BDA Project - Credit Risk Analysis

An Do, Nam Bui, Ayush Pradhan

November 2025

Introduction

Credit provision (handing out loans) is a fundamental function of banking institutions. A loan is essentially a sum of money or other assets provided to a borrower with the expectation of repayment in the future, including the original amount plus any interest or fees. Loans can take various forms and serve multiple purposes, such as financing consumption, business ventures, or investments. They also play a vital role in economic growth by increasing liquidity and supporting new business activities.

However, credit transactions inherently carry the risk that borrowers may fail to meet their repayment obligations, a phenomenon known as credit default or credit risk. When a bank receives a loan application, it must decide whether to approve or reject it based on the applicant's profile. This decision involves two key risks: if a good credit risk is rejected, the bank loses a business opportunity; if a bad credit risk is approved, the bank faces potential financial loss. Accurate assessment of credit risk is therefore crucial, as misjudgment can lead to significant financial losses, institutional instability, and broader systemic consequences, as seen during the 2007–2008 financial crisis.

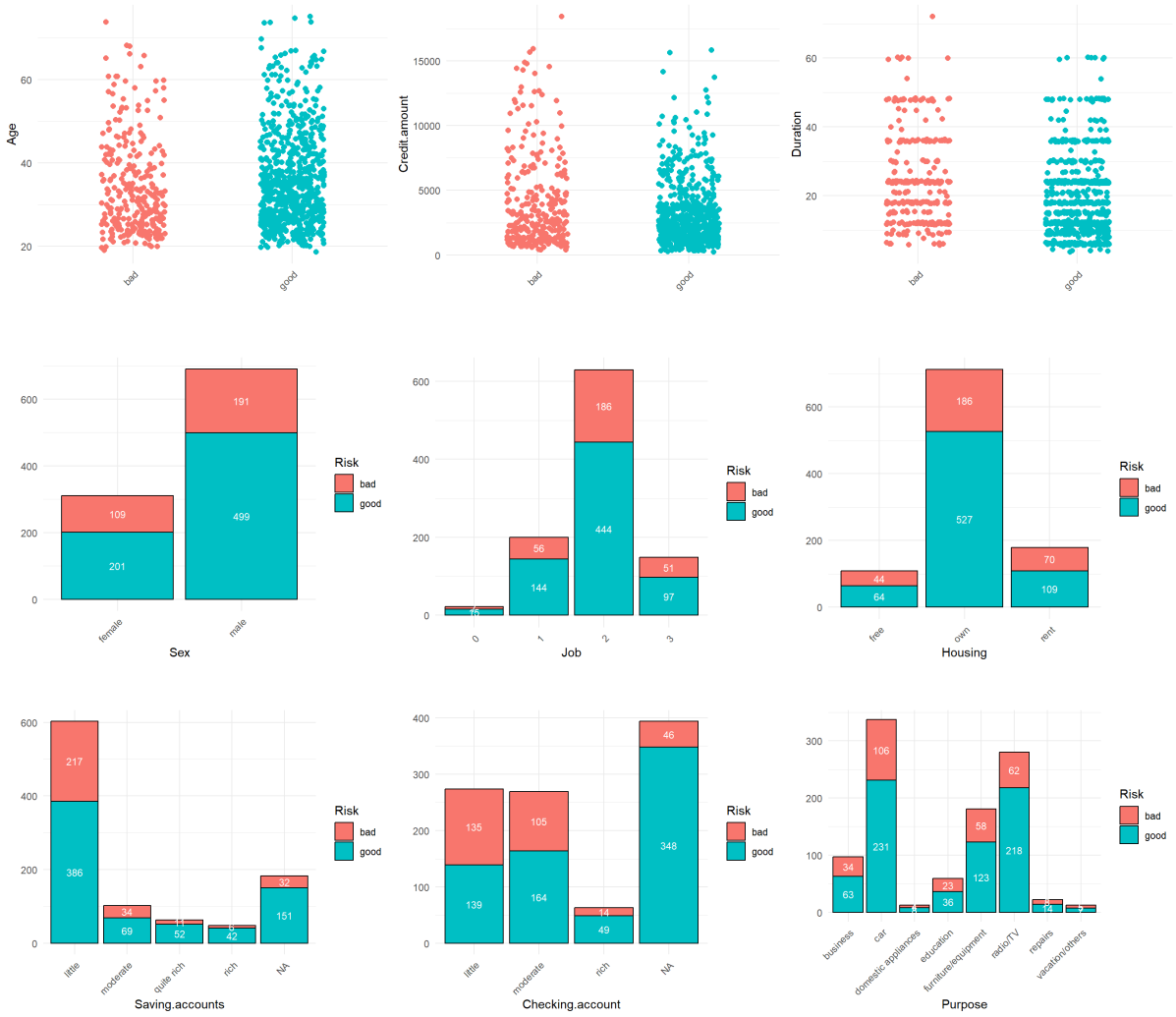
We chose this project because effective risk management is vital for the stability and profitability of banking institutions. Optimizing the process of detecting high-risk borrowers can help minimize losses and maximize returns, therefore, making lending decisions faster and more reliable.

Our project addresses the binary classification problem of credit risk assessment: determining whether a loan application should be classified as low-risk or high-risk (default) based on borrower characteristics and loan features. To model this, we employ both pooled logistic regression and hierarchical logistic regression, which are well-suited to the binary nature of loan outcomes.

Data Description

Data Overview

Firstly, the histograms of the feature variables with regards to the target variable Risk are examined.



The data also contains some missing values.

