Bayesian Data Analysis Project Report

Anonymous

May 2024

1 Introduction

Credit score prediction is a crucial aspect of financial analysis, assisting in risk assessment, loan approval, and financial decision-making processes. In this project, we used techniques of Bayesian data analysis to predict credit scores using a dataset containing information from 1000 customers. Key features such as income, savings, debt, and other relevant financial indicators are leveraged to build predictive models. Moreover, we separated customers based on their gambling habits into three distinct groups: non-gamblers, low-gamblers, and high-gamblers. By changing the dataset in this manner, we aim to test differences in credit scores among these groups, shedding light on the potential impact of gambling behavior on individuals' financial health.

In our project, we utilized separated and hierarchical models to capture the intricate relationships between customer attributes and creditworthiness across different gambling habits. To ensure the reliability of our analyses, we evaluated model convergence through diagnostic tools such $\hat{R}$ and divergence analysis. Furthermore, we conducted posterior predictive checks to validate the adequacy of our models in capturing the underlying data distribution. Sensitivity analyses were also performed to explore the impact of different prior specifications on model outcomes, providing valuable insights into the robustness of our findings and the influence of prior beliefs on posterior inferences. By systematically comparing the performance of various models and priors, we aimed to provide a comprehensive evaluation framework for credit score prediction, enhancing the reliability and interpretability of our results.

2 Data

The dataset utilized in our Bayesian data analysis project was extracted from Kaggle public datasets [1]. Comprising information from 1000 customers, the dataset contains an array of features crucial for credit score prediction and financial analysis. Among these features are fundamental indicators such as credit score, income, savings, and debt, providing essential insights into individuals' financial profiles. Additionally, the dataset includes categorical variables, including information on customers' gambling habits and mortgage status. These categorical variables offer valuable contextual information, allowing for the exploration of how specific behaviors and financial decisions impact creditworthiness.

The features include:

- CREDIT_SCORE: Numerical target variable representing the customer's credit score.
- INCOME: Total income in the last 12 months
- SAVINGS: Total savings in the last 12 months
- DEBT: Total existing debt

[1] `https://www.kaggle.com/datasets/conorsully1/credit-score/data`

- T_EXPENDITURE_6: Total expenditure in the last 6 months
- T_EXPENDITURE_12: Total expenditure in the last 12 months
- CAT_GAMBLING: Gambling category (none, low, high)
- CAT_DEBT: 1 if the customer has debt; 0 otherwise
- CAT_CREDIT_CARD: 1 if the customer has a credit card; 0 otherwise
- CAT_MORTGAGE: 1 if the customer has a mortgage; 0 otherwise
- CAT_SAVINGS_ACCOUNT: 1 if the customer has a savings account; 0 otherwise
- CAT_DEPENDENTS: 1 if the customer has any dependents; 0 otherwise

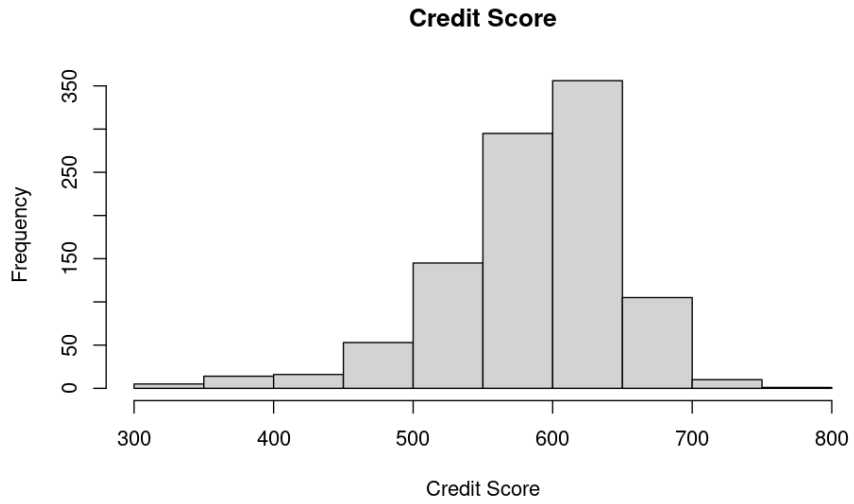

Figure 1: Histogram of credit scores.

