Comparative Study of Regularization Techniques in Logistic Regression

Efua Ainooson Noonoo

March 2, 2025

Introduction

The primary objective of this project is to analyze a leukemia gene expression dataset using various penalized regression methods. In high-dimensional settings—where the number of genes far exceeds the number of samples—traditional logistic regression often suffers from overfitting and multicollinearity. Penalized regression techniques, such as Lasso, Ridge, and Elastic Net, mitigate these challenges by incorporating regularization that shrinks coefficients, effectively selecting the most informative genes while improving model interpretability and predictive accuracy. As demonstrated by Greenwood et al. (2020) [Greenwood et al., 2020], penalized regression is a highly effective approach in such contexts. By applying these methods, this study aims to identify key genes associated with leukemia and to develop robust logistic regression models for classifying leukemia subtypes.

Data Description

The leukemia dataset, originally published by [Golub et al., 1999], was used in this study. Referred to as `Golub_Merge`, it consists of 7129 genes (genes) measured across 72 samples. The dataset is structured as an `ExpressionSet` object in R, containing a gene expression matrix (`exprs`), where rows represent genes and columns correspond to patient samples. The accompanying metadata includes leukemia subtype classifications (`ALL` vs. `AML`), determined based on clinical diagnosis and molecular profiling.

While the dataset is well-suited for demonstrating the power of penalized regression techniques, it also has inherent limitations. With only 72 samples, the dataset poses challenges for generalizability and increases the risk of overfitting, even when using regularization methods. Additionally, the distribution of leukemia subtypes is uneven, with 47 samples of ALL and 25 of AML, which could introduce bias in classification results.

Research Questions.

- How do LASSO, Ridge, and Elastic Net compare in terms of predictive performance and feature selection stability for classifying leukemia subtypes?
- How does the choice of regularization method influence the trade-off between model complexity and multicollinearity management in leukemia gene expression analysis?

Modeling and Methods

Modeling

Logistic Regression Model

The logistic regression model was fitted to assess baseline performance. This model was selected as a benchmark due to its simplicity and interpretability, making it suitable for evaluating how well the genes in the dataset predict the binary response variable—leukemia subtype (ALL vs. AML). Logistic regression assumes a linear relationship between the genes and the log-odds of the response, providing an interpretable foundation for comparison against more complex models.

Penalized Logistic Regression Models

Penalized logistic regression models, specifically LASSO, Ridge, and Elastic Net, were employed to address the challenges of high-dimensional data, where the number of genes far exceeds the number of samples. These models incorporate regularization techniques that impose penalties on the magnitude of the coefficients, thereby achieving three key objectives. LASSO, with its L_1-norm penalty, performs variable selection by shrinking some coefficients to exactly zero, thereby identifying the most predictive genes for leukemia subtypes. Ridge regression, using an L_2-norm penalty, minimizes the impact of collinearity among genes by distributing the influence across correlated variables. Elastic Net combines the strengths of LASSO and Ridge by blending L_1 and L_2 penalties. This approach is particularly useful when genes are highly correlated, as it retains feature selection capabilities while also managing multicollinearity [Friedman et al., 2010, Friedman et al., 2010].

Cross-Validation and Hyperparameter Tuning

To ensure the reliability and generalizability of the models, 10-fold cross-validation was performed to optimize the regularization parameter (λ). Two key λ values were identified: λ_{min}, which minimizes the cross-validation error and favors predictive accuracy, and λ_{1se}, which is the largest value of λ within one standard error of the minimum error and emphasizes model parsimony and interpretability. Additionally, the Elastic Net penalty was fine-tuned by optimizing the α parameter through cross-validation, balancing the sparsity of LASSO ($\alpha = 1$) with the grouping effect of Ridge ($\alpha = 0$).

Feature Selection and Stability

Feature selection was conducted independently by each penalized regression method (LASSO, Ridge, and Elastic Net), with non-zero coefficients considered as selected genes. To ensure robustness, a secondary analysis using 10-fold cross-validation was performed to evaluate the frequency of feature selection across folds. Only those genes that were consistently selected (i.e., appearing in a majority of the folds) were considered stable genes. This two-step process ensures that the final set of genes is both reliable and informative.

Prediction and Performance Evaluation

The final models were applied to classify leukemia subtypes, predicting whether samples belonged to ALL or AML. Model performance was evaluated using key metrics:

- **Sensitivity:** The proportion of actual ALL cases that were correctly identified.
- **Specificity:** The proportion of actual AML cases that were correctly identified.
- **Accuracy:** The overall correctness of the model's predictions.
- **Precision:** The proportion of predicted ALL cases that were truly ALL.