Variable	Proportion Missing
X	0.000
Age	0.000
Sex	0.000
Job	0.000
Housing	0.000
Saving.accounts	0.183
Checking.account	0.394
Credit.amount	0.000
Duration	0.000
Purpose	0.000
Risk	0.000

From the histograms, three main issues were identified in the dataset: missing values in *Saving.accounts* and *Checking.account*, category mismatches between these two columns, and dispersed data in *Purpose*.

The dataset contains 18.3% missing values in *Saving.accounts* and 39.4% in *Checking.account*. Deleting these entries or imputing them with the mode could distort the data distribution and remove potentially

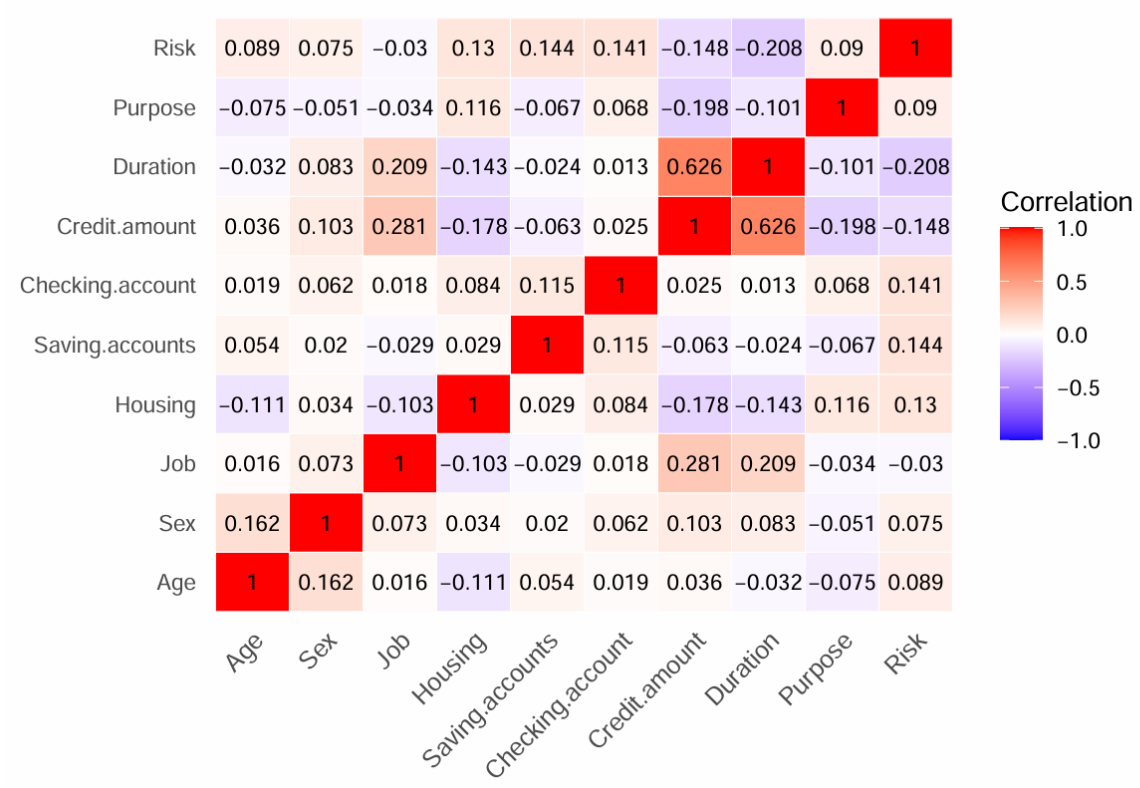
important information, as missing accounts may predict credit risk. To address this, k-Nearest Neighbors (kNN) imputation was applied which estimates missing values based on the similarity to other observations, preserving the underlying patterns in the data.

In terms of category mismatches, both *Saving.accounts* and *Checking.account* share the categories `'little'`, `'moderate'`, and `'rich'`, but *Saving.accounts* includes an additional `'quite rich'` category. All `'quite rich'` values were recoded as `'rich'` to ensure consistency.

The *Purpose* column is highly dispersed, with several low-population categories such as `'repairs'` and `'domestic appliances'`. To reduce sparsity, these categories were merged into `'furniture/equipment'`, creating more balanced groups.

To improve model performance and ensure comparability across features, we perform standardization. To standardize, we transform all variables (including the target) to numerical. Then, we scale the explanatory variables them so that their mean = 0 and standard deviation = 1.

Explanatory Variables Analysis



Overall, the correlations between the explanatory features and the target Risk are quite weak, suggesting that no single feature is a strong linear predictor of Risk on its own. No noticeable multicollinearity is detected. While Duration shows the strongest correlation with Risk (-0.21), Job have the lowest correlation with the target variable (-0.03), indicating virtually no linear correlation between these variables and the target variable.

For the pooled model, we decided to incorporate all 9 explanatory variables into the prediction models, with Job and Housing being weakly correlated to the target variable Risk. Detailed measures to mitigate the effects of the weakly correlated variable on the prediction results will be discussed later on. For the hierarchical

model, the variable Purpose is utilized as the grouping variable, and 8 remaining variables will be used as explanatory variables for the prediction model.

