Variable Selection Techniques for High-Dimensional Simulated Data & Real Data (Bike Rentals)

Prince Kofi Asare and Shinjon Ghosh

May 2, 2025

Abstract

In the realm of high-dimensional data analysis, the identification of truly relevant variables is paramount for developing parsimonious and interpretable predictive models. This project investigates and compares the efficacy of several variable selection techniques, including Poisson Generalized Linear Models (GLM), LASSO, Elastic Net, Bayesian Regression with Normal and Laplace priors, Random Forest, and XGBoost. The study employs both meticulously controlled simulated datasets and a real-world bike rental dataset to evaluate the performance of these methods using relevant evaluation metrics. The findings will provide valuable insights into the strengths and weaknesses of each technique in different data contexts, ultimately contributing to more informed variable selection practices in high-dimensional regression problems.

1 Introduction

The proliferation of high-dimensional datasets across various domains, from genomics and finance to social sciences and environmental modeling, presents unprecedented opportunities and significant analytical challenges. A key challenge lies in many potentially irrelevant or redundant variables, which can obscure the underlying relationships, lead to overfitting, and hinder the interpretability of predictive models (Fan & Lv, 2010). Consequently, the task of variable selection, aimed at identifying the subset of truly influential predictors, has become an indispensable step in the data analysis pipeline. Effective variable selection enhances model parsimony and interpretability and improves prediction accuracy and computational efficiency.

This project addresses the critical need for robust variable selection methodologies in high-dimensional settings. Focusing on the project title "Variable Selection Techniques for High-Dimensional Simulated Data & Real Data (Bike Rentals)," this research explores and compares several prominent variable selection techniques. The study strategically leverages the complementary strengths of simulated data and a real-world bike rental dataset. Simulated data offers a controlled environment where the authentic underlying relationships are known, allowing for a rigorous assessment of a method's ability to correctly identify relevant and irrelevant variables, as defined by the True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) concepts you've highlighted. The associated metrics, such as True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR), will serve as crucial quantitative measures for evaluating the performance of each technique.

Complementing the insights gained from simulated data, the project further investigates these variable selection methods' practical applicability and performance on a real-world dataset concerning bike rental counts. Analyzing this "Bike Rentals" dataset will provide valuable insights into how these techniques perform in real-world complexities such as noise, multicollinearity, and potentially non-linear relationships. The discrepancy between findings on simulated and real data can illuminate the robustness and generalizability of each variable selection approach.

2 Objectives

The project will systematically investigate the following variable selection methods:

1. Poisson Generalized Linear Model (GLM): Particularly relevant for count data such as bike rentals, Poisson GLMs can inherently perform variable selection through techniques like stepwise selection or by examining the significance of coefficients.

2. LASSO (Least Absolute Shrinkage and Selection Operator): A popular regularization technique that adds an L1 penalty to the linear regression coefficients, forcing some coefficients to become exactly zero, thus performing variable selection.

3. Elastic Net: Combining the L1 and L2 penalties of LASSO and Ridge regression, Elastic Net offers a balance between variable selection and handling multi-

collinearity, often outperforming LASSO when dealing with groups of correlated variables.

4. Bayesian Regression with normal prior (Mean 0 and Variance 1): The choice of prior distribution can influence variable selection within a Bayesian framework. A normal prior with mean 0 tends to shrink coefficients towards zero, with the variance controlling the degree of shrinkage.

5. Bayesian Regression with Laplace priors: The Laplace prior, due to its peakedness at zero and heavy tails, encourages sparsity in the posterior distribution of coefficients, effectively performing variable selection similar to LASSO.

6. Random Forest: This ensemble learning method can assess variable importance based on how much each feature contributes to reducing impurity across all trees, providing a non-linear approach to variable selection.

7. XGBoost (Extreme Gradient Boosting): Another powerful ensemble method that builds trees sequentially, XGBoost also provides measures of feature importance that can be used for variable selection.

3 Methodology: Simulated Data Generation

This section details the process of generating the high-dimensional simulated datasets used to evaluate the variable selection techniques. The simulation is designed to create data with a known underlying structure, directly assessing each technique's ability to identify relevant variables.

3.1 Design Matrix (X) Generation:

For the simulated data component of this study, we generated a design matrix $\mathbf{X}$ with dimensions $n = 100$ observations and $p = 100$ predictor variables, establishing a high-dimensional scenario where the number of variables equals the number of observations. Each of the $p = 100$ variables within the design matrix $\mathbf{X}$ was independently sampled from a standard normal distribution, denoted as $X_{ij} \sim \mathcal{N}(0, 1)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$. To assess the stability and robustness of the variable selection techniques, generating the 100×100 design matrix was replicated 30 times, allowing for evaluation across different instantiations of the simulated data.

3.2 Response Variable (Y) Generation:

Three different response variables (Y_1, Y_2, and Y_3) will be generated based on the design matrix $\mathbf{X}$ according to the following equations:

- **Response Variable Y_1 (Linear Model with Sparse Effects):**

  $$Y_1 = 0.25 + 0.5X_1 + 0.1X_{20} + 0.2X_{40} + 0.7X_{60} + 1.2X_{80} + 1.4X_{90} + \epsilon_1$$

  where ϵ_1 represents random noise drawn from a normal distribution with a mean of 0 and a specified standard deviation (e.g., $\epsilon_1 \sim \mathcal{N}(0, \sigma_1^2)$). This model includes a sparse set of truly relevant variables: $X_1, X_{20}, X_{40}, X_{60}, X_{80}$, and X_{90}. The remaining 94 variables are irrelevant.

- **Response Variable Y_2 (Linear Model with Interaction and Dependence):**

  $$Y_2 = Y_1 + X_1 \cdot X_{40} - 1.7X_{60} \cdot X_{90} + \epsilon_2$$

  where ϵ_2 represents random noise (e.g., $\epsilon_2 \sim \mathcal{N}(0, \sigma_2^2)$). This model introduces interaction terms between some of the relevant variables from Y_1, as well as a dependence on the previously generated Y_1. This adds complexity to the variable selection task.

- **Response Variable Y_3 (Non-linear Model):**

  $$Y_3 = Y_2 + 0.20X_{40}^2 + \epsilon_3$$

  where ϵ_3 represents random noise (e.g., $\epsilon_3 \sim \mathcal{N}(0, \sigma_3^2)$). This model introduces a non-linear relationship with one of the predictor variables (X_{40}), challenging the linear assumptions of some variable selection techniques.

For each of the 30 simulated design matrices, all three response variables (Y_1, Y_2, Y_3) will be generated. This will result in a total of $30 \times 3 = 90$ unique simulated datasets (combinations of design matrix and response variable).

3.3 Noise Implementation:

For each generated response variable (Y_1, Y_2, Y_3), an independent noise term ($\epsilon_1, \epsilon_2, \epsilon_3$) was introduced. These noise terms were sampled from a normal distribution with a mean of 0 and a standard deviation ($\sigma_1, \sigma_2, \sigma_3$) set to 1 for all response variables in this study. This specific choice of standard deviation aimed to control the signal-to-noise ratio within each simulated dataset. While different noise levels could be explored in future work to further assess the robustness of the variable selection techniques under varying degrees of noise, a consistent standard deviation of 1 was maintained across all simulations for the present analysis.

3.4 Utilization in Variable Selection:

These 90 simulated datasets will serve as the primary data source for evaluating the performance of the seven variable selection techniques outlined in the introduction (Poisson GLM, LASSO, Elastic Net, Bayesian Regression with typical prior, Bayesian Regression with Laplace priors, Random Forest, and XGBoost). Each variable selection technique will be applied to each simulated data set to identify the set of relevant predictors. The performance of each method will then be assessed by comparing the selected variables to the known actual pertinent variables of each simulation using the evaluation metrics (TPR, FPR, TNR, FNR) defined earlier.

This structured simulation methodology will allow for a systematic and controlled evaluation of variable selection techniques under different linearity, sparsity, interaction effects, and non-linear relationships scenarios. The 30 replications for each response model will provide insights into the variability of the techniques' performance due to random data generation.

