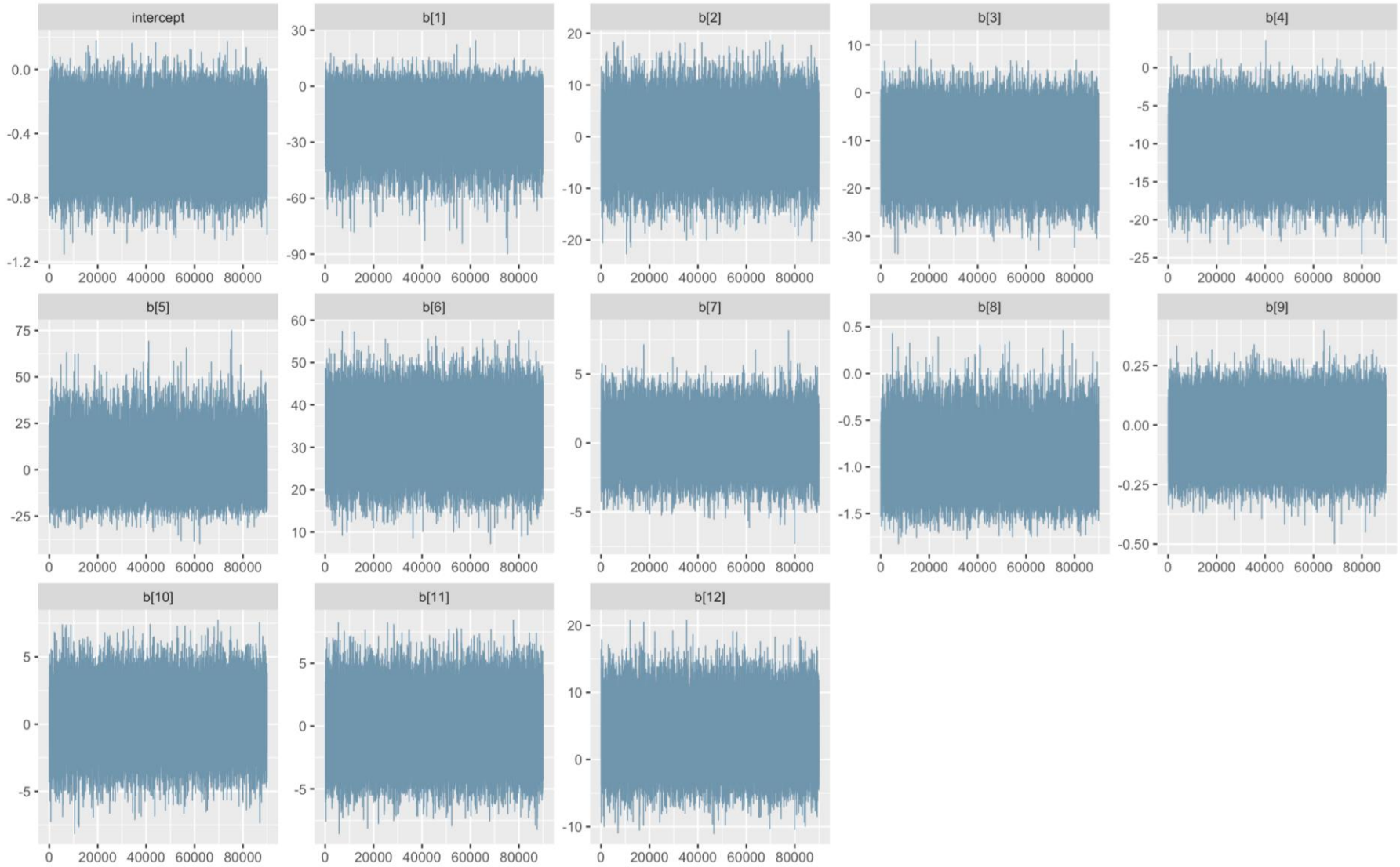
Model 1: Non-informative Priors

- 3 chains, 6000 warm-ups, 36000 iters
- intercept ~ normal(0, 10);
- b ~ double_exponential(0, 10)