Models Description

Pooled Logistic Regression

Formula

The Logistic Regression model is used to classify the credit risk y associated with lending, based on a vector of observed features x.
The probability of default p is modeled using an intercept β_0 and coefficients $\beta_1, \beta_2, ..., \beta_k$ applied to each covariate:

$$p = \frac{e^{\beta_0+\beta_1 x_{n,1}+\cdots+\beta_k x_{n,k}}}{1+e^{\beta_0+\beta_1 x_{n,1}+\cdots+\beta_k x_{n,k}}} = \text{logit}^{-1}\left(\beta_0 + \beta_1 x_{n,1} + \cdots + \beta_k x_{n,k}\right)$$

Given this predictor, the target outcome for each observation follows a Bernoulli likelihood:

$$y_n \sim \text{Bernoulli}(p_n) = \text{Bernoulli}\Big(\text{logit}^{-1}(\beta_0 + \beta_1 x_{n,1} + \cdots + \beta_k x_{n,k})\Big)$$

Here,
- β_0 is the intercept,
- $\beta_1, \beta_2, ..., \beta_k$ are regression coefficients corresponding to each feature in x,
- and p_n is the predicted probability of default for observation n.

Priors justification

For the β coefficients, a common choice is to use Normal distributions as priors due to their mathematical convenience and their theoretical justification via the central limit theorem. A typical form is

$$\beta_i \sim \mathcal{N}(0, \sigma^2),$$

where σ^2 controls the degree of prior uncertainty.
In this project, we specify a Normal prior with mean 0 and variance 1 for most of the coefficients. This choice reflects weak prior information : we do not have strong knowledge about how each predictor influences the outcome, so we adopt a broad prior that allows the data substantial flexibility while still preventing extreme parameter values.

Hierarchical Logistic Regression

Formula

The hierarchical model, while based on the same logistic regression framework as the pooled model, introduces an additional layer of complexity. As described previously in the explanatory variable analysis, the dataset is partitioned into L groups according to the *Job* category. Instead of estimating a single set of coefficients, the model assigns each group its own parameters $\beta_0, \beta_1, \ldots, \beta_k$, allowing the relationship between predictors and credit risk to vary across job types. These group-level coefficients are drawn from hyper-distributions characterized by mean μ and variance σ^2, enabling partial pooling and capturing shared structure while still accounting for between-group heterogeneity.

With this hierarchical structure, the likelihood for each observation is defined as:

$$y_n \sim \text{Bernoulli}\left(\text{logit}^{-1}\left(\beta_{0,L(n)} + \beta_{1,L(n)} x_{n,1} + \cdots + \beta_{k,L(n)} x_{n,k}\right)\right),$$

where $L(n)$ refers to the job group associated with observation n.

Priors justification

The coefficients for each group are assigned Normal priors with shared hyper-parameters:

$$\beta_{i,L(n)} \sim \mathcal{N}(\mu_i, \sigma_i^2), \quad i = 1 : k.$$

The hyper-mean parameters follow weakly informative priors:

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

which is appropriate given that all predictors have been standardized, preventing strong prior influence on coefficient magnitude and direction.

To model uncertainty in group-level variability, the hyper-variance parameters are assigned inverse-gamma priors:

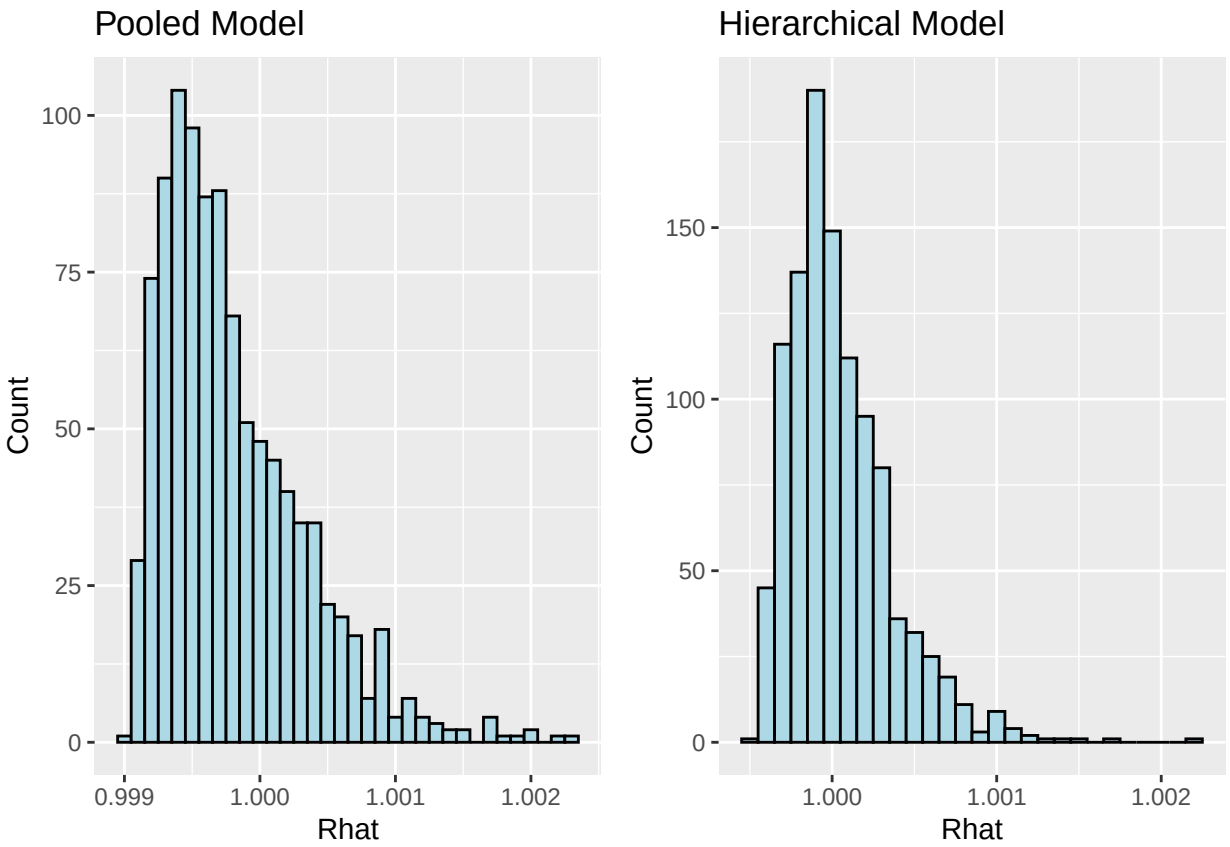
$$\sigma_i^2 \sim \text{InvGamma}(0.5, 1).$$