4 Simulation Results: Poisson GLM vs Lasso Vs Elastic Net

Table 1 below presents the average performance metrics of three variable selection techniques (Elastic Net, GLM, and LASSO) across 30 simulations for each response variable (Y_1, Y_2, and Y_3). The metrics include True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN), False Positive Rate (FPR), and False Negative Rate (FNR).

Response	Model	TP	FP	FN	TN	FPR	FNR
Y_1	ElasticNet	5.87	41.9	0.13	52.1	0.446	0.0222
Y_1	GLM	6.00	94.0	0.00	0.00	1.000	0.0000
Y_1	LASSO	5.90	23.3	0.10	70.7	0.248	0.0167
Y_2	ElasticNet	3.53	31.2	2.47	62.8	0.332	0.4110
Y_2	GLM	6.00	94.0	0.00	0.00	1.000	0.0000
Y_2	LASSO	3.70	27.7	2.30	66.3	0.294	0.3830
Y_3	ElasticNet	3.43	25.7	2.57	68.3	0.273	0.4280
Y_3	GLM	6.00	94.0	0.00	0.00	1.000	0.0000
Y_3	LASSO	3.67	24.5	2.33	69.5	0.261	0.3890

Table 1: **Poisson GLM vs Lasso Vs Elastic Net**

4.1 Performance comparison: Poisson GLM vs Lasso Vs Elastic Net

The Poisson GLM consistently selects all 94 irrelevant variables as positive (FP = 94) and identifies all the true positive variables (FN = 0), resulting in a perfect false positive rate (FPR) of 1.000 and a false negative rate (FNR) of 0.000, which indicates it is not a suitable mechanism for variable selection in these simulations. In contrast, LASSO and Elastic Net demonstrate more balanced performance, identifying some true positives while also incurring false positives and negatives, with their effectiveness influenced by the complexity of the data-generating process. For Y_1 (a linear model with sparse effects), both LASSO and Elastic Net perform relatively well, with LASSO achieving a lower FPR (0.248) compared to Elastic Net (0.446), and both methods showing low FNRs (0.0167 and 0.0222, respectively). In the case of Y_2 (a linear model with interaction and dependence), performance declines for both methods, with fewer true positives detected and higher FNRs, suggesting that interactions and dependencies complicate variable selection. For Y_3 (a non-linear model), the performance remains similar to that on Y_2, indicating that the introduction of a non-linear term does not significantly affect the variable selection capabilities of LASSO and Elastic Net in this context.

The Poisson GLM performs poorly in variable selection, consistently selecting all irrelevant variables and yielding a perfect but misleading classification profile. In contrast, LASSO and Elastic Net offer a more balanced approach, effectively identifying relevant variables in simpler linear settings like Y_1. However, their performance declines in more complex scenarios involving interactions (Y_2) or non-linear effects (Y_3), with increased false negatives and reduced true positive rates. This highlights the sensitivity of regularized regression methods to the structure of the underlying data-generating process and suggests their limitations in handling model complexity.

5 Simulation Results: Random Forest and XGBoost

Table 2 below presents the performance metrics of Random Forest and XGBoost across 30 simulations for each response variable (Y_1, Y_2, and Y_3). The metrics include True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN), True Positive Rate (TPR), and True Negative Rate (TNR).

Response	Model	TP	FP	FN	TN	TPR	TNR
Y_1	RandomForest	2.93	7.07	3.07	86.9	0.489	0.925
Y_1	XGBoost	3.10	6.90	2.90	87.1	0.517	0.927
Y_2	RandomForest	1.80	8.20	4.20	85.8	0.300	0.913
Y_2	XGBoost	2.97	7.03	3.03	87.0	0.494	0.925
Y_3	RandomForest	1.77	8.23	4.23	85.8	0.294	0.912
Y_3	XGBoost	3.13	6.87	2.87	87.1	0.522	0.927

Table 2: **Random Forest vs XGBoost**

5.1 Performance Comparison: Random Forest and XGBoost

Random Forest and XGBoost achieve moderate true positive rates (TPR) and high true negative rates (TNR) on Y_1, a linear model with sparse effects. XGBoost performs slightly better, with a TPR of 0.517 and TNR of 0.927, compared to Random Forest's TPR of 0.489 and TNR of 0.925, and both maintain relatively low false positive rates. For Y_2, a linear model with interaction and dependence, the TPR for both models declines, indicating challenges in identifying relevant variables under complex dependencies. XGBoost (TPR = 0.494) continues outperforming Random Forest (TPR = 0.300), while both retain high TNR. In the Y_3 scenario, representing a non-linear model, Random Forest's TPR decreases slightly to 0.294. At the same time, XGBoost's TPR improves to 0.522, suggesting that XGBoost may be more effective at capturing non-linear patterns relevant for variable selection. Across all settings, both models demonstrate conservative behavior in selecting irrelevant variables, achieving high TNR and low false positives. However, they identify fewer true positives than regularized regression methods like LASSO and Elastic Net. Overall, XGBoost consistently outperforms Random Forest in these simulations and appears more adept at handling non-linear relationships. However, increasing complexity in the response variable negatively impacts the performance of both models.

While tree-based methods like Random Forest and XGBoost effectively limit false positives, their ability to recover true signals declines with increasing model complexity. XGBoost consistently performs better than Random Forest and shows greater flexibility in handling linear and non-linear structures in variable selection tasks.

6 Simulation Results: Bayesian Regression with Normal and Laplace Priors

Table 3 below presents the average performance metrics of Bayesian Regression with Normal and Laplace priors across 30 simulations for each of the three response variables (Y_1, Y_2, and Y_3). The metrics include Mean True Positives (Mean TP), Mean False Positives (Mean FP), Mean False Negatives (Mean FN), Mean True Negatives (Mean TN), Mean True Positive Rate (Mean TPR), and Mean True Negative Rate (Mean TNR).

Response	Model	Mean TP	Mean FP	Mean FN	Mean TN	Mean TPR	Mean TNR
Y_1	NormalPrior	0.300	3.10	4.70	91.90	0.060	0.967
Y_1	LaplacePrior	0.367	2.97	4.63	92.00	0.073	0.969
Y_2	NormalPrior	0.700	10.00	4.30	85.00	0.140	0.895
Y_2	LaplacePrior	0.600	8.70	4.40	86.30	0.120	0.908
Y_3	NormalPrior	0.833	12.40	4.17	82.60	0.167	0.870
Y_3	LaplacePrior	0.767	11.30	4.23	83.70	0.153	0.881

Table 3: Bayesian Regression with Normal and Laplace Priors

6.1 Performance Comparison: Bayesian Regression with Normal and Laplace Priors

In the case of Y_1, a linear model with sparse effects, the Laplace prior demonstrates a slight advantage with a higher mean true positive rate (TPR = 0.073) compared to the Normal prior (TPR = 0.060). In contrast, both priors yield high mean true negative rates (TNR $\approx$ 0.968) and low mean false positives (around 3). For Y_2, which includes interaction and dependence, the Normal prior performs marginally better in terms of mean TPR (0.140 vs. 0.120), though both models maintain high TNRs $\approx$ 0.90) with increased false positives relative to Y_1. In the non-linear model Y_3, the Normal prior again shows a slightly better mean TPR (0.167) than the Laplace prior (0.153), although TNRs decline further to around 0.87–0.88 and false positives increase. Overall, both Bayesian models are conservative in variable selection, achieving high TNRs but low TPRs, particularly as the complexity of the response variable increases. The Laplace prior is more effective in simpler settings, while the Normal prior has a slight edge in more complex models.