2.1 Data preprocessing and analysis

To preprocess the data, we performed an initial cleaning process aimed at enhancing its clarity and reducing dimensionality to enable effective pattern recognition by the models. This involved the removal of columns containing minor or redundant information, such as customer IDs. By pruning irrelevant features, we streamlined the dataset for improved model performance. Additionally, we generated visualizations to gain insights into the distribution and relationships within the data. For example, Figure 2 presents histograms illustrating the distributions of income, savings, and debt. Furthermore, Figure 4 showcases a scatter plot illustrating the relationship between debt and income. These visualizations provide valuable context for understanding the underlying patterns and dynamics present in the dataset.

We segmented the data into three distinct groups based on the gambling column: non-gambling, low-gambling, and high-gambling. Subsequently, we employed one-hot encoding to convert categorical variables into binary vectors and normalized all columns to ensure uniform scales and prepare the data for model utilization. This preprocessing step enabled us to effectively incorporate these features into our models while preserving their inherent characteristics.

```
# Standardizing columns of data except last column (credit score)
data_normalized <- data
data_normalized[1:(length(data)-1)] <- as.data.frame(scale(data[1:(length(data)-1)]))

# Separating data into three parts for each CAT_GAMBLING value
```

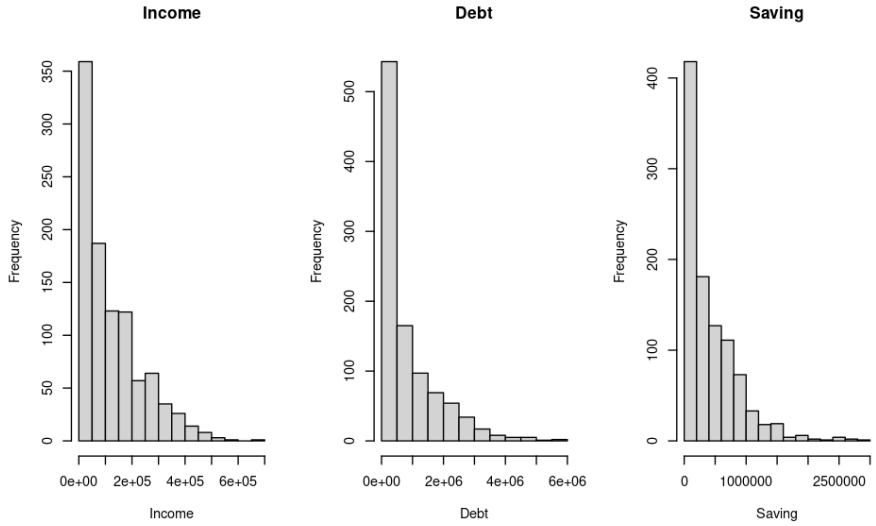

Figure 2: Histogram income, savings, and debt.

```
data_0 <- data_normalized[data$CAT_GAMBLING == 0,]
data_1 <- data_normalized[data$CAT_GAMBLING == 1,]
data_2 <- data_normalized[data$CAT_GAMBLING == 2,]
```

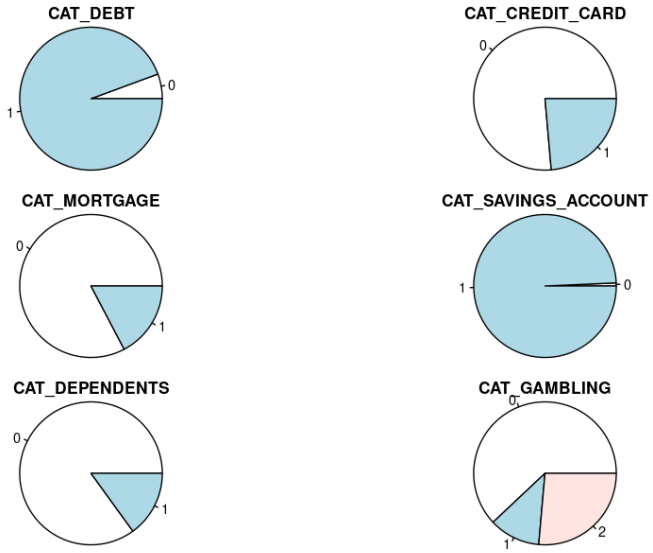

Figure 3: Pie chart of categorical variables.