Result Analysis

Convergence Diagnostics

$\hat{R}$ Diagnostics

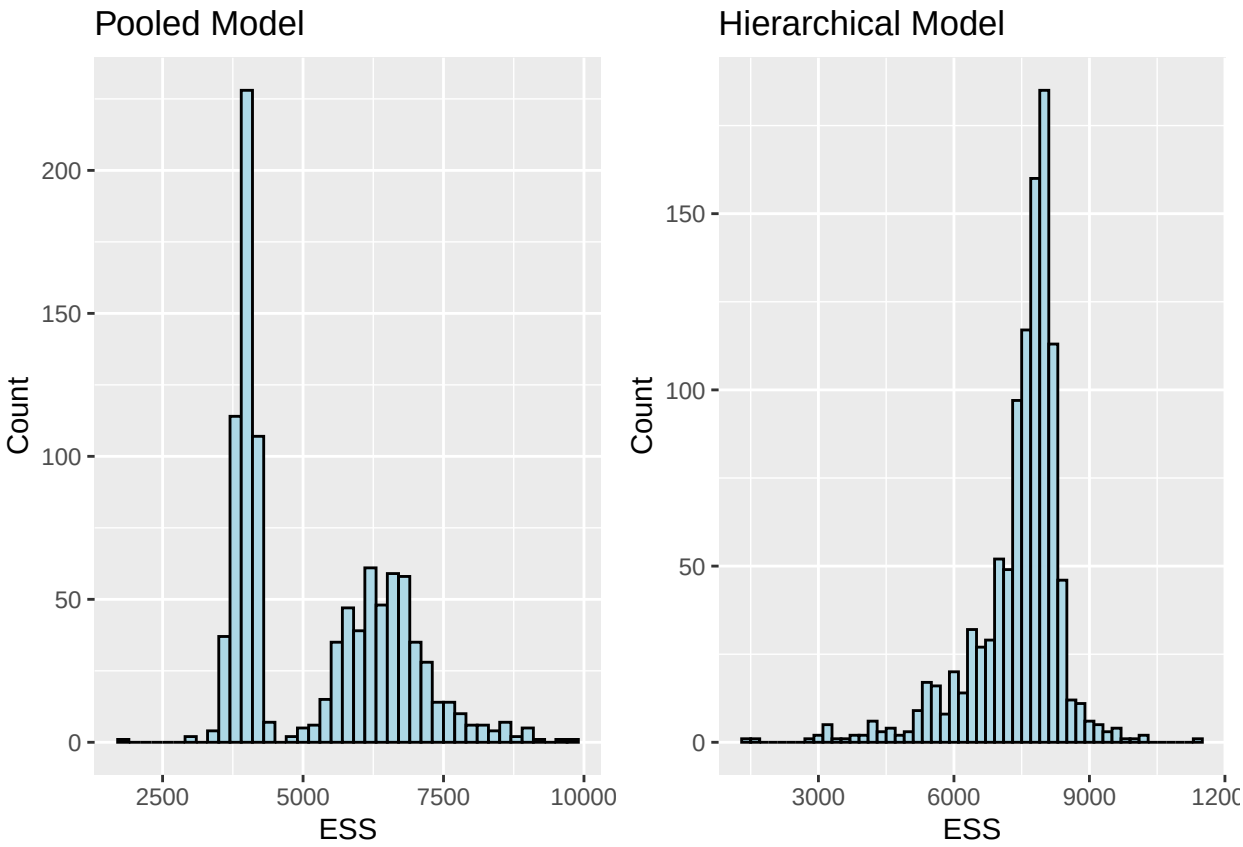
The $\hat{R}$ values for the models are plotted as histograms.



The histograms clearly show that all $\hat{R}$ values are below 1.05, and their distributions are tightly concentrated around 1.00. This indicates that the MCMC chains for both models have converged well.

ESS diagnostics

The Effective Sample Size (ESS) values of the parameters for the two models are plotted in the histograms below.



Across both the pooled and hierarchical models, all parameters exhibit ESS values well above 100 times the number of chains, indicating that the Markov chains mixed efficiently and achieved strong sampling performance. This suggests that each parameter was estimated with a sufficiently large number of effective posterior draws, supporting stable and reliable inference. Overall, the consistently high ESS values across all parameters provide confidence in the precision and robustness of our posterior estimates.

Divergence

We find no evidence of divergence; the ESS values are sufficiently large and stable across all chains which is a sign of good mixing and reliable posterior estimates.

