Bayesian Logistic Regression for Diabetes Prediction

Team 26: Anitej Biradar, Arun Mishra, Fangru Linghu

Abstract

This project provides an in-depth exploration into Bayesian Logistic Regression, covering the statistical concepts behind Bayesian analysis, MCMC for sampling posterior distributions, and evaluating the effects of different prior distributions when training a model. The project also includes obtaining and cleaning a dataset, fitting a model using the stan_glm package, evaluating variable selection on improving a model's performance, and comparing its performance against a baseline model. Bayesian models were found to be flexible and easy to build, and effective for classification, achieving an accuracy of 80%. However, the study did not find significant differences in model performance with the use of different priors, possibly due to the dataset used. Future work could include analyzing these techniques on a different dataset and tuning hyperparameters for specific prior distributions. Overall, this project highlights the usefulness of Bayesian Logistic Regression in practical applications.

Materials and Methods

Bayesian Logistic Regression

Bayesian logistic regression is a statistical technique utilized to model binary outcome data. In this method, the dependent variable is dichotomous, taking on one of two possible values. Unlike traditional logistic regression, which employs maximum likelihood estimation (MLE) to determine model parameters, Bayesian logistic regression incorporates prior information about the model or its parameters into the analysis. This is achieved through the application of Bayes' theorem, which posits that the posterior distribution of the parameters is proportional to the product of the likelihood function and the prior density. The prior density reflects our a priori beliefs about the parameters before any data is observed.

Bayesian Logistic Regression

In Bayesian logistic regression, we use Bayes' theorem to update our beliefs about the model parameters after observing the data. This involves calculating the posterior distribution of the parameters given the data: P(b|y) = P(y|b) * P(b) / P(y).

- P(b|y): This represents the posterior distribution of the model parameters b given the observed data y. It reflects our updated beliefs about the parameters after observing the data.
- P(y|b): This is the likelihood function, which represents the probability of observing the data y given the model parameters b. It is calculated using the logistic regression model: p(y=1|x) = 1 / (1 + exp(-x^T * b)).
- P(b): This is the prior density of the model parameters b. It represents the beliefs about the parameters before observing any data. Common choices for the prior include normal or Laplace distributions.
- P(y): This is the marginal likelihood of the data y. It is calculated by integrating over all possible values of b: P(y) = integral(P(y|b) * P(b) db). However, this integral is often difficult or impossible to compute analytically.
- integral(P(y|b) * P(b) db): This represents the integral of the product of the likelihood function and the prior density over all possible values of b. It is used to calculate the marginal likelihood of the data.

MCMC

Markov Chain Monte Carlo (MCMC): This is a class of algorithms used to generate samples from a probability distribution. In Bayesian logistic regression, MCMC methods such as Metropolis-Hastings or Gibbs sampling can be used to generate samples from the posterior distribution of the model parameters. In the stan_glm function from rstanarm, which we are using in this project, the method used is HMC, Hamiltonian Monte Carlo. HMC is used to generate a posterior distribution through MCMC sampling based on a given prior distribution.

Dataset

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Variables

- Pregnancies
- Glucose
- Blood Pressure
- Thickness
- Insulin
- BMI
- DiabetesPedigreeFunction
- Age
- **Outcome**