Model Checking

Model fit and complexity were assessed using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) on the test dataset. LASSO achieved the lowest AIC (54.3740) and BIC (62.0221), indicating its strong generalization performance and preference for sparse models. Elastic Net followed with AIC and BIC values of 99.5926 and 145.4812, respectively, demonstrating a balance between sparsity and feature retention. Ridge exhibited substantially higher AIC (14303.32) and BIC (27934.13) due to its lack of feature selection and high degrees of freedom, retaining all genes.

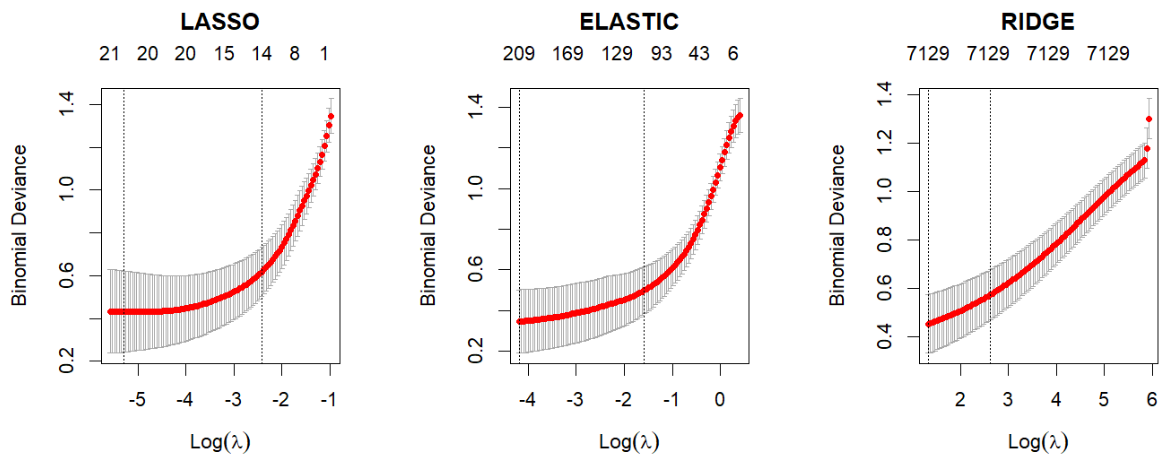
Also, prediction metrics were also analysed. Overall, these diagnostics, together with key prediction metrics, support the selection of LASSO as the most parsimonious and predictive model.

Results

Table 1 summarizes the optimal regularization parameters selected via 10-fold cross-validation for each penalized regression method. Figures 1a, 1c, and 1b display the corresponding cross-validation curves for Lasso, Ridge, and Elastic Net, respectively. Notably, Lasso regression achieved the lowest λ_{min} value of 0.004995635, reflecting its strong ability to shrink less informative coefficients to zero. In contrast, Ridge regression produced a much larger λ_{min} of 3.779015, consistent with its tendency to retain all genes while addressing multicollinearity. Additionally, the Elastic Net model, with an optimized α of 0.25, provided a balance between Lasso's sparsity and Ridge's grouping effects.

Penalized Technique	λ_{min}	λ_{1se}	α
Lasso	0.004995635	0.089354407	1
Ridge	3.779015	13.900652	0
Elastic Net	0.01511606	0.20452751	0.25

Table 1: Optimal λ values selected by Lasso, Ridge, and Elastic Net via 10-fold cross-validation.



(a) Lasso (b) Elastic Net (c) Ridge

Figure 1: Cross-validation curves for the penalized regression methods. The vertical lines indicate λ_{min} and $\lambda_{1\text{se}}$ for each method.

Feature Selection and Stability Analysis

The pure Ridge regression rarely drives any coefficient to exactly zero, it does not explicitly 'select' variables the way Lasso does. To ensure a fair comparison, we impose a small numeric threshold—here $\beta_j > 1 \times 10^{-3}$ and treat any coefficient with an absolute value below that as effectively zero. This way, all three methods (Lasso, Ridge, Elastic Net) are evaluated using the same cutoff criterion, making their selected feature sets more directly comparable.