3 Methods

In our analysis, we explored the effectiveness of two distinct modeling approaches: a separate model and a hierarchical model, each offering unique perspectives on the dataset. The separate model operates by independently learning distributions for each group—non-gambling, low-gambling, and high-gambling—allowing for insights into the specific characteristics of each subgroup. Conversely, the hierarchical model adopts a more comprehensive viewpoint by simultaneously examining individual groups alongside the entire population. This model assumes distinct prior distributions for each subgroup, yet

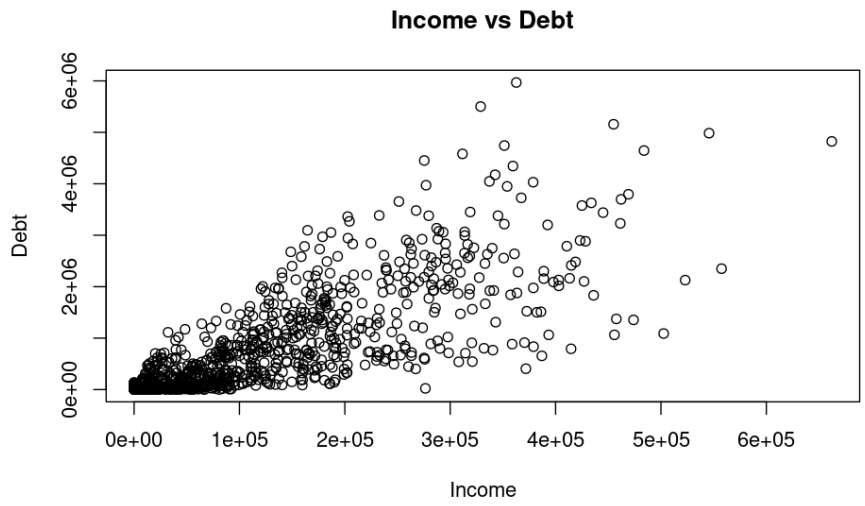

Figure 4: Scatter plot of debt vs income.

crucially, the parameters of these distributions are pulled from a single overarching distribution. By doing so, the hierarchical model not only captures the nuanced differences within each group but also considers the collective behavior of the entire population. This holistic approach enhances the model's ability to discern underlying trends and patterns while accommodating the diverse range of behaviors exhibited across different gambling habits.

In both our models, the underlying idea is a simple linear regression in the form of:

$$y \sim \mathcal{N}(\alpha + \beta x, \sigma)$$

The separate and hierarchical models will set different implementations of priors on variables α, β, and σ.

In our modeling, we used weakly informative priors to reduce unnecessary influence on the posterior distribution while still providing a meaningful scale for the model. By incorporating these priors, we aimed to strike a balance between accounting for relevant prior knowledge and allowing the data to predominantly guide the inference process. This strategy ensured that the models remained robust and unbiased, with the priors serving primarily to establish a sensible scale rather than imposing strong constraints on the posterior estimates. In doing so, we preserved the flexibility of the models to adapt to the intricacies of the dataset while effectively capturing the underlying patterns and relationships.

3.1 Separate

The separate model uses distinct priors for each group, enabling the independent learning of posterior distributions for each subgroup. When selecting priors for α, we chose values proportional to the scale of credit scores, striking a balance between informativeness and objectivity. As for β and σ, we employed weakly informative normal priors to maintain flexibility in the models.

The priors for the separate models are as follows:

$$\alpha \sim \mathcal{N}(1000, 100)$$
$$\beta \sim \mathcal{N}(0, 10)$$
$$\sigma \sim \mathcal{N}(0, 10)$$

3.2 Hierarchical

In hierarchical models, instead of having a prior for the α and β parameters of different groups, we use μ_0 and σ_0 with their own priors and use these values as the mean and std to sample α and β. In our hierarchical models, we used the separate model α and β priors for μ and σ parameters. The model priors and parameter declartion is as follows:

$$
\begin{aligned}
\mu_\alpha &\sim \mathcal{N}(1000, 100) \\
\sigma_\alpha &\sim \mathcal{N}(0, 30) \\
\mu_\beta &\sim \mathcal{N}(0, 100) \\
\sigma_\beta &\sim \mathcal{N}(0, 10) \\
\alpha &\sim \mathcal{N}(\mu_\alpha, \sigma_\alpha) \\
\beta &\sim \mathcal{N}(\mu_\beta, \sigma_\beta) \\
\sigma &\sim \mathcal{N}(0, 10)
\end{aligned}
$$