Data Preparation

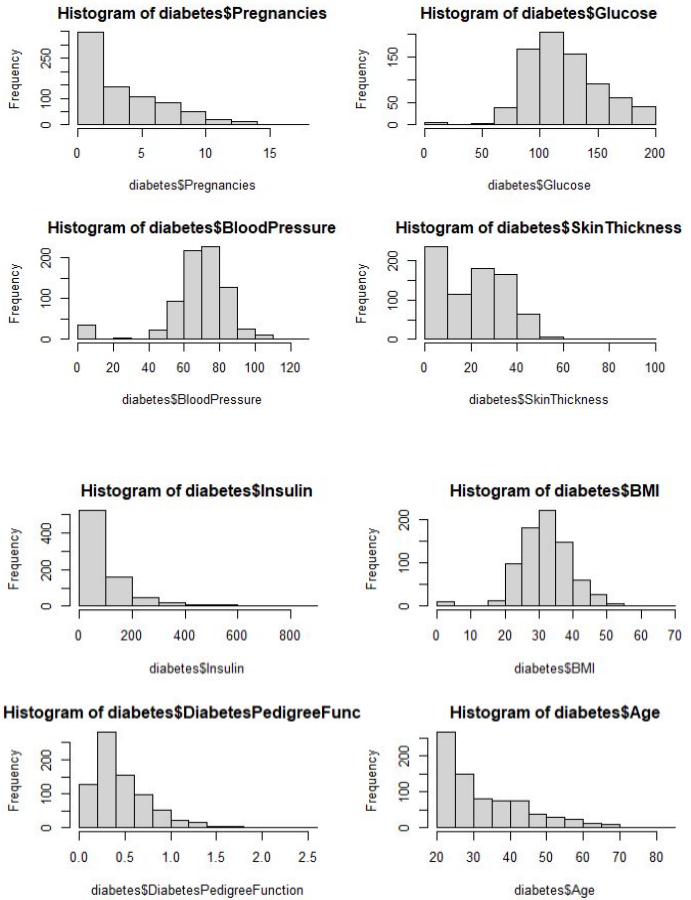
```
# Load dataset
diabetes <- read.csv("diabetes.csv")
head(diabetes, 5)
summary(diabetes)
dim(diabetes)

# Check for missing values
sum(is.na(diabetes))

# Check for duplicates
diabetes[duplicated(diabetes), ]
```

The dataset contains 768 records and 9 columns. There were no missing values or duplicates in the dataset. The summary command helped to get an understanding of the distributions, and we saw that many fields have a value of 0 as the minimum, when this may not be possible (for example, BloodPressure cannot be 0). This helps us to understand that we need to clean some records..

Data Preparation

```
# Plot the distributions of each column
par(mfrow = c(2, 2))
hist(diabetes$Pregnancies)
hist(diabetes$Glucose)
hist(diabetes$BloodPressure)
hist(diabetes$SkinThickness)
hist(diabetes$Insulin)
hist(diabetes$BMI)
hist(diabetes$DiabetesPedigreeFunction)
hist(diabetes$Age)
```

Histograms help us to further visualize the distributions of each variable in the dataset. It also helps to easily see which variables may contain outliers. Besides 0, we don't see any obvious outliers in the dataset.



Data Preparation

```
# Removing observation rows with 0 in any of the variables
for (i in 2:6) {
   diabetes <- diabetes[-which(diabetes[, i] == 0), ]
}

# Checking how many records are in the cleaned dataset
table(diabetes$Outcome)

# Normalizing mean/standard deviation for each predictor
for (i in 1:8) {
   diabetes[i] <- scale(diabetes[i])
}

# Converting Outcome from numerical to categorical
diabetes$Outcome <- factor(diabetes$Outcome)
```

Since Glucose, BloodPressure, SkinThickness, Insulin, BMI, and DiabetesPedigreeFunction cannot take values of zero, we assume that there must have been an error in the dataset, and remove those records. This leaves us with **392 records**, of which 262 are Outcome 0, and 130 are Outcome 1.

Additionally, we normalize the predictor variables, and convert the Outcome field into a categorical variable from numerical.