$summary

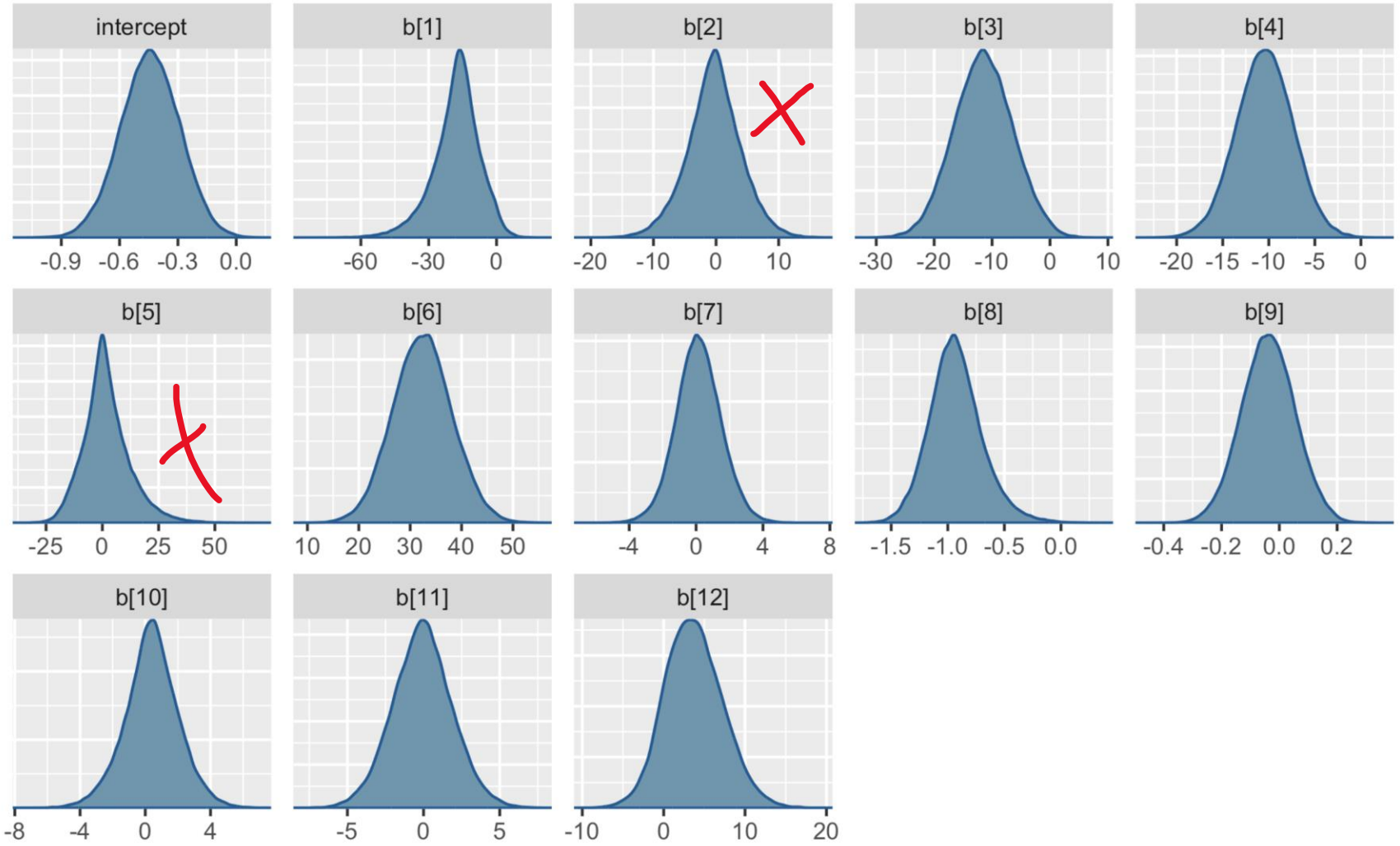
	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.44290093	0.0006677213	0.1528390	-0.7470997	-0.5449755	-0.44235030	-0.33959436	-0.1475183	52393.59	1.0000683
b[1]	-17.86212358	0.0529249477	10.3621631	-41.8099184	-23.3740765	-16.90739042	-11.35398882	0.2061233	38333.63	0.9999897
b[2]	-0.32301164	0.0210874811	4.3327099	-9.2742584	-2.9463389	-0.28068532	2.36663721	8.3512093	42215.29	0.9999994
b[3]	-11.48377980	0.0268726661	5.1009141	-21.4144730	-14.9386516	-11.49995569	-8.04154537	-1.4556357	36030.85	1.0000609
b[4]	-10.43558866	0.0152364292	3.0809916	-16.4950703	-12.4948755	-10.42254245	-8.36140303	-4.4078349	40889.77	1.0000447
b[5]	1.86751817	0.0534282644	10.3680947	-16.4717402	-4.4716494	0.85352545	7.24933798	25.9099291	37657.87	0.9999935
b[6]	32.38528765	0.0274968147	5.7695955	21.1824144	28.4903344	32.36568193	36.21549867	43.7754767	44027.70	1.0000400
b[7]	0.15412814	0.0072993765	1.3572557	-2.5158656	-0.7307337	0.13383323	1.02942096	2.8680577	34574.17	1.0000566
b[8]	-0.92725714	0.0011979570	0.2365083	-1.3626191	-1.0819484	-0.94025351	-0.78879363	-0.4086294	38977.18	0.9999909
b[9]	-0.03815968	0.0003200102	0.0926428	-0.2203729	-0.1006274	-0.03819162	0.02440634	0.1439441	83809.99	0.9999791
b[10]	0.39856626	0.0081425685	1.6594654	-3.0144432	-0.6084148	0.40927153	1.43504786	3.7120875	41534.94	1.0000208
b[11]	-0.07560755	0.0082438509	1.9628175	-3.9616891	-1.3701020	-0.07984204	1.19124646	3.8580288	56689.10	1.0000378
b[12]	3.71121729	0.0191713120	3.5727429	-2.9227306	1.2313062	3.56998269	6.05149077	11.0591751	34729.61	1.0000524
lp__	-571.38927111	0.0159109232	2.7174086	-577.5850852	-572.9795157	-571.06099836	-569.42673251	-567.0660533	29168.84	1.0000503



- The traceplots show decent mixing.

```
Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052
```