3.3 Stan

Complete Stan codes for the models are given in Appendix to improve the readability of the report. Here we can see the commands used for running the Stan codes. We mostly used default Stan arguments (4 chains, 2000 iterations).

```
stan_data <- list(
                  N1=nrow(data_0),
                  N2=nrow(data_1),
                  N3=nrow(data_2),
                  N_features=n_features,
                  x1=data_0[1:n_features],
                  x2=data_1[1:n_features],
                  x3=data_2[1:n_features],
                  y1=data_0[['CREDIT_SCORE']],
                  y2=data_1[['CREDIT_SCORE']],
                  y3=data_2[['CREDIT_SCORE']]
                  )

separate_model = cmdstan_model("./Separate.stan")
out <- capture.output(
  separate_fit <- separate_model$sample(data=stan_data, refresh=0, show_messages=FALSE)
)
separate_summary = separate_fit$summary()

hierarchical_model = cmdstan_model("./Hierarchical.stan")
out <- capture.output(
  hier_fit <- hierarchical_model$sample(data=stan_data, refresh=0, show_messages=FALSE)
)

hier_summary = hier_fit$summary()
```

3.4 Convergence diagnostic

To evaluate model convergence, several diagnostic measures are employed, including $\hat{R}$, effective sample size, and identification of divergent transitions. The potential scale reduction factor, $\hat{R}$, serves as an indicator of chain mixing and convergence; larger values suggest inadequate mixing. In our analysis, we check that $\hat{R}$ values are below 1.01, indicating satisfactory convergence.

```
sum(separate_summary$rhat>1.01)
## [1] 0
sum(hier_summary$rhat>1.01)
## [1] 0
```

For a more complete diagnosis of all convergence attributes like $\hat{R}$, ESS, transitions divergent, and transitions treedepth, we can use 'cmdst_diagnose' command:

```
separate_fit$cmdstan_diagnose()
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete, no problems detected.
```

```
hier_fit$cmdstan_diagnose()
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## 331 of 4000 (8.28%) transitions ended with a divergence.
## These divergent transitions indicate that HMC is not fully able to
## explore the posterior distribution.
## Try increasing adapt delta closer to 1.
## If this doesnt remove all divergences, try to reparameterize the ## model.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete.
```

We can see that the hierarchical model has some issues in transition divergence section and about 8% of transitions have ended with a divergence.

During our initial attempts, the $\hat{R}$ values and divergence indicators showed concerning trends, with none of the models converging to satisfactory states despite various prior choices. However, a key breakthrough emerged when we normalized all columns to have zero mean and unit standard deviation. This adjustment proved essential, considering the difference in scales of the columns was problematic for the models to distinguish feature importance accurately. Consequently, the reported results relate to models trained on normalized data columns, a crucial step that enabled the models to better adapt to the dataset's characteristics.

3.5 Posterior predictive checks

For checking posterior predictions, we selected 100 random sampels from each model and group and plotted them against the true distribution in Figure 5.