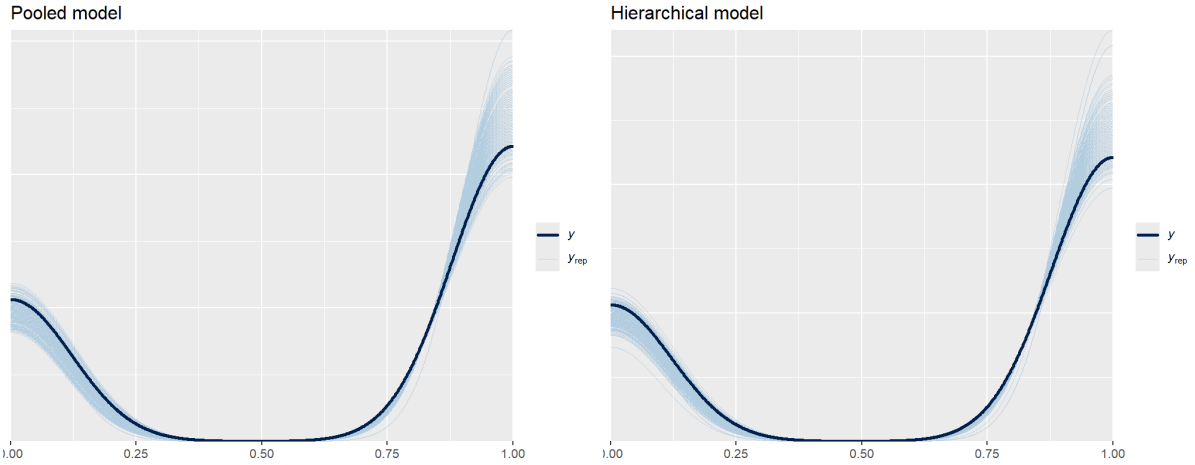
```
## Pooled model
```

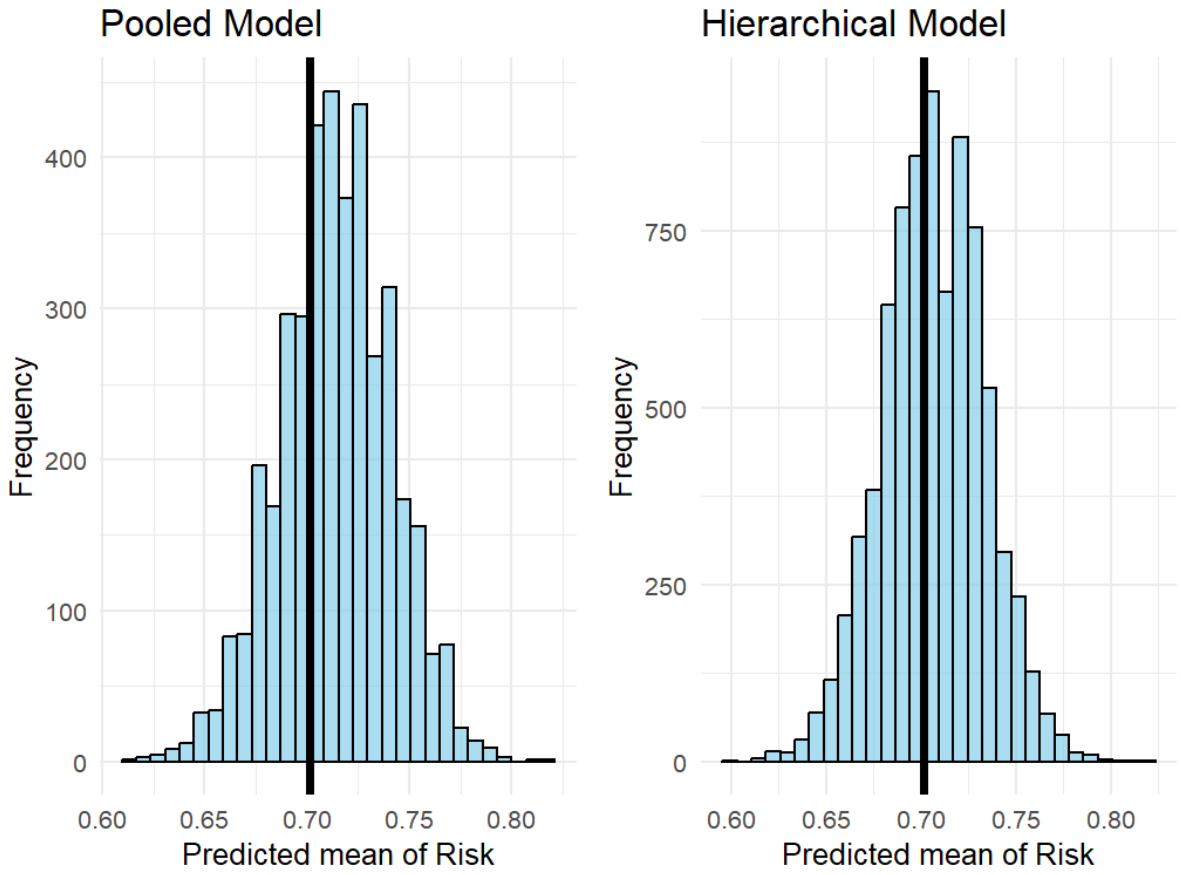
```
##  accept_stat__      stepsize__      treedepth__     n_leapfrog__    divergent__
##  Min.   :0.3526   Min.   :0.5188   Min.   :2.000   Min.   : 3.00   Min.   :0
##  1st Qu.:0.8572   1st Qu.:0.5317   1st Qu.:3.000   1st Qu.: 7.00   1st Qu.:0
##  Median :0.9387   Median :0.5535   Median :3.000   Median : 7.00   Median :0
##  Mean   :0.9050   Mean   :0.5506   Mean   :2.943   Mean   : 6.94   Mean   :0
##  3rd Qu.:0.9827   3rd Qu.:0.5724   3rd Qu.:3.000   3rd Qu.: 7.00   3rd Qu.:0
##  Max.   :1.0000   Max.   :0.5764   Max.   :4.000   Max.   :15.00   Max.   :0
##     energy__
##  Min.   :270.8
##  1st Qu.:276.1
##  Median :277.9
##  Mean   :278.3
##  3rd Qu.:280.3
##  Max.   :290.8
```

```
## Hierarchical model
```

```
##  accept_stat__      stepsize__       treedepth__     n_leapfrog__
##  Min.   :0.3685   Min.   :0.01003   Min.   :8.000   Min.   : 255.0
##  1st Qu.:0.8945   1st Qu.:0.01032   1st Qu.:8.000   1st Qu.: 255.0
##  Median :0.9564   Median :0.01137   Median :8.000   Median : 511.0
##  Mean   :0.9250   Mean   :0.01131   Mean   :8.431   Mean   : 408.5
##  3rd Qu.:0.9871   3rd Qu.:0.01236   3rd Qu.:9.000   3rd Qu.: 511.0
##  Max.   :1.0000   Max.   :0.01246   Max.   :9.000   Max.   :1023.0
##   divergent__    energy__
##  Min.   :0    Min.   :257.3
##  1st Qu.:0    1st Qu.:284.4
##  Median :0    Median :291.0
##  Mean   :0    Mean   :291.4
##  3rd Qu.:0    3rd Qu.:298.1
##  Max.   :0    Max.   :338.6
```