Bayesian Logistic Regression

```
# Weak priors
normal_prior <- stan_glm(Outcome ~ ., data = diabetes, family = binomial(link = "logit"),
            prior = normal(0, 10), prior_intercept = normal(0, 10), QR = TRUE, seed = 1089)
uniform_prior <- stan_glm(Outcome ~ ., data = diabetes, family = binomial(link = "logit"),
            prior = NULL, prior_intercept = NULL, QR = TRUE, seed = 1089)

# Stronger priors
cauchy_prior <- stan_glm(Outcome ~ ., data = diabetes, family = binomial(link = "logit"),
            prior = cauchy(0, 2.5), prior_intercept = cauchy(0, 2.5), QR = TRUE, seed = 1089)
student_t_prior <- stan_glm(Outcome ~ ., data = diabetes, family = binomial(link = "logit"),
            prior = student_t(df = 7, location = 0, scale = 2.5), prior_intercept = student_t(df = 7, location = 0, scale = 2.5), QR = TRUE)
```

To evaluate the effect of choosing different types of priors on the Logistic Regression model, we train 4 separate models, with a different prior (normal, uniform, cauchy, and student_t). stan_glm uses MCMC sampling to generate a posterior distribution. The default parameters for stan_glm include running 4 chains with 2000 iterations for sampling. For these models, we are using the formula (Outcome ~ .), meaning Outcome against all predictor variables.

Results & Discussion

Posterior Predictive Checks

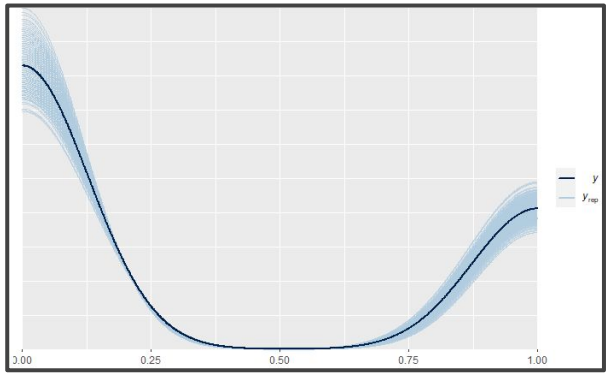
```
pp_check(normal_prior, nreps=500)
pp_check(cauchy_prior, nreps=500)
pp_check(student_t_prior, nreps=500)
pp_check(uniform_prior, nreps=500)
```

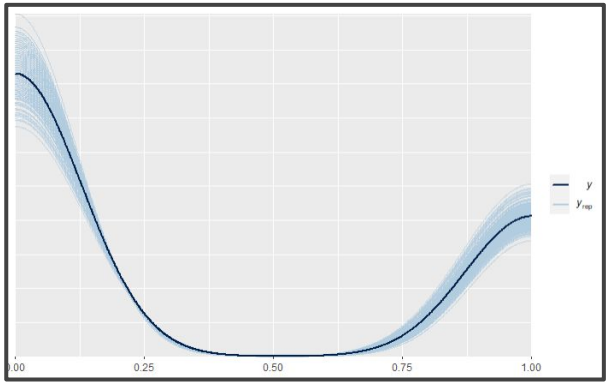
We can use the pp_check function to generate Posterior Predictive Checks (PPCs). This will generate a plot showing the distribution of observed data (black line) compared to the distribution of data simulated from the posterior predictive distribution (colored lines)*. The goal of a PPC is to assess whether the model adequately captures the patterns in the observed data. If the observed data falls within the range of the simulated data, then this suggests that the model is a good fit to the data. We can see that all of the models appear to be good fits for the data.