The Lasso, Ridge, and Elastic Net methods were employed to identify genes relevant for classifying leukemia subtypes (ALL vs. AML) without incorporating cross-validation. Lasso identified 19 genes, optimizing for sparsity by shrinking coefficients of less important genes to zero. Ridge retained all 7129 genes, effectively addressing multicollinearity using its L_2-norm penalty. Elastic Net, with an optimized alpha value of 0.25, achieved a balance between Lasso's sparsity and Ridge's inclusiveness by selecting 210 genes. Table 2 summarizes the number of genes selected and the deviance values for each method.

Method	**genes Selected**	**Deviance**
Lasso	19	98.69
Ridge	1179	92.80
Elastic Net	193	98.61

Table 2: genes selected and deviance values for Lasso, Ridge, and Elastic Net.

Number of genes selected are displayed in Figures 2.

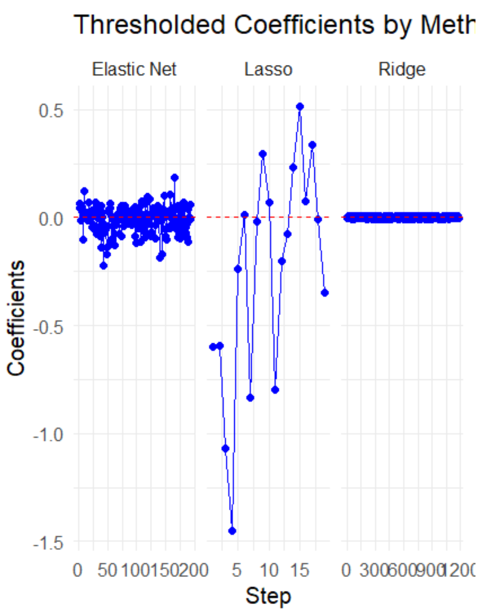


Figure 2: Plot showing genes selected by the penalised regression methods

Feature overlaps among the three methods were examined using a threshold of 1×10^{-3} for Ridge. Lasso and Ridge shared 19 common genes.
Lasso and Elastic Net shared 19 genes, while Elastic Net and Ridge shared 193 genes. All three methods identified the same 19 core genes; L20861_at, M15990_at, M19483_at, M23161_at, M27878_at, M31606_at, M31951_at, M84349_at, X51466_at, X59405_at, X95715_at, Y07596_at, Y12556_at, M60828_at, X06182_s_at, U40002_s_at, M92843_s_at, Z31690_s_at, and X83490_s_at underscoring their importance in classifying ALL and AML. These overlaps are visually represented in Figure 3

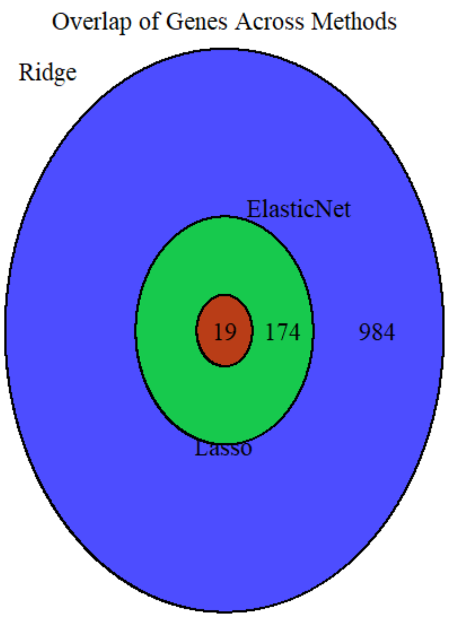


Figure 3: Venn diagram showing overlap and unique genes selected by Lasso, Ridge, and Elastic Net across folds.

To evaluate the stability and robustness of feature selection, 10-fold cross-validation was performed. The Lasso, Ridge, and Elastic Net methods were employed to identify stable and robust genes relevant for classifying leukemia subtypes (ALL vs. AML). genes that appeared in at least 80% of the cross-validation folds (8 out of 10 folds) were considered stable and selected for the final models. Lasso identified 10 genes. Ridge identified all 806 genes. Elastic Net identified 133 genes. Table 3 summarizes the number of genes selected and the deviance values for each method. Feature overlaps among the methods were examined using Venn diagrams. All three methods consistently identified a core set of 10 genes; `HG2562-HT2658_s_at M23197_at X95735_at M19507_at Y07604_at M16038_at M31994_at M63838_s_at X51521_at`. Figure 4 shows the overlap among the stable genes (selected in ≥8 folds).

Method	**Stable genes Selected**
Lasso	10
Ridge	806
Elastic Net	133

Table 3: Stable genes selected and deviance values for Lasso, Ridge, and Elastic Net.