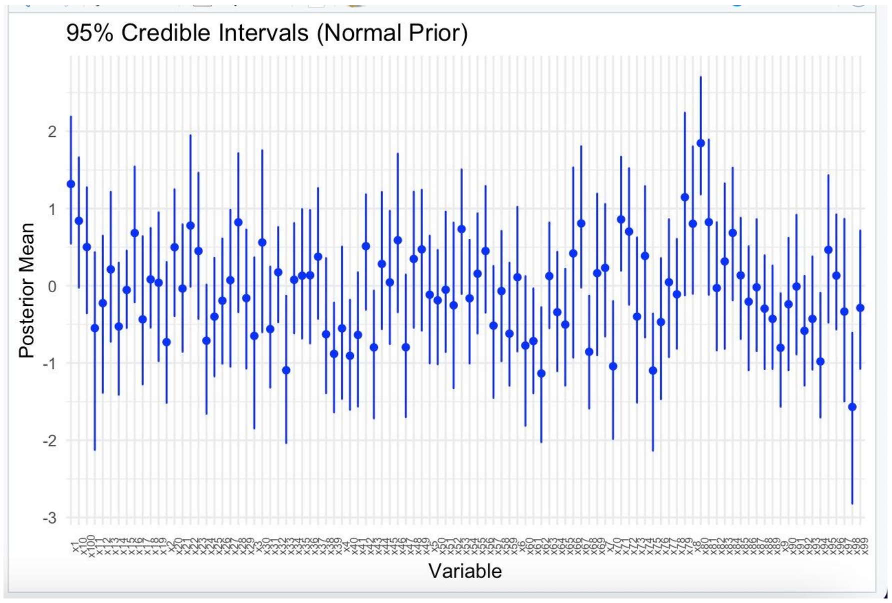


Figure 1: Bayesian Regression with Normal Prior

6.2 Credible Interval Comparison: Normal Prior

From Figure 1 above, variables with shorter intervals are estimated more precisely, while wider intervals indicate more uncertainty. Moreover, Credible intervals that do not cross the zero line (i.e., the horizontal axis at y = 0) suggest more substantial evidence of a non-zero effect. According to Figure 3, most intervals cross zero, indicating that the corresponding variables may not be significantly associated with the response variable under the Normal prior. A few variables (e.g., around the left-most and mid-right sections of the plot) have intervals entirely above or below zero, suggesting potentially important predictors that are positively and negatively correlated. In addition, using a Normal prior tends to **regularize** coefficients toward zero but does not enforce sparsity on a large scale.

6.3 Credible Interval Comparison: Laplace Prior

From Figure 2 below, many credible intervals include zero, indicating that many variables are likely insignificant under the prior Laplace. A smaller subset of variables has credible intervals that do not overlap with zero, highlighting a few potentially essential predictors with more substantial evidence of association with the response variable. From the analysis of Figures 3 and 4, we illustrate that the credible intervals under the Laplace prior are generally tighter, with more zero-centered coefficients, suggesting more confident identification of relevant vs. irrelevant variables.

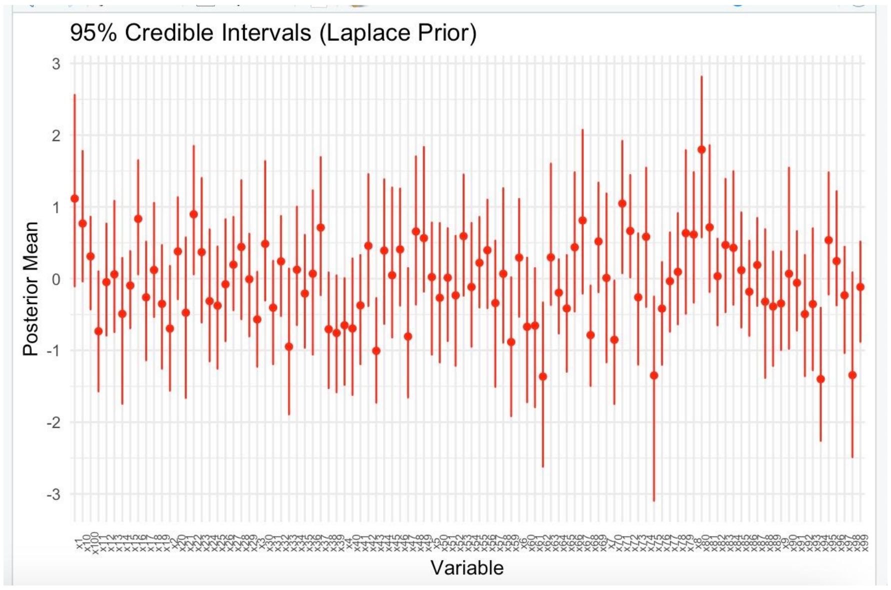


Figure 2: Bayesian Regression with Laplace Prior

Bayesian regression models with Normal and Laplace priors show strong regularization behavior, yielding low true positive rates and high true negative rates across all scenarios. The Laplace prior is more advantageous in sparse linear settings, whereas the Normal prior proves slightly more effective under complex interactions or non-linear structures. Both models, however, tend to underselect relevant variables compared to more flexible or less conservative methods.

7 Methodology: Real Data Analysis Bike Rentals

This section outlines the methodology for analyzing the real-world bike rental dataset to evaluate the performance of the selected variable selection techniques in a practical application.

7.1 Data Source and Description:

The dataset used in this analysis is sourced from **Capital Bikeshare** in Washington D.C., covering the years 2011 and 2012. It is a secondary dataset that is publicly available for analytical use. Additional weather-related variables—including temperature, humidity, and windspeed—will be integrated from **FreeMeteo** to enhance the feature set. The dataset contains **731 daily records**, each consisting of **16 variables**. The level of temporal aggregation (hourly or daily) will be clarified during the initial data exploration phase.

2. Key Variables:

- **Response Variable (Dependent Variable):**
  - `cnt`: Total number of bike rentals per day.
- **Predictor Variables (Independent Variables):**
  - **Temporal Variables:**
    - `Season`: Categorical (spring, summer, autumn, winter).
    - `Holiday`: Binary (whether the day was a holiday).
    - `Working Day`: Binary (whether the day was a working day).
  - **Weather Variables:**
    - `Temp (℃)`: Temperature in Celsius (numerical).
    - `Humidity (%)`: Relative humidity (numerical).
    - `Windspeed (m/s)`: Wind speed in meters per second (numerical).

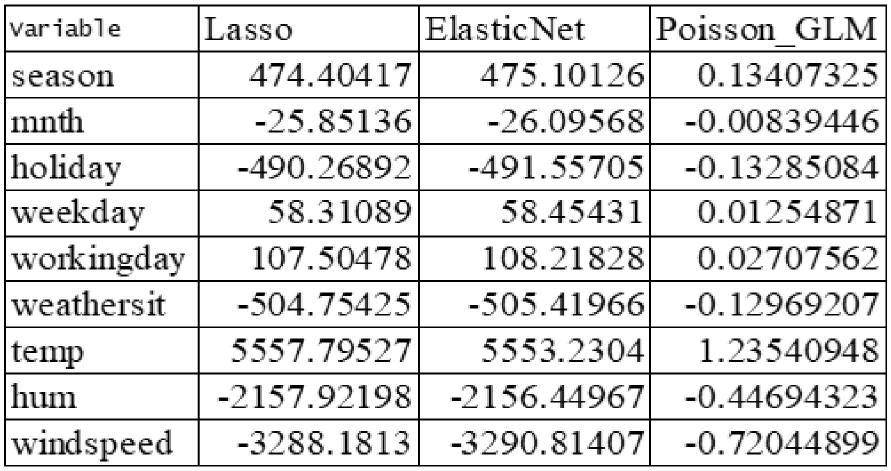
8 Real Data Analysis: Poisson GLM

Coefficients	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	8.014965	0.004142	1935.05	<2e-16 ***
season	0.134073	0.001106	121.19	<2e-16 ***
mnth	-0.00839	0.00036	-23.3	<2e-16 ***
holiday	-0.13285	0.003738	-35.54	<2e-16 ***
weekday	0.012549	0.00028	44.8	<2e-16 ***
workingday	0.027076	0.001237	21.89	<2e-16 ***
weathersit	-0.12969	0.001388	-93.42	<2e-16 ***
temp	1.23541	0.0033	374.37	<2e-16 ***
hum	-0.44694	0.005425	-82.38	<2e-16 ***
windspeed	-0.72045	0.008076	-89.21	<2e-16 ***