*see next slide for plots

Posterior Predictive Checks

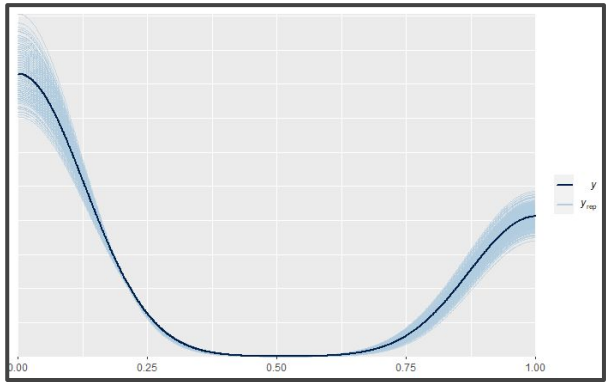
Normal

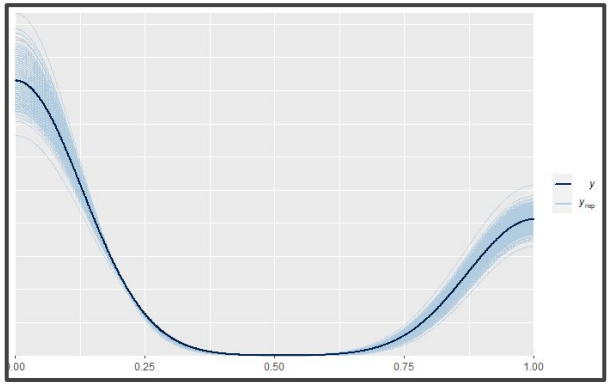




Uniform

Cauchy





Student T

Posterior Distributions

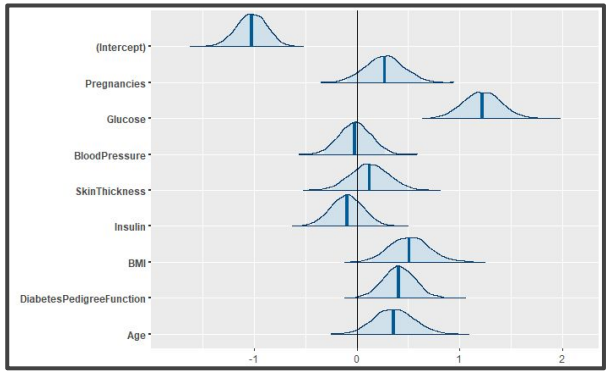
```
normal_pplot <- plot(normal_prior, "areas", prob = 0.95, prob_outer =
1)
normal_pplot +
  geom_vline(xintercept = 0)

uniform_pplot <- plot(uniform_prior, "areas", prob = 0.95, prob_outer =
1)
uniform_pplot +
  geom_vline(xintercept = 0)

cauchy_pplot <- plot(cauchy_prior, "areas", prob = 0.95, prob_outer =
1)
cauchy_pplot +
  geom_vline(xintercept = 0)

student_t_pplot <- plot(student_t_prior, "areas", prob = 0.95,
prob_outer = 1)
student_t_pplot +
  geom_vline(xintercept = 0)
```

We use the plot() function along with the “areas” parameter to display density plots of the posterior distributions of all 4 models, with a 95% central probability interval. From visual inspection*, while all 4 models appear to be reasonably good estimates of the posterior, there is not much difference between them to indicate whether a specific prior is “better” (they are all very similar). We also note that the variables Glucose, BMI, DiabetesPedigreeFunction, and Age are distributed positively (the furthest away from zero), indicating that these variables have the most influence on the Outcome in the model.