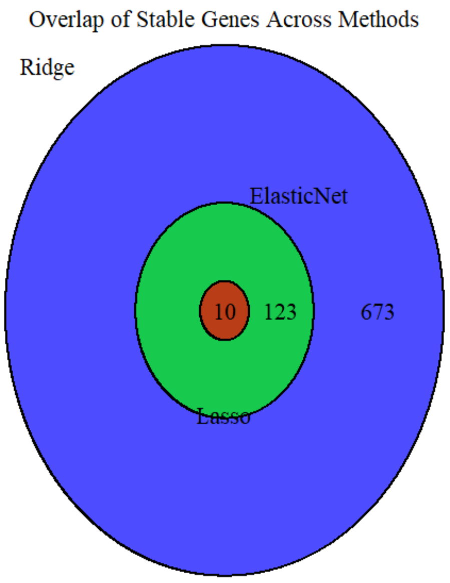


Figure 4: Venn diagram showing stable genes selected by Lasso, Ridge, and Elastic Net across folds.

Figure 5 shows the stable genes selected by Lasso. The stable Elastic Net genes are displayed in a grid format in Figure **??** shows the first 80 stable genes selected by Elastic. For clarity, the genes have been grouped into subsets. Figure 11b presents the first 80 stable Ridge genes in a grid, grouped for clarity.

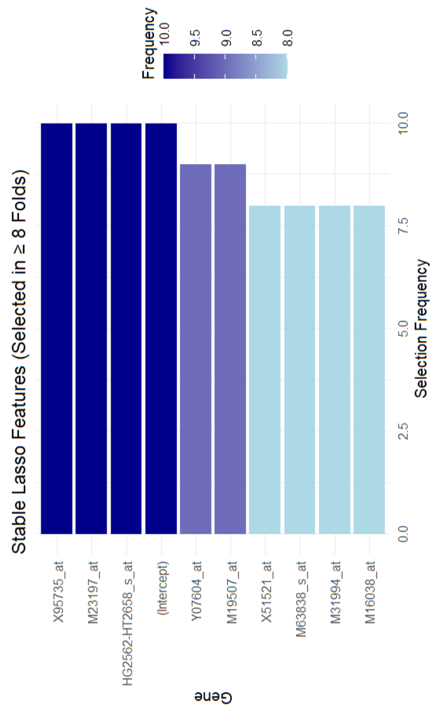
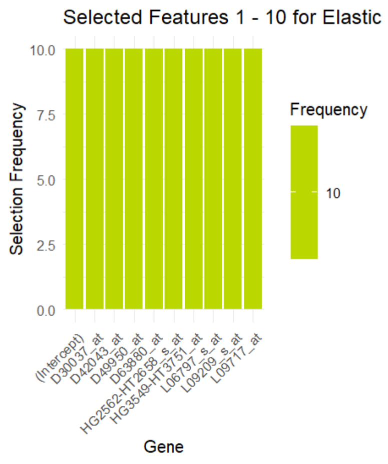
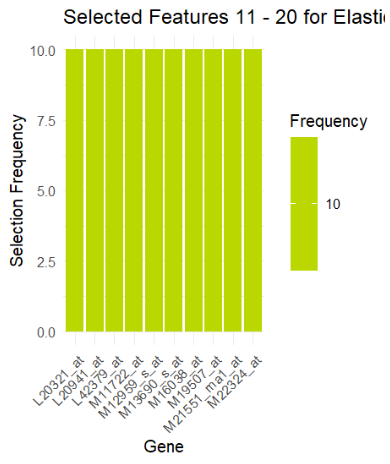


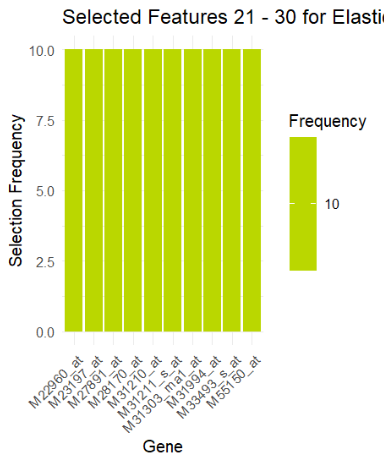
Figure 5: Stable Lasso genes (selected in ≥8 folds).