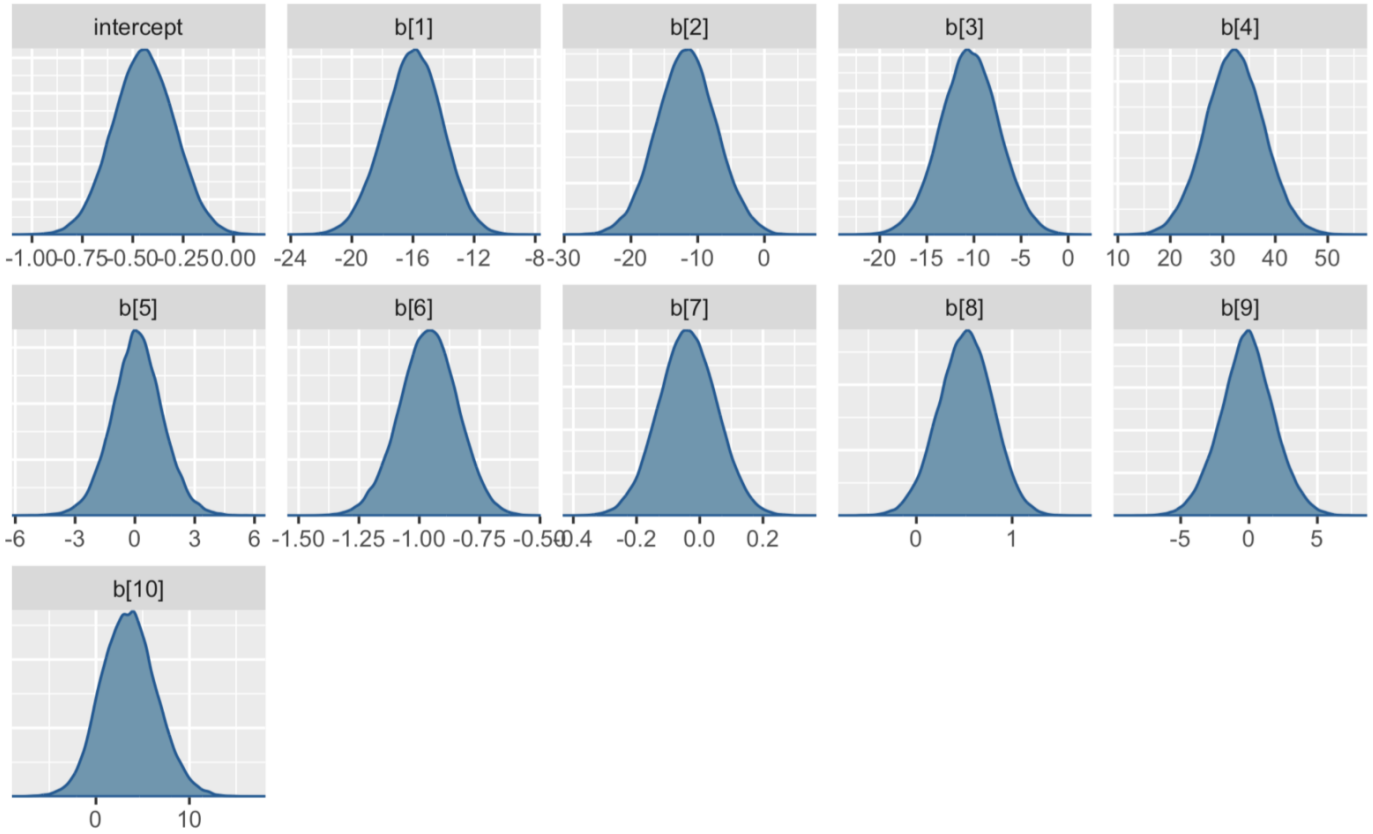
- These performance metrics indicate that the model with normal prior N(0, 10) and Laplace prior Laplace(0, 10) overall appears to be satisfactory. It achieves a good balance between predicting both positive and negative classes. the F1-score of 0.8671, which is a harmonic mean of precision and recall, shows that the model achieves a good balance between precision and recall.



- The posterior distribution of parameters b[2](*perimeter*),b[5](*Convex_Area*) are centered around 0 and resemble the normal or Laplace distribution priors. This suggests that the data have little influence on the parameter estimate. We can remove them and rerun the MCMC.

Model 2:

- 3 chains, 6000 warm-ups, 36000 iters
- remove b[2](perimeter),b[5](Convex_Area)
- intercept ~ normal(0, 10);
- b ~ double_exponential(0, 10)




```
Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052
```

- Convergence diagnostics show that the samples already converge. However, the performance of this model on the test set is the same as the previous one, suggesting that the removed predictors have minimal impact on the model's predictive performance.

Model 1(includes all the predictors)

$summary

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.44290093	0.0006677213	0.1528390	-0.7470997	-0.5449755	-0.44235030	-0.33959436	-0.1475183	52393.59	1.0000683
b[1]	-17.86212358	0.0529249477	10.3621631	-41.8099184	-23.3740765	-16.90739042	-11.35398882	0.2061233	38333.63	0.9999897
b[2]	-0.32301164	0.0210874811	4.3327099	-9.2742584	-2.9463389	-0.28068532	2.36663721	8.3512093	42215.29	0.9999994
b[3]	-11.48377980	0.0268726661	5.1009141	-21.4144730	-14.9386516	-11.49995569	-8.04154537	-1.4556357	36030.85	1.0000609
b[4]	-10.43558866	0.0152364292	3.0809916	-16.4950703	-12.4948755	-10.42254245	-8.36140303	-4.4078349	40889.77	1.0000447
b[5]	1.86751817	0.0534282644	10.3680947	-16.4717402	-4.4716494	0.85352545	7.24933798	25.9099291	37657.87	0.9999935
b[6]	32.38528765	0.0274968147	5.7695955	21.1824144	28.4903344	32.36568193	36.21549867	43.7754767	44027.70	1.0000400
b[7]	0.15412814	0.0072993765	1.3572557	-2.5158656	-0.7307337	0.13383323	1.02942096	2.8680577	34574.17	1.0000566
b[8]	-0.92725714	0.0011979570	0.2365083	-1.3626191	-1.0819484	-0.94025351	-0.78879363	-0.4086294	38977.18	0.9999909
b[9]	-0.03815968	0.0003200102	0.0926428	-0.2203729	-0.1006274	-0.03819162	0.02440634	0.1439441	83809.99	0.9999791
b[10]	0.39856626	0.0081425685	1.6594654	-3.0144432	-0.6084148	0.40927153	1.43504786	3.7120875	41534.94	1.0000208
b[11]	-0.07560755	0.0082438509	1.9628175	-3.9616891	-1.3701020	-0.07984204	1.19124646	3.8580288	56689.10	1.0000378
b[12]	3.71121729	0.0191713120	3.5727429	-2.9227306	1.2313062	3.56998269	6.05149077	11.0591751	34729.61	1.0000524
lp__	-571.38927111	0.0159109232	2.7174086	-577.5850852	-572.9795157	-571.06099836	-569.42673251	-567.0660533	29168.84	1.0000503