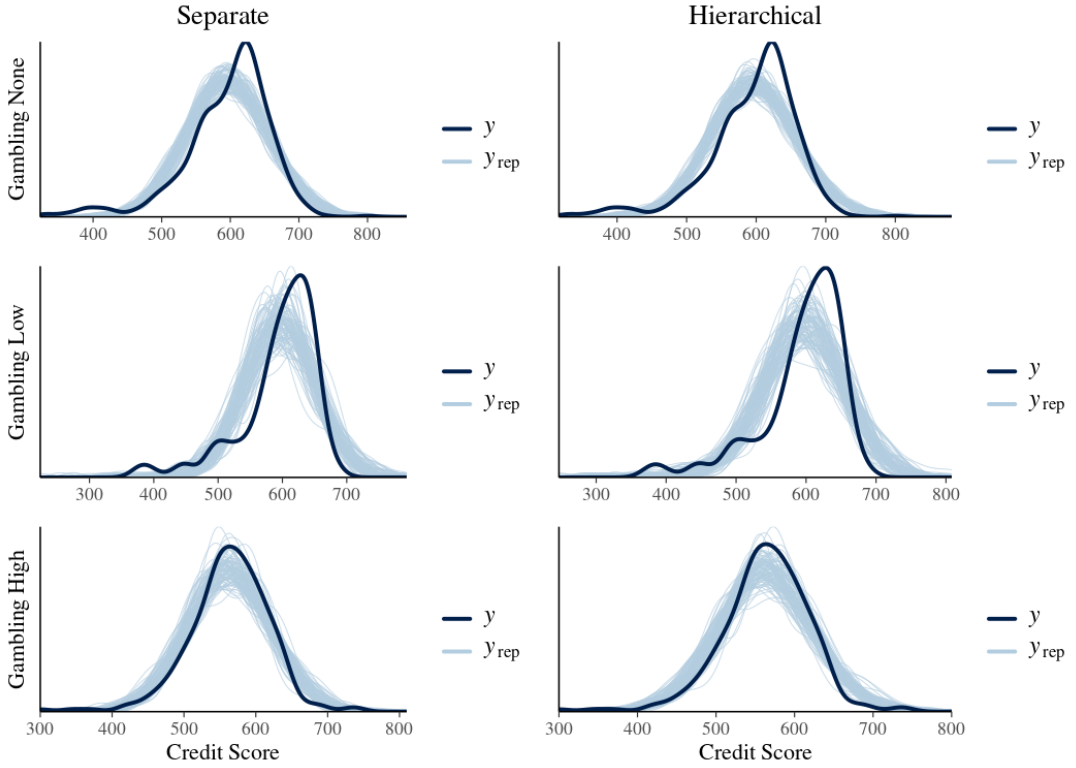

Figure 5: Predictive posterior sample vs true distribution.

Observing the outcomes of both the separate and hierarchical models reveals striking similarity, with predictions closely aligned with the true distribution across both methods. Notably, the high-gambling group shows the most accurate predictions, primarily attributable to the intrinsic simplicity of its true distributions. However, challenges appear with the predictions for the other two groups, as evidenced by slight shifts in mean within their distributions. There are some irregularities present in their true distributions which makes it difficult for the models to learn underlying patterns effectively. Overall, the predicted posterior distributions seem satisfactory with respect to the true distribution of data.

3.6 Sensitivity analysis

To assess the sensitivity of the models to priors, we employed narrower versions of our original priors by reducing the standard deviation for both the separate and hierarchical models on α and β. The modified priors, outlined below, were applied to these variables, while those not explicitly mentioned remained unchanged from the base model:

- **Separate narrow alpha**:

$$\alpha \sim \mathcal{N}(1000, 5)$$

- **Separate narrow beta**:

$$\beta \sim \mathcal{N}(0, 1)$$

- **Hierarcchical narrow alpha**:

$$\mu_\alpha \sim \mathcal{N}(1000, 5)$$
$$\sigma_\alpha \sim \mathcal{N}(0, 3)$$

- **Hierarcchical narrow beta**:

$$\mu_\beta \sim \mathcal{N}(0, 5)$$
$$\sigma_\beta \sim \mathcal{N}(0, 1)$$

The sample predictive posterior plots for these models are depicted in Figures 6 and 7. Notably, changing the priors for α wields the most significant influence on the results, which aligns intuitively with α's role in determining the credit score's scale. Moreover, another observation is that the hierarchical models appear less bothered by changes in priors compared to the separate models. This stability emphasizes the hierarchical model's robustness in handling variations in priors, highlighting its superior adaptability in this regard.