Posterior predictive check



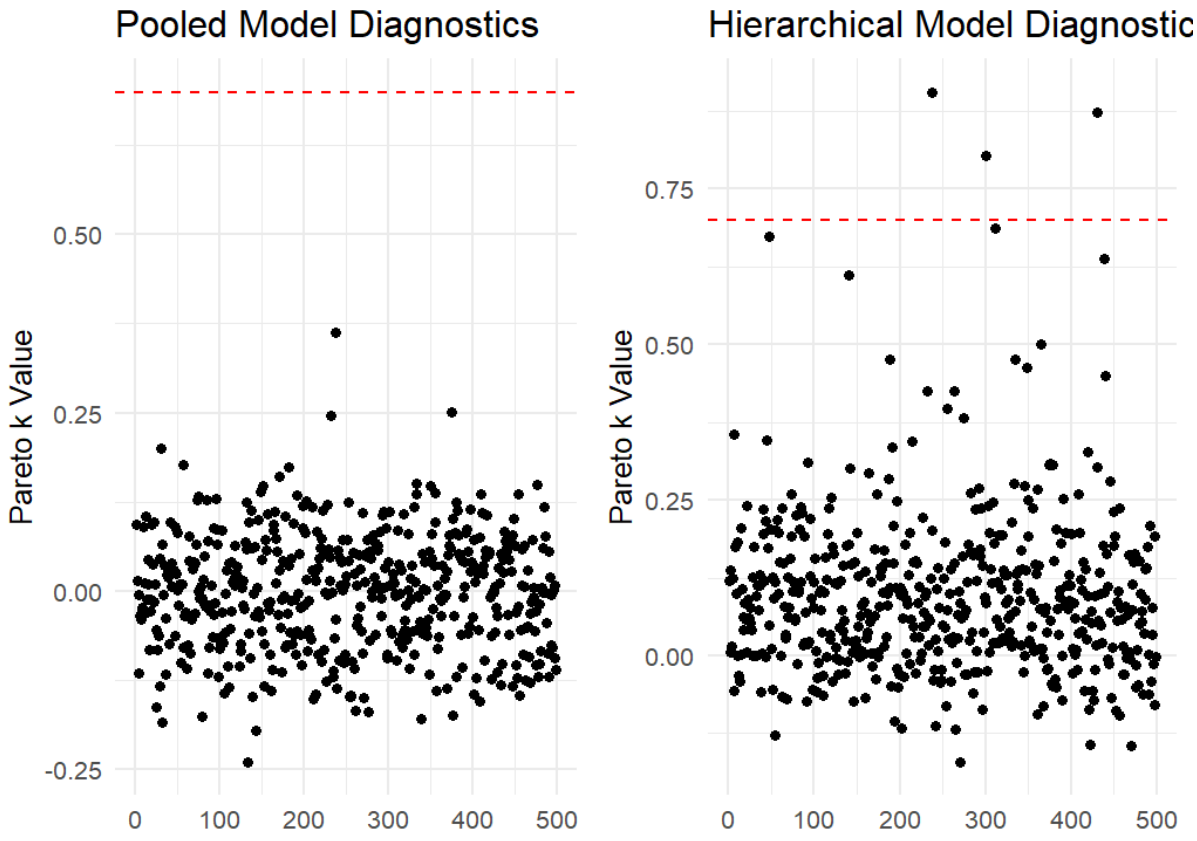


The posterior predictive check (PPC) plots show a strong alignment between the observed data and the simulated predictions in terms of central tendency. The posterior predictive means closely match the true mean which indicates that the model captures the main patterns in the risk defaulting data.

Comparison

Model comparison is performed using Leave-One-Out Cross-Validation (LOO-CV), which estimates the models' generalization performance.

Firstly, the Pareto k values are plotted to assess the reliability of the elpd estimates.



While all Pareto k values of the pooled model are below 0.5, indicating very reliable estimates, three of the hierarchical model's k values exceeds 0.7, which is usually considered a threshold for concern. This suggests that the LOO estimate for the hierarchical model might be too optimistic.