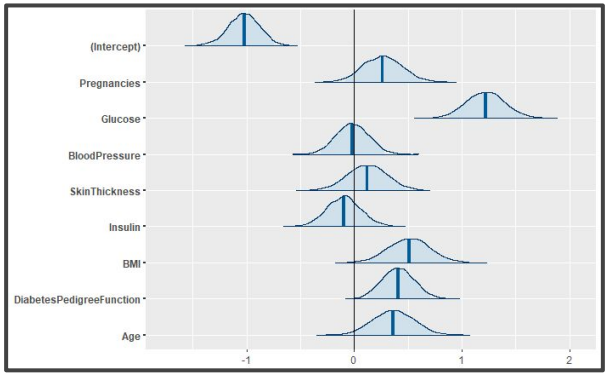
*see next slide for plots

Posterior Distributions

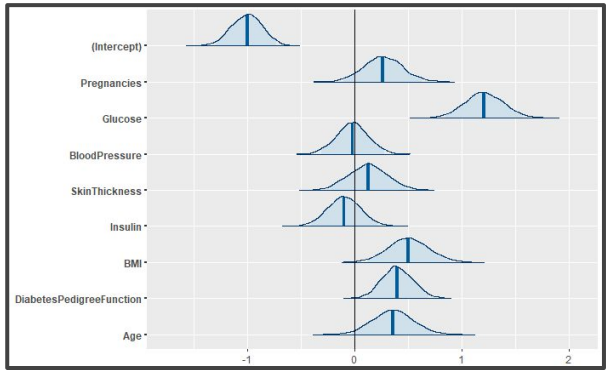
Normal



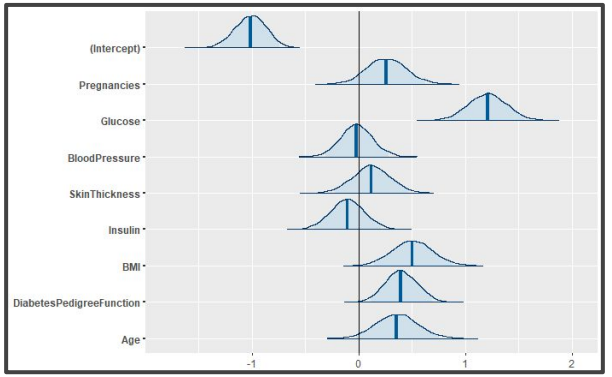
Uniform



Cauchy



Student T



Posterior Intervals

```r
round(posterior_interval(normal_prior, prob = 0.9), 2)
round(posterior_interval(uniform_prior, prob = 0.9), 2)
round(posterior_interval(cauchy_prior, prob = 0.9), 2)
round(posterior_interval(student_t_prior, prob = 0.9), 2)
```

```
                            5%   95%
(Intercept)              -1.27 -0.79
Pregnancies              -0.04  0.57
Glucose                   0.92  1.52
BloodPressure            -0.27  0.24
SkinThickness            -0.18  0.43
Insulin                  -0.36  0.16
BMI                       0.20  0.83
DiabetesPedigreeFunction  0.16  0.66
Age                       0.06  0.68
                            5%   95%
(Intercept)              -1.28 -0.79
Pregnancies              -0.02  0.56
Glucose                   0.93  1.52
BloodPressure            -0.26  0.23
SkinThickness            -0.18  0.42
Insulin                  -0.36  0.17
BMI                       0.19  0.83
DiabetesPedigreeFunction  0.16  0.66
Age                       0.05  0.67
                            5%   95%
(Intercept)              -1.24 -0.78
Pregnancies              -0.04  0.57
Glucose                   0.91  1.51
BloodPressure            -0.27  0.22
SkinThickness            -0.17  0.42
Insulin                  -0.36  0.16
BMI                       0.18  0.82
DiabetesPedigreeFunction  0.17  0.64
Age                       0.04  0.67
                            5%   95%
(Intercept)              -1.26 -0.78
Pregnancies              -0.02  0.57
Glucose                   0.92  1.50
BloodPressure            -0.26  0.23
SkinThickness            -0.17  0.41
Insulin                  -0.36  0.16
BMI                       0.19  0.82
DiabetesPedigreeFunction  0.16  0.65
Age                       0.04  0.68
```

The posterior_interval() function in R is used to calculate credible intervals for a given posterior distribution. Credible intervals estimate the range of plausible values for a parameter based on a posterior distribution.

With a 90% credible interval, we can see that the values for all 4 models are quite similar, but only the Glucose, BMI, DiabetesPedigreeFunction, and Age predictors have an entirely positive range. This indicates that the other variables might have 0 as their coefficient, meaning that they may have no effect on the outcome variable. We saw similar results in the Posterior Distributions. This is noted for when we compare against baseline and subset models.