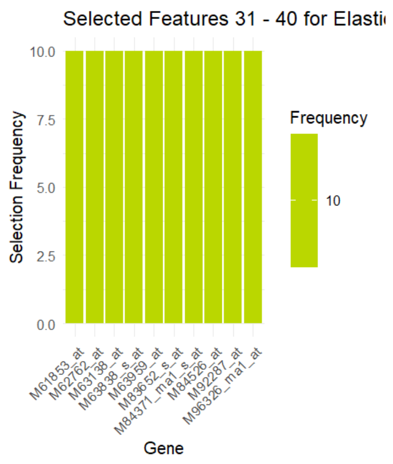
(a) Group 1



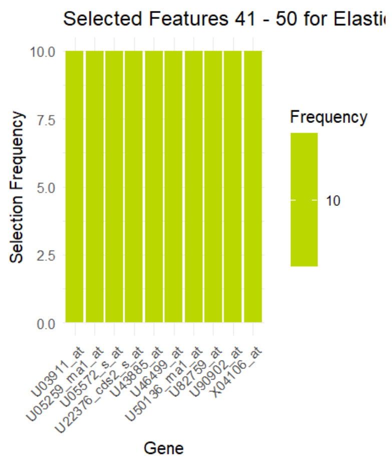
(b) Group 2



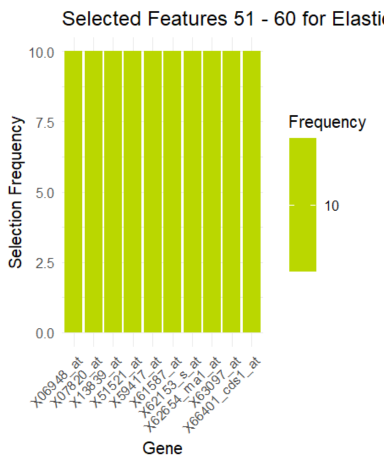
(c) Group 3



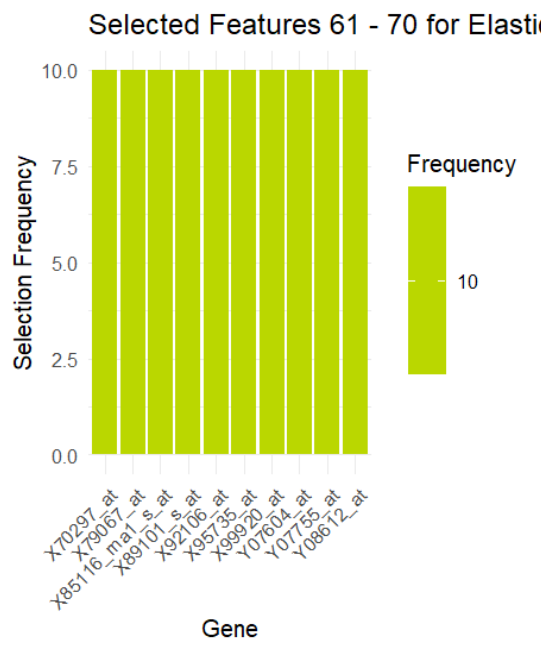
(a) Group 4



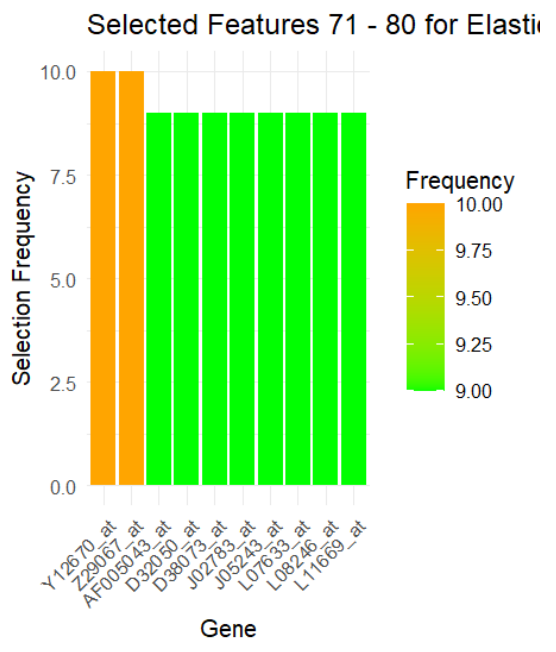
(b) Group 5



(c) Group 6

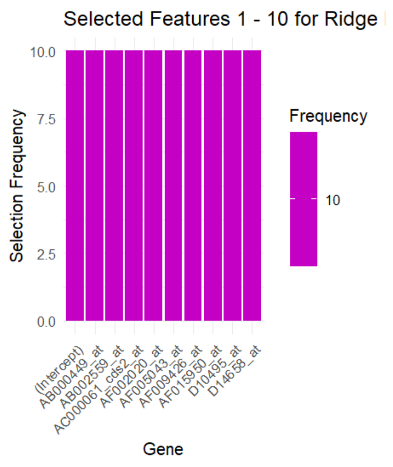


(a) Group 7