The LOO-CV estimation of the expected log predictive density (elpd) for both models is provided and compared below.

```
##
## Computed from 4000 by 499 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -276.9 11.5
## p_loo         9.6  0.7
## looic       553.8 23.0
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume independent draws (r_eff=1).
##
## All Pareto k estimates are good (k < 0.7).
```

```
## See help('pareto-k-diagnostic') for details.
##
## Computed from 8000 by 499 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -290.4 14.2
## p_loo        46.9  4.5
## looic       580.8 28.4
## ------
## MCSE of elpd_loo is NA.
## MCSE and ESS estimates assume independent draws (r_eff=1).
##
## Pareto k diagnostic values:
##                           Count Pct.    Min. ESS
## (-Inf, 0.7]   (good)     497   99.6%   349
##    (0.7, 1]   (bad)        2    0.4%   <NA>
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
##        elpd_diff se_diff
## model1   0.0       0.0
## model2 -13.5       7.5
```

The LOO-CV results indicate that the pooled model slightly outperforms the hierarchical model in terms of predictive accuracy. All Pareto k values for the pooled model are below 0.7, reflecting highly reliable LOO estimates. In contrast, the hierarchical model has a few k values above 0.7, indicating minor instability. The elpd difference of -2.1 (SE = 5.9) further favors the pooled model, suggesting it should be preferred for out-of-sample prediction.

Predictive Performance Assessment

To assess the models' predictive performance, we compute a confusion matrix for the test set. Predictions are obtained by taking the mean over all posterior draws for each observation; values > 0.5 are classified as 1, otherwise 0.

```
                Actual
Predicted        Actual True Actual False
  Predicted True          320          133
  Predicted False          17           29
[1] 0.6993988
[1] 0.1790123
                Actual
Predicted        Actual True Actual False
  Predicted True          303          123
  Predicted False          34           39
[1] 0.6853707
[1] 0.2407407
```

Both models achieve moderate overall accuracy, but their ability to correctly detect bad loans (Specificity) is very low—below 25% for both models. While the pooled model is slightly better in overall predictions, neither model is reliable enough for practical use in banking, where correctly identifying bad loans is critical to prevent financial losses.

Prior Sensitivity Analysis

Conclusion

Summary

In this project, we developed two Bayesian models—pooled logistic regression and hierarchical logistic regression to classify credit applications as high or low risk. Both models showed strong convergence and sampling behavior, with R_hat values close to 1.00 and high effective sample sizes, indicating reliable posterior estimation. Posterior predictive checks further suggested that the models captured the central structure of the observed risk outcomes.

Model comparison via Leave-One-Out Cross-Validation (LOO-CV) showed a slight performance advantage for the pooled model, supported by more stable Pareto-k diagnostics and a better estimated log predictive density. Although predictive accuracy for low-risk cases was acceptable, both models struggled to identify high-risk borrowers, limiting their practical utility for real banking decisions.

Challenges and Potential Improvements

While the pooled model demonstrated marginally stronger predictive performance, neither model performed sufficiently well in detecting defaulting applicants, with specificity remaining below an actionable threshold. This highlights the need for improvement in both feature representation and model structure.

Future work can include the elements: - Adding more informative financial variables (e.g., income, credit history, external credit score).
- Testing more flexible models such as Bayesian additive trees or boosted decision methods.
- Applying class rebalancing or asymmetric loss weighting to better penalize false negatives.
- Extending hierarchical grouping beyond job category to capture broader population heterogeneity.

Self Reflection

Throughout the project, we gained a deeper understanding of various Bayesian modelling techniques and their practical applications. One of the most significant challenges we encountered was handling the large proportion of missing values in the dataset. After considering several options, we chose to address this issue using kNN imputation due to its ability to preserve underlying data structure. However, we recognize that this may not be the optimal approach, and other imputation techniques could potentially yield better performance. Identifying the most suitable missing-value strategy remains an important point of consideration and a direction for future improvement.