Cross Validation (LOOCV)

```r
loo1 <- loo(normal_prior, save_psis = TRUE)
loo2 <- loo(uniform_prior, save_psis = TRUE)
loo3 <- loo(cauchy_prior, save_psis = TRUE)
loo4 <- loo(student_t_prior, save_psis = TRUE)
loo_compare(loo1,loo2,loo3,loo4)
```

```
                elpd_diff se_diff
student_t_prior  0.0       0.0
cauchy_prior     0.0       0.1
uniform_prior    0.0       0.2
normal_prior    -0.3       0.2
```

Another evaluation technique is LOOCV (Leave One Out Cross Validation). stan_glm works very well with a package called loo, which performs this in R.

The loo function computes the LOO-CV estimate of the log-likelihood of a model, and we can use the loo_compare function to compute the log_likelihood of multiple models.

elpd_diff stands for "Expected Log Pointwise Predictive Density difference" and represents the difference between the LOO cross-validation scores of two models. A positive value of elpd_diff indicates that the first model has a higher cross-validation score (i.e., is more accurate) than the second model.

se_diff stands for "standard error of the difference" and is an estimate of the uncertainty associated with the elpd_diff value. A larger value of se_diff indicates that the difference in LOO scores between the two models is less certain.

Both of these statistics are used to evaluate and compare the performance of different Bayesian models, with smaller values of elpd_diff and larger values of se_diff indicating that the models have similar performance, while larger values of elpd_diff and smaller values of se_diff indicate that one model is clearly better than the other.

From these results, we can see that the student_t model has the best score, but only by a very small margin. We will use this model going forward.

Baseline and Subset Models

```
# Baseline and Improved (Subset) Models

```{r}
baseline <- update(student_t_prior, formula = Outcome ~ 1, QR = FALSE, refresh=0)
loo_baseline <- loo(baseline)
loo_compare(loo_baseline, loo4)
```

                  elpd_diff se_diff
 student_t_prior   0.0   0.0
 baseline   -67.7   11.5

```{r}
subset <- update(student_t_prior, formula = Outcome ~ BMI + Glucose + Age + DiabetesPedigreeFunction, QR = FALSE, refresh=0)
loo_subset <- loo(subset)
loo_compare(loo_subset, loo4)
```

                  elpd_diff se_diff
 subset   0.0   0.0
 student_t_prior -3.0   2.1
```

Based on our evaluation so far, we have seen that the four models trained on different priors have relatively similar performance, and they appear to perform well with the dataset. To evaluate our formula, we trained two new models, one baseline (no predictor variables, only the intercept) and one subset (using only the 4 variables that stood out in previous evaluations). We found that the student_t model performed significantly better than the baseline model, indicating that the predictor variables helped in improving the performance of the model. Simultaneously, we see that the subset model actually outperforms the student_t model, indicating that removing predictors that may have a lesser impact on the model will improve performance. This was calculated using the same loo_compare library that we used in the previous slide.

Confusion Matrix and Posterior Classification Accuracy

As our final evaluation method, we computed posterior predictive probabilities and used them to compute classification errors. For each of our 4 models, plus the baseline and subset models, we calculated the confusion matrix, as well as the posterior classification accuracy and posterior balanced classification accuracy (taking into account the difference of size in classes). See next 2 slides for the code/outputs.

We see that our subset model performs the best, with an accuracy of 80% (75% balanced). The 4 original models perform exactly the same (with minor differences that are lost in rounding), with an accuracy of 78% (73% balanced). Lastly, our baseline (only intercept) model performed the worst, with an accuracy of 67% (50% balanced). This is a good way to conclude our analysis of Bayesian logistic regression, and show that it is an accurate model for data classification in our use case, that performs well in multiple scenarios.

Confusion Matrix and Posterior Classification Accuracy

```
# predicted probability
normal_pr <- as.integer(colMeans(posterior_epred(normal_prior)) >= 0.5)
# confusion matrix
confusionMatrix(as.factor(as.numeric(normal_pr > 0.5)), y)[2]
# posterior classification accuracy
round(mean(xor(normal_pr,as.integer(y==0))),2)
# posterior balanced classification accuracy
round((mean(xor(normal_pr[y==0]>0.5,as.integer(y[y==0])))+mean(xor(normal_pr[y==1]<0.5,as.integer(y[y==1]))))/2,2)
```

```
$table
          Reference
Prediction   0   1
         0 233  56
         1  29  74

[1] 0.78
[1] 0.73
```

Example code for calculating predicted probability, confusion matrix, posterior classification accuracy, and posterior balanced classification accuracy. The same template is followed for all six models.