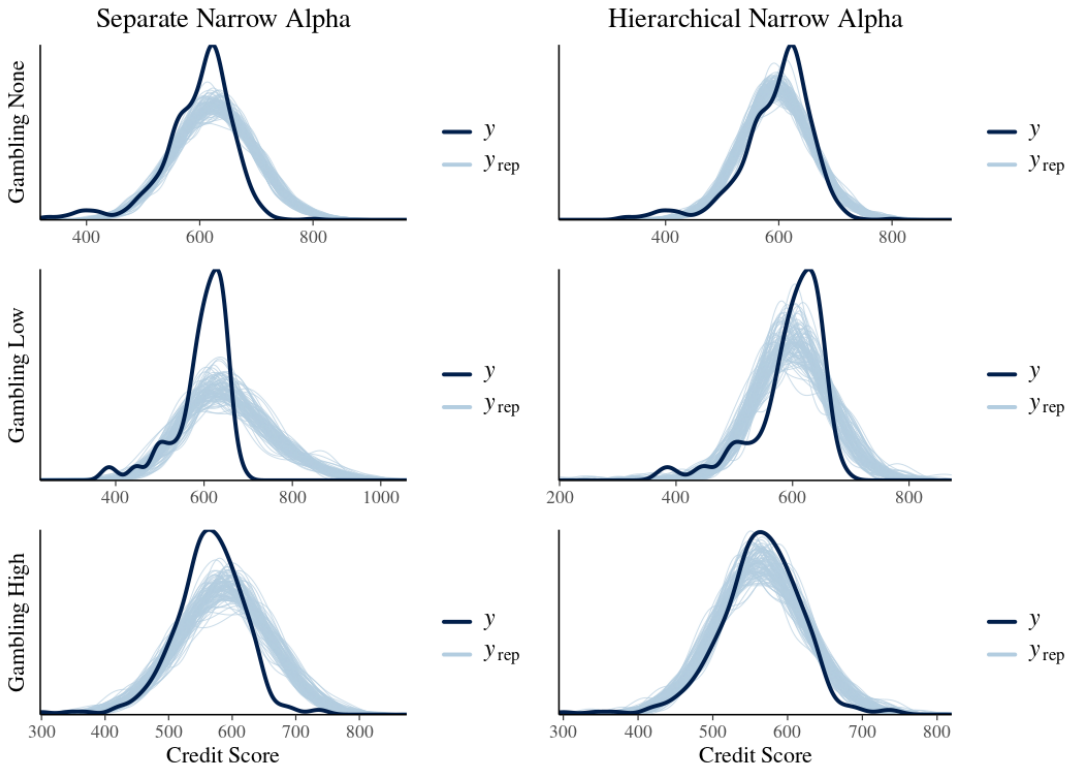

Figure 6: Predictive posterior sample vs true distribution for models with narrow alpha prior.

3.7 Model comparison

Leave-One-Out Cross-Validation is a widely-used technique for model evaluation, systematically assessing predictive performance by iteratively removing each data point and estimating its likelihood based on the

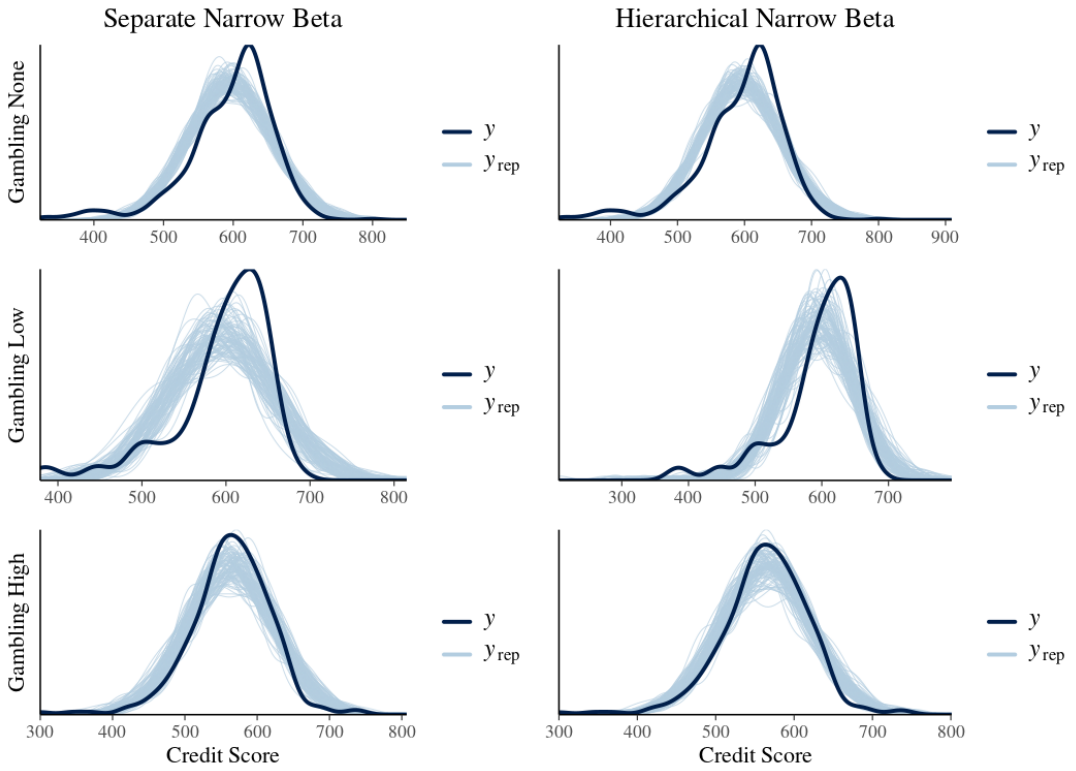

Figure 7: Predictive posterior sample vs true distribution for models with narrow beta prior.

remaining data. LOO provides a reliable measure of model fit, typically expressed through the expected log pointwise predictive density (ELPD). Comparing models involves evaluating the difference in ELPD (ELPD-diff).