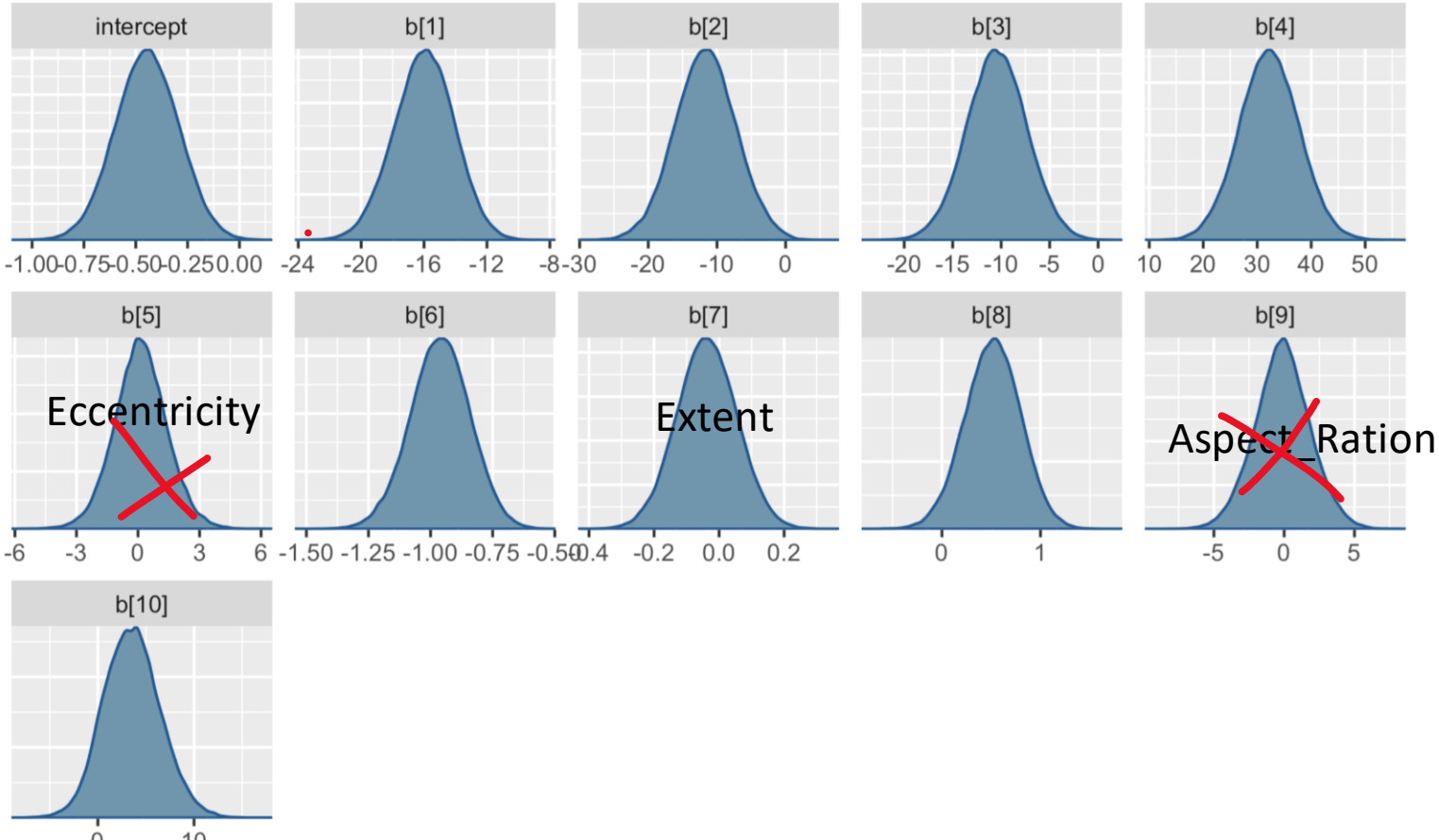
Model 2 (removed original b[2] and b[5])

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.45	0.00	0.15	-0.74	-0.55	-0.45	-0.34	-0.15	39776	1
b[1]	-16.00	0.01	1.91	-19.81	-17.27	-15.96	-14.69	-12.33	73697	1
b[2]	-11.66	0.03	4.45	-20.38	-14.66	-11.66	-8.68	-2.89	28602	1
b[3]	-10.45	0.02	3.12	-16.62	-12.54	-10.46	-8.35	-4.35	28689	1
b[4]	32.24	0.03	5.54	21.38	28.51	32.22	35.97	43.20	26398	1
b[5]	0.13	0.01	1.29	-2.42	-0.70	0.12	0.96	2.70	26884	1
b[6]	-0.96	0.00	0.12	-1.20	-1.04	-0.96	-0.88	-0.73	43275	1
b[7]	-0.04	0.00	0.09	-0.22	-0.10	-0.04	0.02	0.14	82998	1
b[8]	0.51	0.00	0.28	-0.05	0.33	0.52	0.71	1.06	64180	1
b[9]	-0.12	0.01	1.97	-4.03	-1.40	-0.12	1.16	3.81	44617	1
b[10]	3.54	0.02	2.95	-2.03	1.51	3.49	5.47	9.54	30529	1
lp__	-569.81	0.01	2.43	-575.43	-571.22	-569.48	-568.04	-566.08	33883	1

- Comparing the posterior means and 95% credible intervals of the two models, it can be concluded that they are very similar. An examination of the correlation between b[2], b[5], and other predictors reveals that they have a high correlation with predictors such as *Major_Axis_Length* and *Equiv_Diameter*. As a result, their influence on the model is minimal.

Model 3

- 3 chains, 6000 warm-ups, 36000 iters
- remove b[2](perimeter),b[5](Convex_Area), b[7](Eccentricity), and b[11](Aspect_Ration)
- intercept ~ normal(0, 10);
- b ~ double_exponential(0, 10)



- Since the posterior distributions of *Eccentricity* and *Aspect_Ration* in Model 2 are centered around 0 and they have high correlations with some of the remaining predictors, we can fit Model 3 with them removed. For b[7], because it has a low correlation with other predictors, we will keep it in the model and see what happens.