Confusion Matrix and Posterior Classification Accuracy

```
$table
          Reference
Prediction   0   1
         0 233  56
         1  29  74

[1] 0.78
[1] 0.73
```

Normal

```
$table
          Reference
Prediction   0   1
         0 233  56
         1  29  74

[1] 0.78
[1] 0.73
```

Cauchy

```
$table
          Reference
Prediction   0   1
         0 262 130
         1   0   0

[1] 0.67
[1] 0.5
```

Baseline

```
$table
          Reference
Prediction   0   1
         0 233  56
         1  29  74

[1] 0.78
[1] 0.73
```

Uniform

```
$table
          Reference
Prediction   0   1
         0 233  56
         1  29  74

[1] 0.78
[1] 0.73
```

Student T

```
$table
          Reference
Prediction   0   1
         0 233  51
         1  29  79

[1] 0.8
[1] 0.75
```

Subset

Discussion

Overall, this project provides an in-depth exploration into Bayesian Logistic Regression. Specifically, we covered the statistical concepts behind Bayesian analysis, MCMC for sampling posterior distributions; obtaining and cleaning a dataset relating to diabetes; fitting a model using the stan_glm package; evaluating the effects of different prior distributions when training a model using methods such as posterior predictive checks, posterior distributions/intervals, LOOCV, and confusion matrix/posterior classification accuracy; evaluating the effects of variable selection on improving a model's performance, as well as its performance against a baseline model. We have found that Bayesian models are flexible and easy to build, and are effective for classification (best performance of 80% accuracy).

While we did not see much significant differences with the use of different priors for the models, that could be a result of the dataset we used. Future work could include analyzing these techniques on a different dataset, and also tuning hyperparameters for specific prior distributions (e.g. trying different degrees of freedom if using a student T distribution).

References

Literature Cited

1. Vehtari, A., Gelman, A., & Gabry, J. (2017). Bayesian logistic regression with rstanarm. Kaggle. Retrieved from https://www.kaggle.com/code/avehtari/bayesian-logistic-regression-with-rstanarm/notebook
2. Zhang, L. (n.d.). Statistical Learning with R. Kaggle. Retrieved from https://www.kaggle.com/code/laozhang/statistical-learning-with-r?scriptVersionId=445129
3. UCI Machine Learning Repository. (n.d.). Pima Indians Diabetes Database. Retrieved from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
4. RStanArm Development Team. (2021). rstanarm: Bayesian Applied Regression Modeling via Stan. Retrieved from https://mc-stan.org/rstanarm/
5. Goodrich, B., Gabry, J., Ali, I., & Brilleman, S. (2021). rstanarm: Bayesian applied regression modeling via Stan (Version 2.26.1) [Computer software manual]. Retrieved from https://cran.r-project.org/web/packages/rstanarm/vignettes/rstanarm.html
6. Gabry, J., & Mahr, T. (2021). Bayesplot: Plotting for Bayesian models (Version 1.8.1) [Computer software manual]. Retrieved from https://mc-stan.org/bayesplot/
7. Vehtari, A., Gelman, A., & Gabry, J. (2021). Glossary of Terms in the Bayesian Workflow. LOO: Leave-One-Out Cross-Validation and Related Estimators. Retrieved from https://mc-stan.org/loo/reference/loo-glossary.html
8. Stan Development Team. (2021). pp_check.stanreg: Posterior predictive checks for Bayesian regression models. Retrieved from https://mc-stan.org/rstanarm/reference/pp_check.stanreg.html