However, in cases where the tail behavior of importance sampling weights deviates significantly from expected patterns, the reliability of LOO estimates may be compromised. The Pareto Smoothed Importance Sampling (PSIS) diagnostic identifies unreliable estimates and quantifies the severity of issues through K-values. LOO combined with PSIS and K-values ensures the robustness of Bayesian models, facilitating informed decision-making in diverse data analysis contexts.

Using the code below, we've compared all of the models with every prior. We can see that the best model is hierarchical model with weak infromative priors.

```
loo_hier <- loo(hier_fit$draws("log_lik"))
loo_sep <- loo(separate_fit$draws("log_lik"))
loo_hier_na <- loo(hier_narrow_alpha_fit$draws("log_lik"))
loo_sep_na <- loo(separate_narrow_alpha_fit$draws("log_lik"))
loo_hier_nb <- loo(hier_narrow_beta_fit$draws("log_lik"))
loo_sep_nb <- loo(separate_narrow_beta_fit$draws("log_lik"))

loo_compare(loo_hier, loo_hier_na, loo_hier_nb, loo_sep, loo_sep_na, loo_sep_nb)
```

```
elpd_diff se_diff
model1    0.0       0.0
model2   -8.9       4.4
model4  -32.5       7.4
model3  -37.7       8.0
```

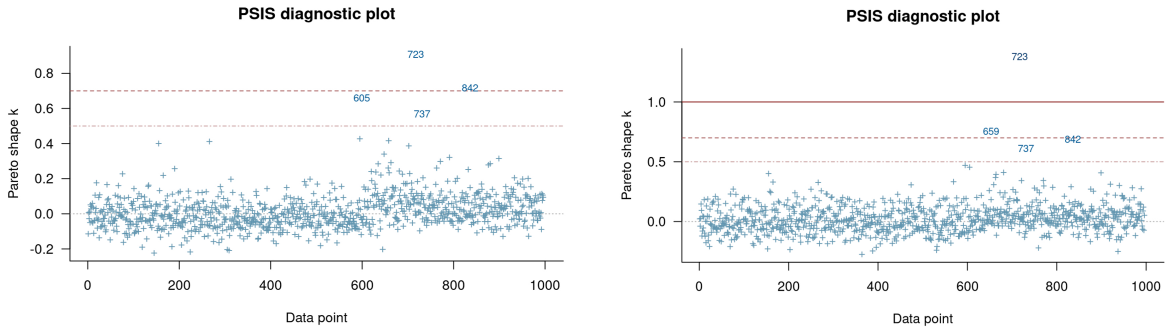

Figure 8: K-value plots for the separate models (on left) and the hierarchical model (on right)

```
model6 -246.7      20.0
model5 -337.1      35.0
```

We can see the the K-value plots for the separate models (on left) and the hierarchical model (on right) with weak priors in Fig-8. It can be observed that one of the data points has a relatively high k-value but apart from that, all of the data points have an acceptable k-value.

4 Conclusion

In this study, we attempted to develop a credit score prediction model designed for three distinct groups based on gambling habits, utilizing features such as income, debt, and savings, among others. Utilizing separate and hierarchical models with differing priors, we examined their performance through the lens of leave-one-out cross-validation (LOO-CV) and sensitivity to prior specifications. Through comprehensive evaluation, the hierarchical model emerged as the most robust model. Thus, based on the evidence obtained from this analysis, the base hierarchical model stands out as the optimal choice for credit score prediction among the models explored.

5 Issues and potential improvements

While the hierarchical model demonstrated superior performance and robustness overall, it encountered convergence challenges, with not all chains converging as desired. Addressing this issue may involve adjusting the number of chains and increasing iteration and warmup steps to promote convergence. Additionally, the utilization of a simple linear regression model, while effective, may not fully capture the irregularities present in the true distributions. Exploring more complex models could potentially enhance prediction accuracy by better adapting the nuances inherent in the data.

6 Self-reflection

Through the course of this project, we gained valuable insights into the significant impact of feature scale on model performance. Our findings underscored the importance of normalizing features to a consistent scale, as disparities in scale can highly influence model outcomes. Furthermore, our exploration into hierarchical models revealed their remarkable robustness against variations in priors. Compared to separate models, hierarchical approaches exhibited greater stability and reliability, particularly in scenarios where

prior information was limited. This insight highlights the versatility of hierarchical models, suggesting their potential superiority in settings where there's uncertainty about prior distributions.