Table 4: Poisson GLM Summary

From table 4 above it can be seen that all predictor variables are statistically significant based on their p-values. Specifically, for each one-unit increase in **season**, the expected number of bike rentals increases by a log of 0.1340, with a highly significant p-value (<2e-16), indicating a strong effect. Variables such as **weekday**, **workingday**, and **temp** also contribute positively to bike rentals, with respective log increases of 0.0125, 0.0270, and 1.2354, all statistically significant. In contrast, a one-unit increase in **mnth** leads to a slight decrease in expected bike rentals by a log of 0.0083. Furthermore, increases in **holiday**, **weathersit**, **hum**, and **windspeed** are associated with decreases in bike rentals

8.1 Real Data Analysis: LASSO vs Elastic Net



variable	Lasso	ElasticNet	Poisson_GLM
season	474.40417	475.10126	0.13407325
mnth	-25.85136	-26.09568	-0.00839446
holiday	-490.26892	-491.55705	-0.13285084
weekday	58.31089	58.45431	0.01254871
workingday	107.50478	108.21828	0.02707562
weathersit	-504.75425	-505.41966	-0.12969207
temp	5557.79527	5553.2304	1.23540948
hum	-2157.92198	-2156.44967	-0.44694323
windspeed	-3288.1813	-3290.81407	-0.72044899

Figure 3: LASSO vs Elastic Net

From Figure 3 above, both LASSO and Elastic Net identified key predictors of bike rentals, selecting variables such as **season**, **holiday**, **weathersit**, **temp**, **hum**, and **windspeed**, which consistently showed strong predictive influence. In the LASSO model, positive coefficients for **season** (474.404) and **temp** (5557.795) suggest these variables contribute to increased bike rentals. In contrast, negative coefficients for **holiday** (-490.2689), **weathersit** (-504.752), **hum** (-2157.9219), and **windspeed** (-3288.1813) indicate reductions in rentals. Similarly, the Elastic Net model produced nearly identical results, with positive effects from **season** (475.101) and **temp** (5553.230), and negative effects from **holiday** (-491.557), **weathersit** (-505.419), **hum** (-2156.449), and **windspeed** (-3290.814). In both models, variables such as **mnth** and **weekday** were likely excluded due to coefficient shrinkage, reflecting their comparatively weaker contribution to predicting bike rental counts.

8.2 Real Data Analysis: Random Forest vs XGBoost

From Figure 4 below, which is the Random Forest variable importance plot, **temperature** and **humidity** have the greatest impact on predicting bike rentals. Additionally, variables such as **season**, **month**, **weather situation**, and **windspeed** demonstrate moderate influence. In contrast, **working day**, **holiday**, and **weekday** show minimal contributions to model performance, suggesting they may act as noise in the prediction.

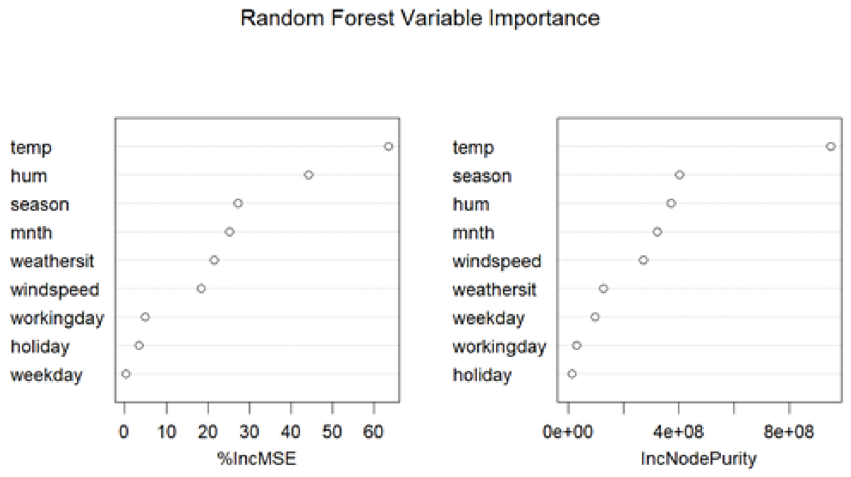


Figure 4: Random Forest Variable Importance Plot

From Figure 5 below, which is the XGBoost feature importance plot, **temperature** and **humidity** are the most influential predictors of bike rentals. Variables such as **season**, **month**, **windspeed**, and **weekday** also demonstrate moderate predictive power. Meanwhile, **working day**, **holiday**, and **weathersit** contribute the least, indicating they may add noise to the prediction process.

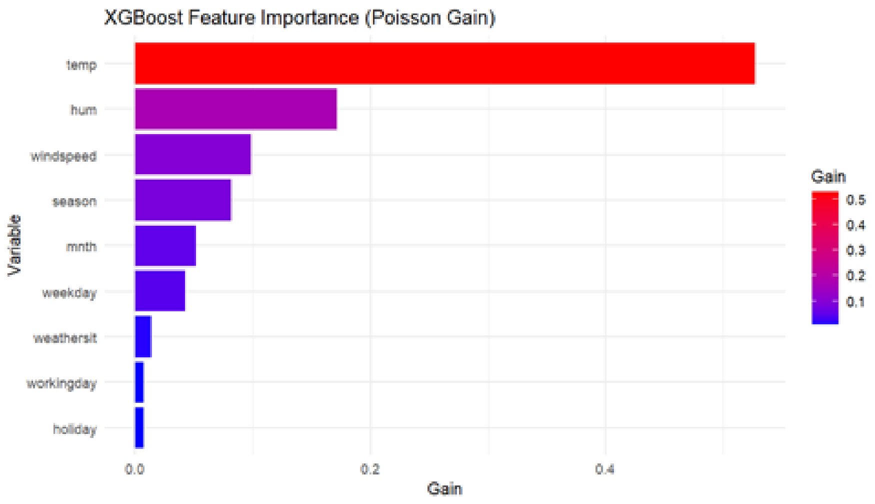


Figure 5: XGBoost Feature Importance Plot

8.3 Real Data Analysis: Bayesian Regression with Normal Prior

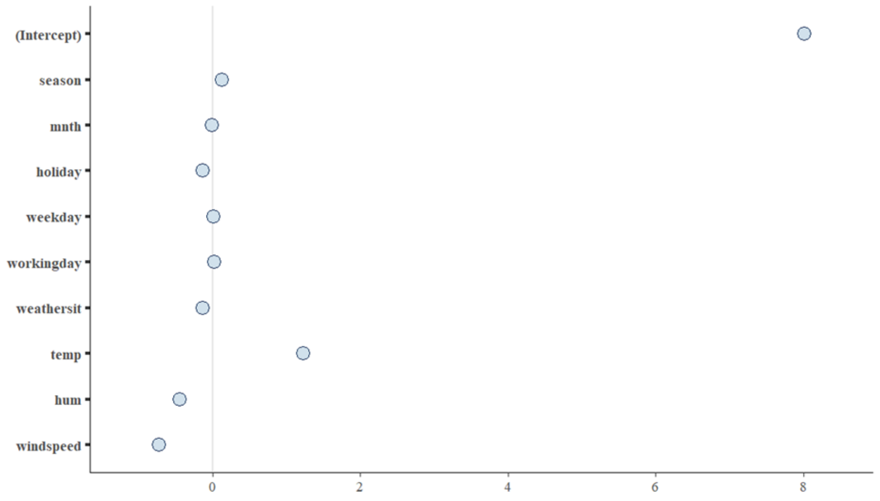


Figure 6: 95% Credible Intervals Under Normal Prior

From Figure 6 above, temp is the most influential predictor among the explanatory variables, with a substantial positive coefficient, suggesting that as temperature increases, the response variable bike rentals also increases. On the other hand, windspeed and hum suggest decreasing bike rentals during the period. Other predictors like season, mnth, holiday, weekday, workingday, and weathersit have coefficients very close to zero, indicating little to no influence on the response variable in this model.