6,000

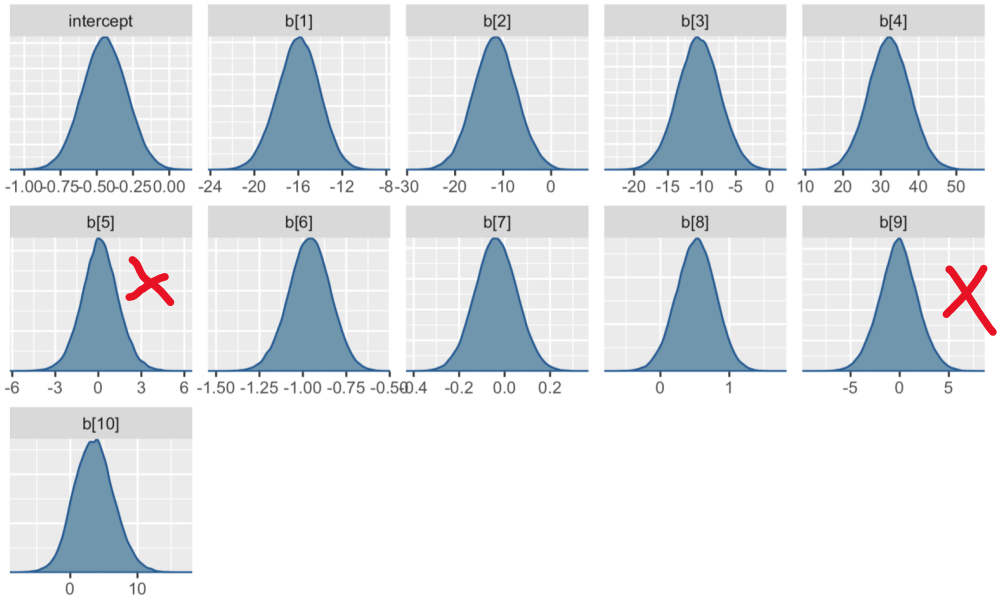
	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.45	0.00	0.12	-0.69	-0.53	-0.45	-0.37	-0.21	47736	1
b[1]	-15.98	0.01	1.90	-19.79	-17.25	-15.96	-14.69	-12.34	65367	1
b[2]	-12.31	0.01	2.30	-16.83	-13.84	-12.31	-10.80	-7.75	42484	1
b[3]	-10.74	0.01	1.97	-14.59	-12.06	-10.74	-9.41	-6.87	45007	1
b[4]	32.90	0.02	3.28	26.50	30.69	32.89	35.08	39.38	39145	1
b[5]	-0.97	0.00	0.11	-1.19	-1.04	-0.97	-0.89	-0.75	57256	1
b[6]	-0.04	0.00	0.09	-0.22	-0.10	-0.04	0.02	0.14	77906	1
b[7]	0.52	0.00	0.27	-0.03	0.33	0.52	0.70	1.05	68116	1
b[8]	3.34	0.01	1.54	0.36	2.29	3.32	4.38	6.40	53586	1
lp__	-568.64	0.01	2.16	-573.74	-569.87	-568.31	-567.05	-565.41	34730	1

10,000

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.45	0.00	0.12	-0.68	-0.53	-0.45	-0.37	-0.21	42684	1
b[1]	-15.96	0.01	1.90	-19.76	-17.23	-15.93	-14.68	-12.29	56804	1
b[2]	-12.30	0.01	2.27	-16.71	-13.81	-12.32	-10.80	-7.77	37479	1
b[3]	-10.72	0.01	1.97	-14.60	-12.05	-10.72	-9.40	-6.87	38686	1
b[4]	32.87	0.02	3.28	26.44	30.66	32.86	35.06	39.29	34599	1
b[5]	-0.97	0.00	0.11	-1.19	-1.04	-0.97	-0.89	-0.75	50097	1
b[6]	-0.04	0.00	0.09	-0.22	-0.10	-0.04	0.02	0.14	68535	1
b[7]	0.52	0.00	0.27	-0.02	0.34	0.52	0.70	1.05	59053	1
b[8]	3.34	0.01	1.53	0.41	2.29	3.32	4.36	6.38	45951	1
lp__	-568.62	0.01	2.17	-573.72	-569.84	-568.29	-567.04	-565.39	29418	1

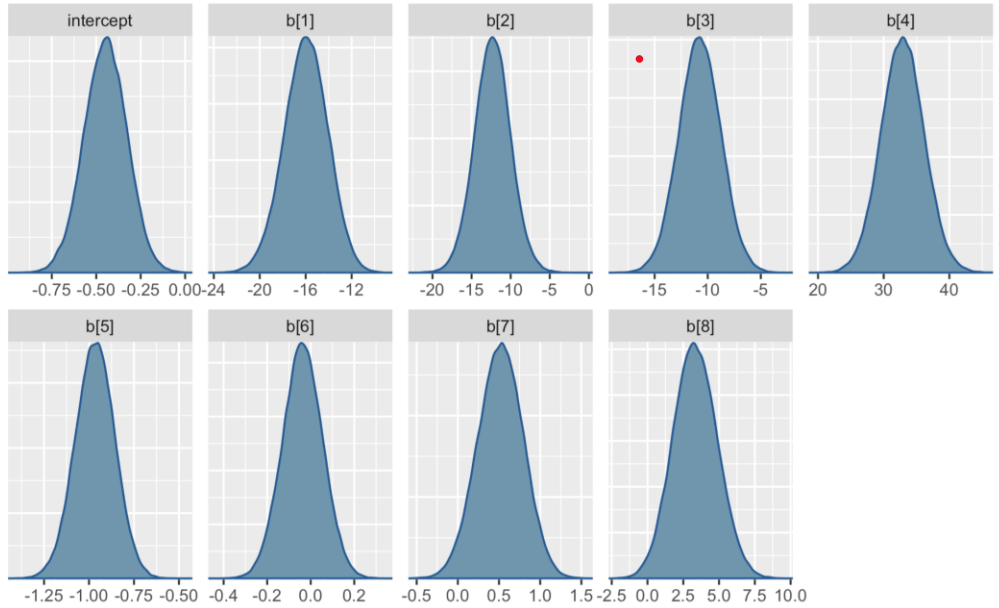
- Model convergence diagnostics show that 6,000 burn-in is not enough for b[2], so it is increased to 10,000. However, it turns out that the posterior estimates are similar.