Appendix

A Sepratate Stan code

```
data {
  int<lower=1> N1;
  int<lower=1> N2;
  int<lower=1> N3;
  int<lower=1> N_features;
  matrix[N1, N_features] x1;
  matrix[N2, N_features] x2;
  matrix[N3, N_features] x3;
  vector[N1] y1;
  vector[N2] y2;
  vector[N3] y3;
}

parameters {
  real alpha1;
  vector[N_features] beta1;
  real alpha2;
  vector[N_features] beta2;
  real alpha3;
  vector[N_features] beta3;

  real<lower=0> sigma;
}

model {
  alpha1 ~ normal(1000, 100);
  beta1 ~ normal(0, 10);
  alpha2 ~ normal(1000, 100);
  beta2 ~ normal(0, 10);
  alpha3 ~ normal(1000, 100);
  beta3 ~ normal(0, 10);

  sigma ~ normal(0, 10);

  y1 ~ normal(alpha1 + x1*beta1, sigma);
  y2 ~ normal(alpha2 + x2*beta2, sigma);
  y3 ~ normal(alpha3 + x3*beta3, sigma);
}

generated quantities {
  array[N1] real y1_rep = normal_rng(alpha1 + x1*beta1, sigma);
  array[N2] real y2_rep = normal_rng(alpha2 + x2*beta2, sigma);
  array[N3] real y3_rep = normal_rng(alpha3 + x3*beta3, sigma);

  vector[N1+N2+N3] log_lik;

  for (i in 1:N1) {
    log_lik[i] = normal_lpdf(y1[i] | x1[i] * beta1 + alpha1, sigma);
  }
  for (i in 1:N2) {
    log_lik[i+N1] = normal_lpdf(y2[i] | x2[i] * beta2 + alpha2, sigma);
  }
```

```
  for (i in 1:N3) {
    log_lik[i+N1+N2] = normal_lpdf(y3[i] | x3[i] * beta3 + alpha3, sigma);
  }
}
```

B Hierarchical Stan code

```
data {
  int<lower=1> N1;
  int<lower=1> N2;
  int<lower=1> N3;
  int<lower=1> N_features;
  matrix[N1, N_features] x1;
  matrix[N2, N_features] x2;
  matrix[N3, N_features] x3;
  vector[N1] y1;
  vector[N2] y2;
  vector[N3] y3;
}

parameters {
  real mu_hier_alpha;
  real mu_hier_beta;

  real std_hier_alpha;
  real std_hier_beta;

  real alpha1;
  vector[N_features] beta1;
  real alpha2;
  vector[N_features] beta2;
  real alpha3;
  vector[N_features] beta3;

  real<lower=0> sigma;
}

model {
  mu_hier_alpha ~ normal(1000, 100);
  std_hier_alpha ~ normal(0, 30);
  mu_hier_beta ~ normal(0, 100);
  std_hier_beta ~ normal(0, 10);

  alpha1 ~ normal(mu_hier_alpha, std_hier_alpha);
  beta1 ~ normal(mu_hier_beta, std_hier_beta);
  alpha2 ~ normal(mu_hier_alpha, std_hier_alpha);
  beta2 ~ normal(mu_hier_beta, std_hier_beta);
  alpha3 ~ normal(mu_hier_alpha, std_hier_alpha);
  beta3 ~ normal(mu_hier_beta, std_hier_beta);

  sigma ~ normal(0, 10);

  y1 ~ normal(alpha1 + x1*beta1, sigma);
```

```
  y2 ~ normal(alpha2 + x2*beta2, sigma);
  y3 ~ normal(alpha3 + x3*beta3, sigma);
}

generated quantities {
  array[N1] real y1_rep = normal_rng(alpha1 + x1*beta1, sigma);
  array[N2] real y2_rep = normal_rng(alpha2 + x2*beta2, sigma);
  array[N3] real y3_rep = normal_rng(alpha3 + x3*beta3, sigma);

  vector[N1+N2+N3] log_lik;

  for (i in 1:N1) {
    log_lik[i] = normal_lpdf(y1[i] | x1[i] * beta1 + alpha1, sigma);
  }
  for (i in 1:N2) {
    log_lik[i+N1] = normal_lpdf(y2[i] | x2[i] * beta2 + alpha2, sigma);
  }
  for (i in 1:N3) {
    log_lik[i+N1+N2] = normal_lpdf(y3[i] | x3[i] * beta3 + alpha3, sigma);
  }
}
```