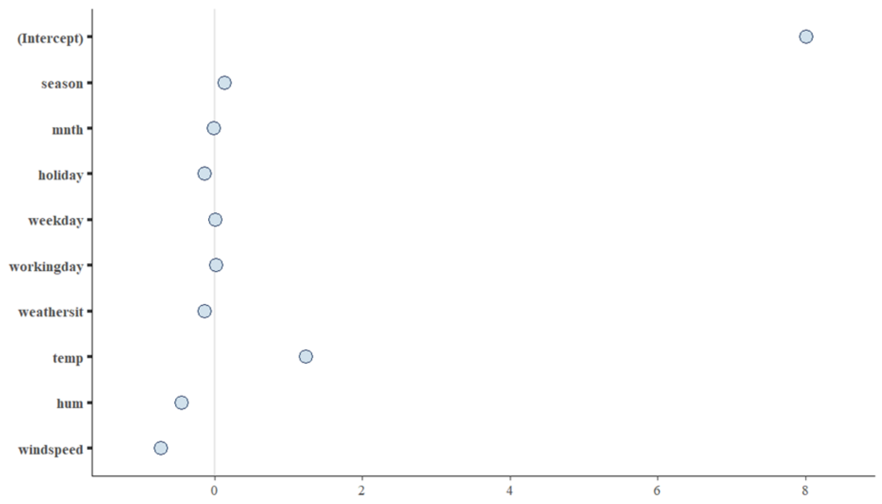


Figure 7: 95% Credible Intervals Under Laplace Prior

8.4 Real Data Analysis: Bayesian Regression with Laplace Prior

Figure 7 above indicates that temp is the most influential predictor among the explanatory variables, showing a substantial positive coefficient—implying that bike rentals increase as temperature rises. In contrast, windspeed and hum exhibit negative coefficients, suggesting a decrease in bike rentals with higher values of these variables. Meanwhile, other predictors such as season, mnth, holiday, weekday, working day, and weathersit have coefficients near zero, indicating minimal or no impact on the response variable in this model.

9 Conclusion

This project successfully compared several variable selection techniques using both controlled simulated data and a real-world application. The results highlight the varying strengths and weaknesses of each method depending on the underlying data structure and the complexity of the relationships between variables. While LASSO, XGBoost and Bayesian Regression with Laplace Prior demonstrated strong performance in the simulated data, the real data analysis provided valuable insights into the factors driving bike rental demand, with consistency observed across different modeling approaches regarding the importance and direction of effect of key variables like temperature, humidity, windspeed, and seasonality. These findings contribute to a better understanding of variable selection methodologies and their practical application in analyzing real-world datasets.

9.1 Limitations of Selected Model

While prior studies have identified different Poisson models, which provide different significant variables in predicting bike rentals, this study highlights the Poisson models with interaction terms between weather situation and temperature, and windspeed emerged as the optimal choice. These discrepancies may stem from several methodological differences. Past research might have employed:

1. Varying datasets: Different locations, sample sizes, and data collection periods can lead to diverse findings.
2. Divergent statistical methods: Statistical approaches can influence the identified significant variables.
3. Incomplete variable sets: The availability of different variables within each dataset can impact the model's results.

Acknowledging that our model's generalizability may be limited to the specific location from which the dataset was obtained is crucial. Additionally, the model did not incorporate sudden weather changes, special events, and potential non-linear relationships between variables. These factors could result in different model selections. Future research that accounts for these broader considerations can provide a more comprehensive understanding of bike rental demand and improve model accuracy.

10 Author Contribution

Shinjon Ghosh and Prince Kofi jointly conducted this project's simulation and real data analyses. For the presentation component, Shinjon prepared slides related to the simulation data analysis, while Prince was responsible for the fundamental data analysis slides. Prince authored the written report's abstract, introduction, conclusion, and simulation data analysis results. Shinjon contributed by writing the real data analysis results and compiling the appendix section.