Mode 2



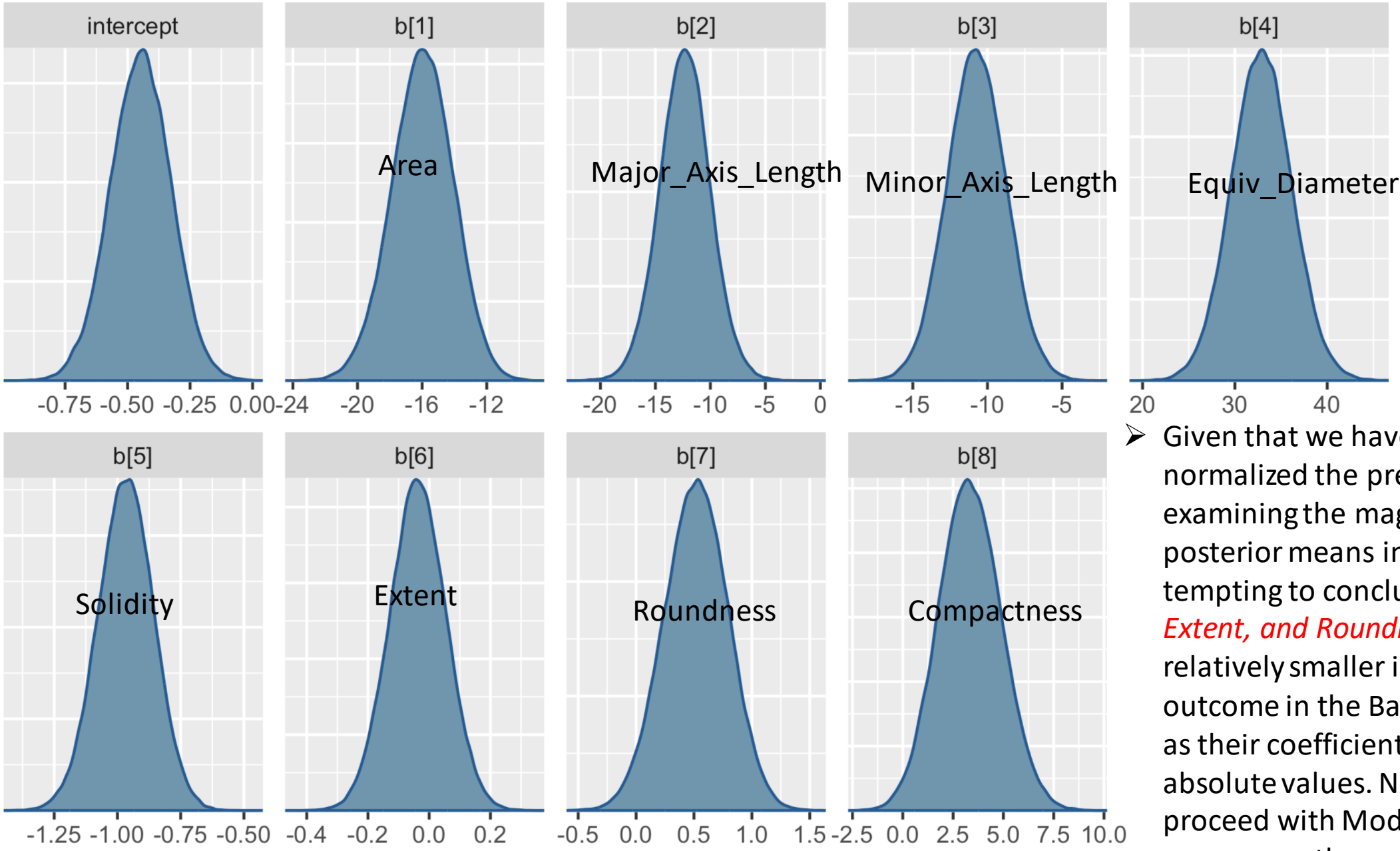
Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052

Model 3



Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052

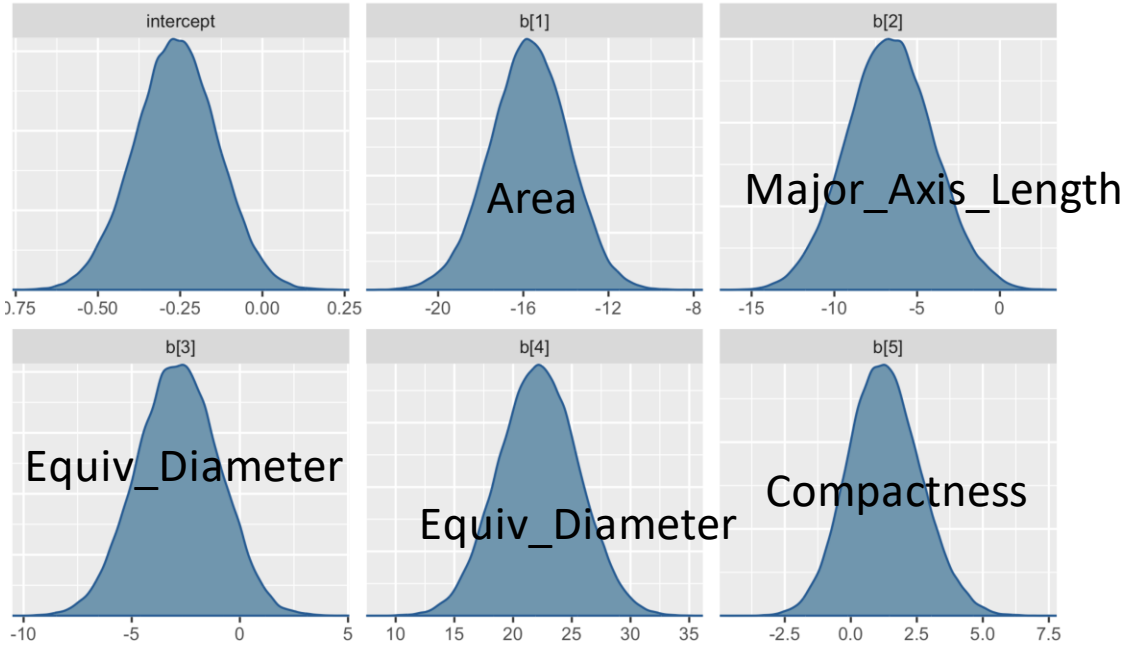
- The two models again have the same performance, meaning that removing the two predictors have no influence on the model performance.