Appendix

```
## library(corrplot)
## library(VIM)
## library(caret)
## library(ggplot2)
## library(dplyr)
## library(rstan)
## library(gridExtra)
##
## setwd("C:/Users/An/OneDrive/Desktop/Master/BDA/Project")
##
## data_pooled<- read.csv("data_cleaned.csv")
##
##
```

```
## pred_pooled <- setdiff(names(data), c("Risk"))
## data_pooled[pred_pooled] <- scale(data[pred_pooled])
##
## split_data <- function(df, target_cols, proportion = 0.5) {
##   n <- nrow(df)
##   train_size <- round(n * proportion)
##   train <- head(df, train_size)
##   test <- tail(df, n - train_size)
##
##   list(
##     train = train,
##     test = test,
##     y_train = train[[target_cols]],
##     y_test = test[[target_cols]]
##   )
## }
##
## # Pooled Model----
## pooled_split <- split_data(data_pooled, "Risk")
##
## X_train_pooled <- subset(pooled_split$train, select = -Risk)
## X_test_pooled  <- subset(pooled_split$test, select = -Risk)
##
## credit_data_pooled <- list(
##   N_train = nrow(X_train_pooled),
##   K = ncol(X_train_pooled),
##   X_train = X_train_pooled,
##   y_train = pooled_split$y_train,
##   N_test = nrow(X_test_pooled),
##   X_test = X_test_pooled
## )
##
## pooled_fit <- stan(
##   file = "pooled.stan",
##   data = credit_data_pooled,
##   chains = 4, iter = 2000, warmup = 1000
## )
##
## fit_pooled <- summary(pooled_fit)$summary
##
## saveRDS(pooled_fit, file = "pooled_fit.rds")
```

```
##  data {
##  int<lower=0> N_train;
##  int<lower=0> K;
##  matrix[N_train, K] X_train;
##  // Number of observation in train set
##  // Number of columns
##  // Observations in train set
##  int<lower=0, upper=1> y_train[N_train]; // Target value in train set
##  int<lower=0> N_test;
##  matrix[N_test, K] X_test;
##  }
##  parameters {
```

```
##  real beta_0;
##  vector[K] beta;
##  }
##  model {
##  // Set priors
##  beta_0 ~ normal(0, 1);
##  beta ~ normal(0, 1);
##  // Set regularization priors
## beta[3] ~ normal(0, 0.01);
##
##  // Compute likelihood
##  // Number of observation in test set
##  // Observations in test set
##  y_train ~ bernoulli_logit(beta_0 + X_train * beta);
##  }
##  generated quantities {
##  // Compute predictions
##  int<lower=0, upper=1> y_pred[N_test] = bernoulli_logit_rng(beta_0 + X_test * beta);
##  // Compute log likelihood
##  vector[N_train] log_lik;
##  for(i in 1:N_train)
##  log_lik[i] = bernoulli_logit_lpmf(y_train[i] | beta_0 + X_train[i] * beta);
##  }
##
## library(corrplot)
## library(VIM)
## library(caret)
## library(ggplot2)
## library(dplyr)
## library(rstan)
## library(gridExtra)
##
## setwd("C:/Users/An/OneDrive/Desktop/Master/BDA/Project")
##
## data_hierarchical <- read.csv("data_cleaned.csv")
##
## pred_hierarchical <- setdiff(names(data), c("Risk", "Job"))
## data_hierarchical["Job"] <- lapply(data_hierarchical["Job"], function(x) x +1)
## data_hierarchical[pred_hierarchical] <- scale(data[pred_hierarchical])
##
## split_data <- function(df, target_cols, proportion = 0.5) {
##   n <- nrow(df)
##   train_size <- round(n * proportion)
##   train <- head(df, train_size)
##   test <- tail(df, n - train_size)
##
##   list(
##     train = train,
##     test = test,
##     y_train = train[[target_cols]],
##     y_test = test[[target_cols]]
##   )
## }
```

```
##
## # Hierarchical Model----
## hier_split <- split_data(data_hierarchical, "Risk")
##
## X_train <- subset(hier_split$train, select = -c(Risk, Job))
## X_test  <- subset(hier_split$test, select = -c(Risk, Job))
## ll_train <- hier_split$train$Job
## ll_test  <- hier_split$test$Job
##
## credit_data_hierarchical <- list(
##   N_train = nrow(X_train),
##   K = ncol(X_train),
##   L_train = length(unique(ll_train)),
##   X_train = as.matrix(X_train),
##   ll_train = ll_train,
##   y_train = hier_split$y_train,
##   N_test = nrow(X_test),
##   L_test = length(unique(ll_test)),
##   X_test = as.matrix(X_test),
##   ll_test = ll_test
## )
##
## hierarchical_fit <- stan(
##   file = "C:/Users/An/OneDrive/Desktop/Master/BDA/Project/hierarchical.stan",
##   data = credit_data_hierarchical,
##   chains = 4, iter = 3000, warmup = 1000
## )
##
## # Diagnostics----
## fit_hierarchical <- summary(hierarchical_fit)$summary
##
## #Save
## saveRDS(hierarchical_fit, file = " hierarchical_fit.rds")
```

```
##  data {
##  int<lower=0> N_train;
##  int<lower=0> K;
##  int<lower=0> L_train;
##  matrix[N_train, K] X_train;
##  // Number of observation in train set
##  // Number of columns
##  // Number of groups in train set
##  // Observations in train set
##  int<lower=0, upper=L_train> ll_train[N_train]; // Group level in train set
##  int<lower=0, upper=1> y_train[N_train];
##  // Target value in train set
##  int<lower=0> N_test;
##  int<lower=0> L_test;
##  matrix[N_test, K] X_test;
##  int<lower=0, upper=L_test> ll_test[N_test];
##  }
##  parameters {
##  real mu_0;
##  real<lower=0> sigma_0;
```

```
##  real beta_0[L_train];
##  real mu[K];
##  real<lower=0> sigma[K];
##  vector[K] beta[L_train];
##  }
##  model {
##  // Set priors
##  mu_0 ~ normal(0, 1);
##  sigma_0 ~ inv_gamma(0.5, 1);
##  beta_0 ~ normal(mu_0, sigma_0);
##  mu ~ normal(0, 1);
##  sigma ~ inv_gamma(0.5, 1);
##  // Set regularization priors
##  //mu[3] ~ normal(0, 0.01);
##  //sigma[3] ~ inv_gamma(1, 0.01);
##  // Set priors
##  for(i in 1:L_train) {
##  beta[i] ~ normal(mu, sigma);
##  }
##  // Compute likelihood
##  for(i in 1:N_train)
##  // Number of observation in test set
##  // Number of groups in test set
##  // Observations in test set
##  // Group level in test set
##  y_train[i] ~ bernoulli_logit(beta_0[ll_train[i]] + X_train[i] * beta[ll_train[i]]);
##  }
##  generated quantities {
##    int<lower=0, upper=1> y_pred[N_test];
##  vector[N_train] log_lik;
##  // Compute predictions
##  for(i in 1:N_test)
##  y_pred[i] = bernoulli_logit_rng(beta_0[ll_test[i]] + X_test[i] * beta[ll_test[i]]);
##  // Compute log likelihood
##  for(i in 1:N_train)
##  log_lik[i] = bernoulli_logit_lpmf(y_train[i]|beta_0[ll_train[i]] + X_train[i] * beta[ll_train[i]]);
##  }
```