11 Appendix

Appendix

```
R Code:
## Simulate Data
setwd("F:/Applied Statistics Course - ISU/Advanced Regression Analysis/Project/Project

library(MASS)
library(Matrix)
library(tidyverse)

set.seed(123)

n <- 200        # observations per dataset
p <- 100        # number of predictors
num_datasets <- 30
true_vars <- c(1, 20, 40, 60, 80, 90)

# Initialize container for all datasets
full_data <- data.frame()

for (i in 1:num_datasets) {
  # Simulate design matrix X
  X <- matrix(rnorm(n * p), nrow = n, ncol = p)
  colnames(X) <- paste0("x", 1:p)

  # --- y1: Sparse linear ---
  mu1 <- 0.25 + 0.5*X[,1] + 0.1*X[,20] + 0.2*X[,40] + 0.7*X[,60] + 1.2*X[,80] + 1.4*X[
  y1 <- rpois(n, lambda = exp(mu1))

  # --- y2: Add interaction ---
  mu2 <- mu1 + X[,1]*X[,40] - 1.7 * X[,60]*X[,90]
  y2 <- rpois(n, lambda = exp(mu2))

  # --- y3: Add nonlinearity ---
  mu3 <- mu2 + 0.2 * X[,40]^2
  y3 <- rpois(n, lambda = exp(mu3))

  # Combine into a dataset: y1, y2, y3, x1 to x100
  dataset <- data.frame(y1 = y1, y2 = y2, y3 = y3, X)

  full_data <- rbind(full_data, dataset)
}

# Save the full dataset
write.csv(full_data, file = "Data/simulated_poisson_6000rows.csv", row.names = FALSE)
saveRDS(full_data, file = "Data/simulated_poisson_6000rows.rds")
```

```
##Simulation Study
##Poisson GLM vs Lasso vs Elastic Net
library(glmnet)
library(tidyverse)

‘%nin%‘ <- Negate(‘%in%‘)

# True variable indices
true_vars_list <- list(
  y1 = c(1, 20, 40, 60, 80, 90),
  y2 = c(1, 20, 40, 60, 80, 90),
  y3 = c(1, 20, 40, 60, 80, 90)
)

# Your previously generated data
data <- full_data

results_all <- data.frame()

for (i in 0:29) {
  for (target in c("y1", "y2", "y3")) {
    start_idx <- i * 200 + 1
    end_idx <- start_idx + 199

    y <- data[[target]][start_idx:end_idx]
    X_raw <- data[start_idx:end_idx, 4:103]

    # ---- Clean X ----
    X <- as.matrix(X_raw)

    # Remove columns with NA, NaN, Inf, or zero variance
    valid_cols <- apply(X, 2, function(col) {
      all(is.finite(col)) && sd(col) > 0
    })
    X <- X[, valid_cols]

    true_vars <- true_vars_list[[target]]

    # ---- POISSON GLM ----
    tryCatch({
      mm <- model.matrix(~ ., data = as.data.frame(X))
      mm <- mm[, apply(mm, 2, function(col) all(is.finite(col)) && sd(col) > 0)]
      fit_glm <- glm(y ~ mm - 1, family = poisson)  # remove intercept since included
      selected_glm <- which(coef(fit_glm) != 0)
      selected_glm_names <- names(coef(fit_glm))[selected_glm]
      selected_glm_idx <- as.integer(gsub("x", "", gsub("mm", "", selected_glm_names))
      selected_glm_idx <- selected_glm_idx[!is.na(selected_glm_idx)]
    }, error = function(e) {
      selected_glm_idx <- integer(0)  # in case glm fails
    })

    # ---- LASSO ----
    fit_lasso <- cv.glmnet(X, y, alpha = 1, family = "poisson")
    selected_lasso <- which(coef(fit_lasso, s = "lambda.min")[-1] != 0)

    # ---- ELASTIC NET ----
    fit_enet <- cv.glmnet(X, y, alpha = 0.5, family = "poisson")
    selected_enet <- which(coef(fit_enet, s = "lambda.min")[-1] != 0)

    # --- Evaluation helper
    eval_selection <- function(selected, true_vars) {
      TP <- sum(true_vars %in% selected)
      FP <- sum(!(1:100 %in% true_vars) & (1:100 %in% selected))
      FN <- sum(true_vars %nin% selected)
      TN <- sum(!(1:100 %in% true_vars) & !(1:100 %in% selected))
      FPR <- if ((FP + TN) == 0) NA else FP / (FP + TN)
      FNR <- if ((FN + TP) == 0) NA else FN / (FN + TP)
      return(data.frame(TP, FP, FN, TN, FPR, FNR))
    }

    results_all <- rbind(
      results_all,
      cbind(eval_selection(selected_glm_idx, true_vars), response = target, dataset =
      cbind(eval_selection(selected_lasso, true_vars), response = target, dataset = i+
      cbind(eval_selection(selected_enet, true_vars), response = target, dataset = i+1
    )
  }
}

# -------- Summary Table --------
summary_table <- results_all %>%
  group_by(response, model) %>%
  summarise(across(c(TP, FP, FN, TN, FPR, FNR), mean), .groups = "drop")

print(summary_table)

# -------- Boxplots --------
ggplot(results_all, aes(x = model, y = FNR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~response) +
  labs(title = "False Negative Rate by Model and Response", y = "FNR") +
  theme_minimal()
```

```
ggplot(results_all, aes(x = model, y = FPR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~response) +
  labs(title = "False Positive Rate by Model and Response", y = "FPR") +
  theme_minimal()

## Random Forest vs XGBoost
setwd("F:/Applied Statistics Course - ISU/Advanced Regression Analysis/Project/Project

library(randomForest)
library(xgboost)
library(dplyr)
library(ggplot2)

‘%nin%‘ <- Negate(‘%in%‘)

# Define the true variable indices for each target
true_vars_list <- list(
  y1 = c(1, 20, 40, 60, 80, 90),
  y2 = c(1, 20, 40, 60, 80, 90),
  y3 = c(1, 20, 40, 60, 80, 90))
data <- read.csv(file = "Data/simulated_poisson_6000rows.csv")

results_all <- data.frame()

# Loop through 30 datasets per each of y1, y2, y3 (total: 90)
for (i in 0:29) {
  for (target in c("y1", "y2", "y3")) {

    dataset_label <- paste0(target, "_dataset", i + 1)
    start_idx <- i * 200 + 1
    end_idx <- start_idx + 99

    y <- data[[target]][start_idx:end_idx]
    X <- data[start_idx:end_idx, 4:103]  # x1 to x100

    # ===== RANDOM FOREST =====
    rf_model <- randomForest(x = X, y = y, ntree = 500, importance = TRUE)
    imp_rf <- importance(rf_model)[, 1]  # %IncMSE
    selected_rf <- as.integer(sub("x", "", names(sort(imp_rf, decreasing = TRUE)[1:10]

    # Evaluation
    true_vars <- true_vars_list[[target]]
    TP_rf <- sum(true_vars %in% selected_rf)
    FP_rf <- sum(!(1:100 %in% true_vars) & (1:100 %in% selected_rf))
    FN_rf <- sum(true_vars %nin% selected_rf)
    TN_rf <- sum(!(1:100 %in% true_vars) & !(1:100 %in% selected_rf))
    TPR_rf <- if ((TP_rf + FN_rf) == 0) NA else TP_rf / (TP_rf + FN_rf)
    TNR_rf <- if ((TN_rf + FP_rf) == 0) NA else TN_rf / (TN_rf + FP_rf)

    results_all <- rbind(results_all, data.frame(
      response = target,
      dataset = i + 1,
      model = "RandomForest",
      TP = TP_rf, FP = FP_rf, FN = FN_rf, TN = TN_rf,
      TPR = TPR_rf, TNR = TNR_rf
    ))

    # ===== XGBOOST =====
    dtrain <- xgb.DMatrix(data = as.matrix(X), label = y)
    xgb_model <- xgboost(data = dtrain, objective = "reg:squarederror", nrounds = 100,
    imp_xgb <- xgb.importance(model = xgb_model)

    selected_xgb <- as.integer(sub("x", "", imp_xgb$Feature[1:10]))

    TP_xgb <- sum(true_vars %in% selected_xgb)
    FP_xgb <- sum(!(1:100 %in% true_vars) & (1:100 %in% selected_xgb))
    FN_xgb <- sum(true_vars %nin% selected_xgb)
    TN_xgb <- sum(!(1:100 %in% true_vars) & !(1:100 %in% selected_xgb))
    TPR_xgb <- if ((TP_xgb + FN_xgb) == 0) NA else TP_xgb / (TP_xgb + FN_xgb)
    TNR_xgb <- if ((TN_xgb + FP_xgb) == 0) NA else TN_xgb / (TN_xgb + FP_xgb)

    results_all <- rbind(results_all, data.frame(
      response = target,
      dataset = i + 1,
      model = "XGBoost",
      TP = TP_xgb, FP = FP_xgb, FN = FN_xgb, TN = TN_xgb,
      TPR = TPR_xgb, TNR = TNR_xgb
    ))
  }
}


results_all_summary <- results_all %>%
  group_by(response, model) %>%
  summarise(across(c(TP, FP, FN, TN, TPR, TNR), mean), .groups = "drop")

print(results_all_summary)

# Boxplots
# pdf(file ="Figure/box_TPR_RF_XGB.pdf")
# op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(results_all, aes(x = model, y = TPR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~ response) +
  #labs(title = "True Positive Rate by Response", y = "TPR") +
  theme_minimal()+
  theme(legend.position = "none")
# par(op)
# dev.off()

# pdf(file ="Figure/box_TNR_RF_XGB.pdf")
# op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(results_all, aes(x = model, y = TNR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~ response) +
  #labs(title = "True Negative Rate by Response", y = "TNR") +
  theme_minimal()+
  theme(legend.position = "none")
# par(op)
# dev.off()
#
library(reshape2)