- Given that we have already normalized the predictors, by examining the magnitudes of the posterior means in Model 3, it is tempting to conclude that *Solidity, Extent, and Roundness* have a relatively smaller influence on the outcome in the Bayesian model, as their coefficients have smaller absolute values. Next, we can proceed with Model 4, in which we remove these parameters, and assess the model's performance.

Model 4

- 3 chains, 10000 warm-ups, 36000 iters
- remove Perimeter, Convex_Area, Eccentricity ,Aspect_Ration, Solidity, Extent, and Roundness
- intercept ~ normal(0, 10);
- b ~ double_exponential(0, 10)



Model 3

```
Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052
```

Model 4

```
Accuracy: 0.854
Precision: 0.85
Recall: 0.8666667
Specificity: 0.8408163
F1-score: 0.8582524
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.26	0.00	0.13	-0.51	-0.35	-0.26	-0.18	-0.02	29060	1
b[1]	-15.72	0.01	1.83	-19.37	-16.94	-15.71	-14.48	-12.21	40990	1
b[2]	-6.62	0.02	2.61	-11.70	-8.40	-6.64	-4.86	-1.44	25846	1
b[3]	-2.85	0.01	1.89	-6.53	-4.14	-2.85	-1.57	0.81	29910	1
b[4]	22.18	0.02	3.38	15.62	19.88	22.17	24.46	28.80	25718	1
b[5]	1.28	0.01	1.37	-1.31	0.34	1.24	2.18	4.07	36995	1
lp__	-606.36	0.01	1.73	-610.60	-607.28	-606.03	-605.09	-603.99	26455	1

- Model 4 might be too simple, and thus results in a worse performance. As the posterior means of *Solidity* and *Roundness in* model3 are not zero, in Model 5, they will be put back to the model.

Model 5

- 3 chains, 10000 warm-ups, 36000 iters
- remove *Perimeter, Convex_Area, Eccentricity ,Aspect_Ration, Extent*
- intercept ~ normal(0, 10);
- b ~ double_exponential(0, 10)

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.44	0.00	0.12	-0.68	-0.52	-0.44	-0.36	-0.21	38407	1
b[1]	-15.95	0.01	1.89	-19.69	-17.22	-15.94	-14.67	-12.30	49821	1
b[2]	-12.24	0.01	2.27	-16.63	-13.76	-12.28	-10.73	-7.72	33558	1
b[3]	-10.68	0.01	1.95	-14.49	-11.99	-10.68	-9.36	-6.81	33793	1
b[4]	32.77	0.02	3.25	26.36	30.60	32.77	34.94	39.17	31176	1
b[5]	-0.97	0.00	0.11	-1.18	-1.04	-0.96	-0.89	-0.76	44437	1
b[6]	0.51	0.00	0.27	-0.03	0.33	0.51	0.69	1.04	51106	1
b[7]	3.33	0.01	1.54	0.35	2.29	3.31	4.35	6.40	38489	1
lp__	-568.19	0.01	2.04	-573.03	-569.33	-567.86	-566.69	-565.21	29922	1

Accuracy: 0.862
Precision: 0.8522727
Recall: 0.8823529
Specificity: 0.8408163
F1-score: 0.867052

- Model 5 has same performance as model 1,2 and 3. It's possible that the Bayesian models we've tried (Model 1, 2, 3, and 5) have reached a point where **adding or removing predictors doesn't significantly impact the performance**. It might indicate that the model has identified an optimal set of predictors that explain the outcome variable well. In this case, all four models seem to have settled on 7 predictors. Several factors might contribute to this result:
- ❑ The selected predictors are sufficient to explain the outcome variable, and any additional predictors don't add much more information.
- ❑ The models with more or fewer predictors might suffer from overfitting or underfitting, leading to a decrease in performance.
- ❑ It's possible that the choice of priors or the Bayesian modeling approach itself might be limiting the ability to identify a better model.

Model 6:

- Only use predictors selected by the Lasso model.
- Again use non-informative priors N(0,10) and Laplace(0,10)

Lasso

Area	.
Perimeter	.
Major_Axis_Length	-0.014201437
Minor_Axis_Length	.
Convex_Area	.
Equiv_Diameter	.
Eccentricity	.
Solidity	-0.058495715
Extent	.
Roundness	0.063653658
Aspect_Ration	-0.003888242
Compactness	0.276571762

Bayesian

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
intercept	-0.21	0.00	0.14	-0.49	-0.30	-0.21	-0.12	0.05	28566	1
b[1]	-0.11	0.00	0.11	-0.34	-0.19	-0.11	-0.04	0.11	46054	1
b[2]	-0.67	0.00	0.11	-0.89	-0.74	-0.66	-0.59	-0.47	38450	1
b[3]	0.73	0.00	0.26	0.21	0.55	0.73	0.90	1.24	39779	1
b[4]	-3.20	0.01	1.43	-6.06	-4.15	-3.17	-2.22	-0.44	26371	1
b[5]	-0.62	0.01	1.37	-3.35	-1.54	-0.60	0.30	2.05	26470	1
lp__	-626.80	0.01	1.73	-631.02	-627.72	-626.47	-625.53	-624.42	30115	1

Accuracy: 0.864
Precision: 0.8351254
Recall: 0.9137255
Specificity: 0.8122449
F1-score: 0.8726592

Accuracy: 0.866
Precision: 0.8533835
Recall: 0.8901961
Specificity: 0.8408163
F1-score: 0.8714012

- The performance metrics are quite close for both models. This indicates that both models are performing similarly well on the given data.
- **Bayesian** model **provides a full posterior distribution** for each parameter, giving a **measure of uncertainty** for each coefficient, but this information is absent in the LASSO logistic model.

Sensitivity Check

- To evaluate the impact of prior scale on the performance and posterior distribution of Model 6, we modified the scale of the priors of model 6 from 10 to 5 and 1, check the posterior distribution of the parameters and the model performance.