library(ggplot2)

# Initialize frequency storage: [variable, response, model]
var_freq <- array(0, dim = c(100, 3, 2),
                  dimnames = list(
                    paste0("x", 1:100),
                    c("y1", "y2", "y3"),
                    c("RandomForest", "XGBoost")
                  ))

# Re-run selection collection only (no evaluation)
for (i in 0:29) {
  for (target in c("y1", "y2", "y3")) {
    start_idx <- i * 200 + 1
    end_idx <- start_idx + 99
    y <- data[[target]][start_idx:end_idx]
    X <- data[start_idx:end_idx, 4:103]

    # Random Forest
    rf_model <- randomForest(x = X, y = y, ntree = 500, importance = TRUE)
    imp_rf <- importance(rf_model)[, 1]
    selected_rf <- names(sort(imp_rf, decreasing = TRUE))[1:10]
    var_freq[selected_rf, target, "RandomForest"] <- var_freq[selected_rf, target, "Ra

    # XGBoost
    dtrain <- xgb.DMatrix(data = as.matrix(X), label = y)
    xgb_model <- xgboost(data = dtrain, objective = "reg:squarederror", nrounds = 100,
    imp_xgb <- xgb.importance(model = xgb_model)
    selected_xgb <- imp_xgb$Feature[1:min(10, nrow(imp_xgb))]  # take top 10 or fewer

    # Keep only features that are present in var_freq row names
    selected_xgb_valid <- selected_xgb[selected_xgb %in% rownames(var_freq)]
    var_freq[selected_xgb_valid, target, "XGBoost"] <- var_freq[selected_xgb_valid, ta
     }
}

# Melt for heatmap
df_heat <- melt(var_freq)
colnames(df_heat) <- c("Variable", "Response", "Model", "Frequency")

df_heat_all <- df_heat

df_heat_all$Variable <- factor(df_heat_all$Variable, levels = paste0("x", 1:100))


# pdf(file ="Figure/VB_Freq_RF_XGB.pdf")
# op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(df_heat_all, aes(x = Model, y = Variable, fill = Frequency)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "red") +
  facet_wrap(~ Response) +
  scale_y_discrete(limits = rev(levels(df_heat_all$Variable))) + # Reverse the y-axis
  labs(
    title = "Selection Frequency of All Variables Across 30 Datasets",
    x = "Model",
    y = "Variable",
    fill = "Frequency"
  ) +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6))
# par(op) # dev.off()

## Bayesian Regression (Normal Prior vs Laplace Prior)
setwd("F:/Applied Statistics Course - ISU/Advanced Regression Analysis/Project/Project

data <-read.csv(file = "Data/simulated_poisson_6000rows.csv")

‘%nin%‘ <- Negate(‘%in%‘)

# Define the true variable indices for each target
true_vars_list <- list(
  y1 = c(1, 20, 40, 60, 80, 90),
  y2 = c(1, 20, 40, 60, 80, 90),
  y3 = c(1, 20, 40, 60, 80, 90)
```

```
)
# Load required libraries
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

# Stan model code for Normal prior
normal_model_code <- "
data {
  int<lower=1> n;
  int<lower=1> p;
  matrix[n, p] X;
  int<lower=0> y[n];
}
parameters {
  real alpha;
  vector[p] beta;
}
model {
  beta ~ normal(0, 1);
  alpha ~ normal(0, 5);
  y ~ poisson_log(X * beta + alpha);
}
"

# Stan model code for Laplace prior
laplace_model_code <- "
functions {
  real laplace_lpdf(real x, real mu, real b) {
    return log(0.5 / b) - fabs(x - mu) / b;
  }
}
data {
  int<lower=1> n;
  int<lower=1> p;
  matrix[n, p] X;
  int<lower=0> y[n];
}
parameters {
  real alpha;
  vector[p] beta;
}
model {
  for (j in 1:p)
    beta[j] ~ laplace(0, 1);
  alpha ~ normal(0, 5);
  y ~ poisson_log(X * beta + alpha);
```

```
}
"

# Compile Stan models once
model_normal <- stan_model(model_code = normal_model_code)
model_laplace <- stan_model(model_code = laplace_model_code)

# Initialize result containers
results_bayes <- data.frame()
selection_record <- list()

for (i in 0:29) {
  for (target in c("y1", "y2", "y3")) {

    start_idx <- i * 200 + 1
    end_idx <- start_idx + 99
    y <- data[[target]][start_idx:end_idx]
    X <- as.matrix(data[start_idx:end_idx, 4:103])
    stan_data <- list(n = nrow(X), p = ncol(X), X = X, y = y)
    true_vars <- true_vars_list[[target]]

    dataset_label <- paste0(target, "_dataset", i + 1)

    # ------------------------
    # Normal Prior
    fit_normal <- sampling(model_normal, data = stan_data, iter = 3000, chains = 2, se
    beta_draws_normal <- extract(fit_normal)$beta
    ci_normal <- apply(beta_draws_normal, 2, quantile, probs = c(0.025, 0.975))
    selected_normal <- which(ci_normal[1, ] * ci_normal[2, ] > 0)
    selected_vector_normal <- rep(0, 100)
    selected_vector_normal[selected_normal] <- 1

    # Evaluation
    TP <- sum(true_vars %in% selected_normal)
    FP <- sum(!(1:100 %in% true_vars) & (1:100 %in% selected_normal))
    FN <- sum(true_vars %in% setdiff(1:100, selected_normal))
    TN <- sum(!(1:100 %in% true_vars) & !(1:100 %in% selected_normal))
    TPR <- if ((TP + FN) == 0) NA else TP / (TP + FN)
    TNR <- if ((TN + FP) == 0) NA else TN / (TN + FP)

    results_bayes <- rbind(results_bayes, data.frame(
      response = target, dataset = i + 1, model = "NormalPrior",
      TP = TP, FP = FP, FN = FN, TN = TN, TPR = TPR, TNR = TNR
    ))

    selection_record[[paste0(dataset_label, "_Normal")]] <- selected_vector_normal
    # ------------------------
    # Laplace Prior
    fit_laplace <- sampling(model_laplace, data = stan_data, iter = 3000, chains = 2,
    beta_draws_laplace <- extract(fit_laplace)$beta
    ci_laplace <- apply(beta_draws_laplace, 2, quantile, probs = c(0.025, 0.975))
    selected_laplace <- which(ci_laplace[1, ] * ci_laplace[2, ] > 0)
    selected_vector_laplace <- rep(0, 100)
    selected_vector_laplace[selected_laplace] <- 1

    TP <- sum(true_vars %in% selected_laplace)
    FP <- sum(!(1:100 %in% true_vars) & (1:100 %in% selected_laplace))
    FN <- sum(true_vars %in% setdiff(1:100, selected_laplace))
    TN <- sum(!(1:100 %in% true_vars) & !(1:100 %in% selected_laplace))
    TPR <- if ((TP + FN) == 0) NA else TP / (TP + FN)
    TNR <- if ((TN + FP) == 0) NA else TN / (TN + FP)

    results_bayes <- rbind(results_bayes, data.frame(
      response = target, dataset = i + 1, model = "LaplacePrior",
      TP = TP, FP = FP, FN = FN, TN = TN, TPR = TPR, TNR = TNR
    ))

    selection_record[[paste0(dataset_label, "_Laplace")]] <- selected_vector_laplace

    cat(target, "_", i + 1, ": Done\n")
  }
}

# Save both objects to RDS files
saveRDS(results_bayes, file = "results_bayes.rds")
saveRDS(selection_record, file = "selection_record_bayes.rds")



library(dplyr)
library(ggplot2)
library(reshape2)

# Combine selection_record list to matrix
selection_df <- do.call(rbind, selection_record)

# Parse labels into response and model
meta_info <- do.call(rbind, strsplit(rownames(selection_df), "_"))
selection_df <- as.data.frame(selection_df)
selection_df$response <- meta_info[, 1]
selection_df$dataset <- meta_info[, 2]
selection_df$model <- meta_info[, 3]
```

```
# Gather to long format
selection_long <- melt(selection_df,
                       id.vars = c("response", "dataset", "model"),
                       variable.name = "variable",
                       value.name = "selected")

# Convert variable to numeric (e.g., x1 → 1)
selection_long$variable <- as.integer(sub("V", "", selection_long$variable))

# Count frequency for each (response, model, variable)
freq_summary <- selection_long %>%
  group_by(response, model, variable) %>%
  summarise(Frequency = sum(selected), .groups = "drop")

# For better ordering of y-axis
freq_summary$variable <- factor(
  paste0("x", freq_summary$variable),
  levels = paste0("x", rev(1:100))  # reversed for y-axis top-to-bottom
)
freq_summary$model <- factor(freq_summary$model, levels = c("Normal", "Laplace"))


pdf(file ="Figure/VB_Freq_bayes.pdf")
op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(freq_summary, aes(x = model, y = variable, fill = Frequency)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  facet_wrap(~ response) +
  labs(
    #title = "Variable Selection Frequency Across 30 Datasets",
    x = "Model",
    y = "Variable",
    fill = "Frequency"
  ) +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6))
par(op)
dev.off()

library(dplyr)

results_bayes_summary <- results_bayes %>%
  group_by(response, model) %>%
  summarise(
    Mean_TP = mean(TP),
    Mean_FP = mean(FP),
    Mean_FN = mean(FN),
    Mean_TN = mean(TN),
    Mean_TPR = mean(TPR, na.rm = TRUE),
    Mean_TNR = mean(TNR, na.rm = TRUE),
    .groups = "drop"
  )

print(results_bayes_summary)

library(ggplot2)


results_bayes$model <- factor(results_bayes$model, levels = c("NormalPrior", "LaplaceP
pdf(file ="Figure/box_TPR_bayes.pdf")
op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(results_bayes, aes(x = model, y = TPR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~ response) +
  #labs(title = "True Positive Rate (TPR) by Response and Model", y = "TPR", x = "") +
  theme_minimal() +
  theme(legend.position = "none")
par(op)
dev.off()

# TNR Boxplot
pdf(file ="Figure/box_TNR_bayes.pdf")
op<-par(mar=c(5,5,4,1),cex.axis=1.5,font.axis=2,font.lab=2,cex.main=2,cex.lab=1.6)
ggplot(results_bayes, aes(x = model, y = TNR, fill = model)) +
  geom_boxplot() +
  facet_wrap(~ response) +
  #labs(title = "True Negative Rate (TNR) by Response and Model", y = "TNR", x = "") +
  theme_minimal() +
  theme(legend.position = "none")
par(op)
dev.off()


# From your Stan fits
beta_normal <- extract(fit_normal)$beta  # matrix: draws × 100
beta_laplace <- extract(fit_laplace)$beta

# Posterior summaries
summary_normal <- apply(beta_normal, 2, function(x) c(mean = mean(x), sd = sd(x),
                                                      lower = quantile(x, 0.025),
                                                      upper = quantile(x, 0.975)))

summary_laplace <- apply(beta_laplace, 2, function(x) c(mean = mean(x), sd = sd(x),
                                                        lower = quantile(x, 0.025),
                                                        upper = quantile(x, 0.975)))

# Combine into data frame
coef_df <- data.frame(
  variable = paste0("x", 1:100),
  mean_normal = summary_normal["mean", ],
  mean_laplace = summary_laplace["mean", ],
  lower_normal = summary_normal["lower.2.5%", ],
  upper_normal = summary_normal["upper.97.5%", ],
  lower_laplace = summary_laplace["lower.2.5%", ],
  upper_laplace = summary_laplace["upper.97.5%", ]
)

library(ggplot2)
ggplot(coef_df, aes(x = mean_normal, y = mean_laplace, label = variable)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(title = "Posterior Mean Comparison",
       x = "Normal Prior", y = "Laplace Prior") +
  theme_minimal()

library(tidyr)

coef_long <- coef_df %>%
  select(variable, mean_normal, mean_laplace) %>%
  pivot_longer(cols = -variable, names_to = "model", values_to = "posterior_mean")

ggplot(coef_long, aes(x = variable, y = posterior_mean, color = model, group = model))
  geom_line() +
  labs(title = "Posterior Coefficients by Variable",
       x = "Variable", y = "Posterior Mean") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))


# Identify selected variables (CI does not include 0)
selected_normal <- with(coef_df, which(lower_normal * upper_normal > 0))
selected_laplace <- with(coef_df, which(lower_laplace * upper_laplace > 0))

intersect_vars <- intersect(selected_normal, selected_laplace)
unique_normal <- setdiff(selected_normal, selected_laplace)
unique_laplace <- setdiff(selected_laplace, selected_normal)

cat("Common selected variables:", paste0("x", intersect_vars), "\n")
cat("Only in normal:", paste0("x", unique_normal), "\n")
cat("Only in laplace:", paste0("x", unique_laplace), "\n")
```

```
## Real Data Analysis
# Load required packages
library(readxl)
library(glmnet)
library(dplyr)

# Load data
data <- read_excel("F:\\Applied Statistics Course - ISU\\Advanced Regression Analysis\

# Prepare data
X <- data %>%
  select(-instant, -dteday, -casual, -registered, -cnt, -yr, -atemp) %>%
  as.data.frame()

y <- data$cnt

# Convert to matrix for glmnet
X_matrix <- model.matrix(~ ., X)[, -1]  # remove intercept added by model.matrix
y_vector <- as.numeric(y)

# Lasso Regression
lasso_model <- cv.glmnet(X_matrix, y_vector, alpha = 1, family = "gaussian", standardi
lasso_coef <- coef(lasso_model, s = "lambda.min")

# Elastic Net Regression
enet_model <- cv.glmnet(X_matrix, y_vector, alpha = 0.5, family = "gaussian", standard
enet_coef <- coef(enet_model, s = "lambda.min")

# Poisson GLM
poisson_model <- glm(cnt ~ ., data = data %>% select(-instant, -dteday, -casual, -regi
                                                     -atemp), family = poisson())
poisson_coef <- coef(poisson_model)
summary(poisson_model)

# Combine results into a table
coef_table <- data.frame(
  Variable = rownames(lasso_coef),
  Lasso = as.numeric(lasso_coef),
  ElasticNet = as.numeric(enet_coef),
  Poisson_GLM = poisson_coef[match(rownames(lasso_coef), names(poisson_coef))]
)

# Remove intercept row for clean display
coef_table <- coef_table[coef_table$Variable != "(Intercept)", ]

# Show result
print(coef_table)
```

```
# Feature Importance
library(randomForest)

# Train a Random Forest model

rf_model <- randomForest(cnt ~ ., data = data %>% select(-instant, -dteday, -casual, -
                                                         -yr, -atemp), importance = TR


print(importance(rf_model))
varImpPlot(rf_model, main = "Random Forest Variable Importance")


# Load necessary libraries
library(xgboost)
library(Matrix)
library(dplyr)
library(ggplot2)

# Prepare the data
target <- data$cnt
df_matrix <- sparse.model.matrix(cnt ~ . -1, data = data %>% select(-instant, -dteday,
                                                                    -yr, -atemp))

# Convert to DMatrix format
dtrain <- xgb.DMatrix(data = df_matrix, label = target)

# Train XGBoost model with Poisson regression
xgb_model <- xgboost(data = dtrain,
                     max_depth = 6,
                     eta = 0.1,
                     nrounds = 200,
                     objective = "count:poisson",  # <-- use Poisson regression object
                     eval_metric = "poisson-nloglik",  # optional evaluation metric
                     verbose = 0)
# Get feature importance
importance_matrix <- xgb.importance(feature_names = colnames(df_matrix), model = xgb_m

# Print feature importance
print(importance_matrix)

# Plot feature importance
ggplot(importance_matrix, aes(x = reorder(Feature, Gain), y = Gain, fill = Gain)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "XGBoost Feature Importance (Poisson Gain)", x = "Variable", y = "Gain"
  theme_minimal() +
  scale_fill_gradient(low = "blue", high = "red")


library(rstanarm)

# Fit Bayesian regression with Normal prior
bayesian_normal <- stan_glm(cnt ~ ., family = poisson, prior = normal(),
                            data = data %>% select(-instant, -dteday, -casual, -regist
                                                                  -yr, -atemp ))
summary(bayesian_normal)

# Plot 95% credible intervals
plot(bayesian_normal, prob = 0.95)
# Fit Bayesian regression with Laplace prior
bayesian_laplace <- stan_glm(cnt ~ ., family = poisson, prior = laplace(),
                          data = data %>% select(-instant, -dteday, -casual, -register
summary(bayesian_laplace)

# Plot 95% credible intervals
plot(bayesian_laplace, prob = 0.95)
```

12 References

References

1. Capital Bikeshare. (2013). Capital Bikeshare system data. Retrieved from https://www.capitalbikeshare.com/syster data
2. Zhang, Y., & Mi, Z. (2016). Environmental benefits of bike sharing: A big data-based analysis. Applied Energy, 220, 296-301.
3. McCullagh, P., Nelder, J. A. (1989). Generalized Linear Models (2nd ed.). Chapman and Hall.
4. Dobson, A. J., Barnett, A. G. (2018). An Introduction to Generalized
5. Linear Models (4th ed.). CRC Press.
6. Cameron, A. C., Trivedi, P. K. (2013). Regression Analysis of Count Data (2nd ed.). Cambridge University Press.
7. Winkelmann, R. (2008). Econometric Analysis of Count Data (5th ed.). Springer.
8. Fox, J., Weisberg, S. (2018). An R Companion to Applied Regression (3rd ed.). Sage.