intercept	b[1]	b[2]	b[3]	b[4]	b[5]	lp__
-0.2125172	-0.1139148	-0.6685553	0.7269397	-3.1968180	-0.6224853	-626.8016015

Accuracy: 0.866
Precision: 0.8533835
Recall: 0.8901961
Specificity: 0.8408163
F1-score: 0.8714012

intercept	b[1]	b[2]	b[3]	b[4]	b[5]	lp__
-0.1914245	-0.1142988	-0.6664093	0.7172348	-2.9468682	-0.3792053	-627.3071825

Accuracy: 0.866
Precision: 0.8533835
Recall: 0.8901961
Specificity: 0.8408163
F1-score: 0.8714012

intercept	b[1]	b[2]	b[3]	b[4]	b[5]	lp__
-0.1189401	-0.1189363	-0.6524462	0.6701319	-2.0995783	0.4492133	-630.6630476

Accuracy: 0.862
Precision: 0.8549618
Recall: 0.8784314
Specificity: 0.844898
F1-score: 0.8665377

Sensitivity Check

- The intercept and coefficients (b[1] to b[5]) change with different prior scales. This indicates that the choice of prior scale does have an impact on the estimated parameters.
- Since the likelihood and model parameters are the same, higher lp__ values (less negative) indicate a better fit. Thus, a prior scale of 10 is the optimal. This implies that Model 6 has the highest ability to explain the observed data and is expected to make better predictions on unseen data compared to other models.
- It is proved by the performance put on the right hand side(previous slide: from top to bottom are 10 ,5, and 1.) .

Model Comparision

	Lasso	Bayesian with Lasso Predictors	Bayesian without any Informative
Intercetpt	0.52	-0.21	-0.44
Area			-15.95
Perimeter			
Major_Axis_Length	0.014	-0.11	-12.24
Minor_Axis_Length			-10.68
Convex_Area			
Equiv_Diameter			32.77
Eccentricity			
Solidity	-0.058	-0.67	-0.97
Extent			
Roundness	0.064	0.73	0.51
Aspect_Ration	-0.004	-3.2	
Compactness	0.277	-0.62	-3.33
lambda	0.01		

Performance Comparision

	Lasso	Bayesian with Lasso Predictors	Bayesian without any Informative
Accuracy	0.864	0.866	0.862
Precision	0.835	0.853	0.852
Recall	0.914	0.890	0.882
Specificity	0.812	0.840	0.841
F1-score	0.872	0.871	0.867

- Adjust the decision threshold of the Bayesian model to a little lower than 0.5 can increase its performance to as high as 0.874. This is not true for the Lasso model.

Conclusion

- Comparing the coefficients directly between Lasso logistic regression and Bayesian logistic regression models might not be meaningful, because the models have different assumptions, estimation methods, and regularization techniques.

- Lasso and Ridge logistic regression have higher computational efficiency for large datasets or high-dimensional problems.

- Bayesian models provide posterior distributions for the coefficients, which can be valuable when interpreting the results or making predictions with uncertainty bounds. When priors are available, Bayesian models can be more flexible and perform better.

- Using Lasso-selected predictors can provide a good starting point for the Bayesian model when no informative priors are available.

Limitations and Future Work

- Limitations: The limitations of our experiment include the absence of non-informative priors and sophisticated hyperpriors, potentially affecting Bayesian model performance. Due to computational power constraints of the personal computer used in this study, cross-validation techniques like WAIC or LOOIC were not employed for comparing models with different predictor combinations, and we were unable to build a Bayesian Ridge model.

- Future Work: Possible future work could involve exploring more advanced Bayesian models, incorporating informative priors(which is difficult for now) or hierarchical structures(requires higher computation power), or investigating other model comparison techniques to enhance the understanding of predictor importance and model performance.

- Summary: Despite the Bayesian models not outperforming the classical models in this case, the experiment still yielded valuable insights into the role of different predictors and the potential of Bayesian modeling in this context.

Reference

- Koklu, M., Sarigil, S. & Ozbek, O. The use of machine learning methods in classification of pumpkin seeds (*Cucurbita pepo* L.). *Genet Resour Crop Evol* **68**, 2713–2726 (2021). https://doi.org/10.1007/s10722-021-01226-0
- https://cran.r-project.org/web/packages/rstanarm/vignettes/binomial.html
- https://www.acsu.buffalo.edu/~adamcunn/probability/laplace.html
- https://cran.r-project.org/web/packages/rstan/vignettes/rstan.html
- University of California, Santa Cruz. (n.d.). Bayesian Statistics: MCMC Techniques and Applications [Online course]. Coursera. Retrieved from https://www.coursera.org/learn/mcmc-bayesian-statistics
- Markov Chain Monte Carlo in Python: *A Complete Real-World Implementation* https://towardsdatascience.com/markov-chain-monte-carlo-in-python-44f7e609be98
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Generalized linear models. In Bayesian Data Analysis (3rd ed., pp. [405-432]). CRC Press. http://www.stat.columbia.edu/~gelman/book/BDA3.pdf