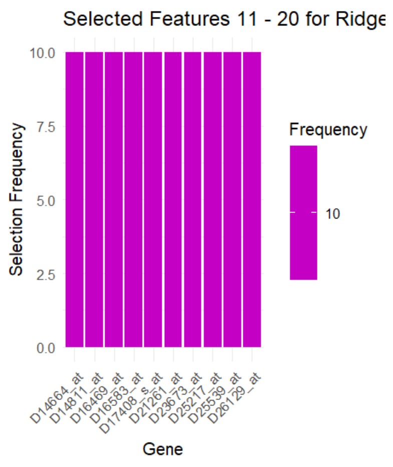


(b) Group 8

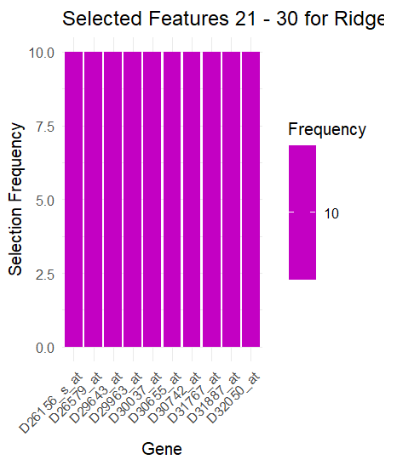
Figure 8: Stable Elastic Net genes (groups of 10).



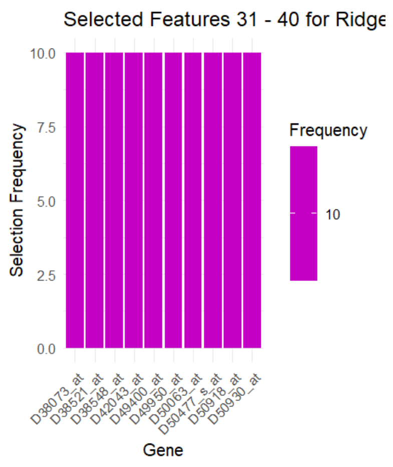
(a) Group 1



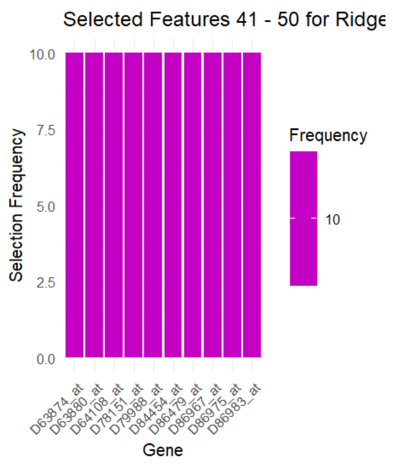
(b) Group 2



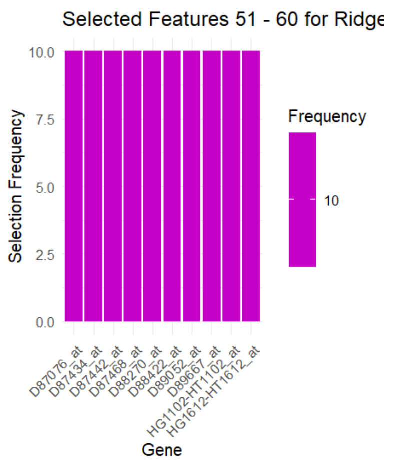
(c) Group 3



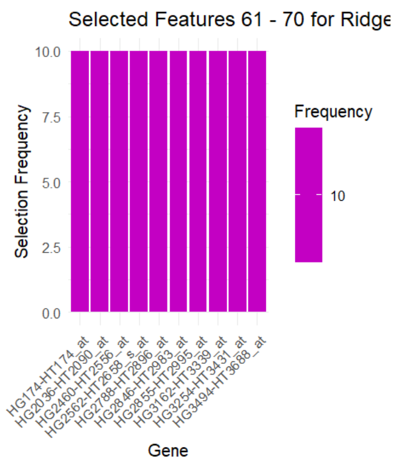
(a) Group 4



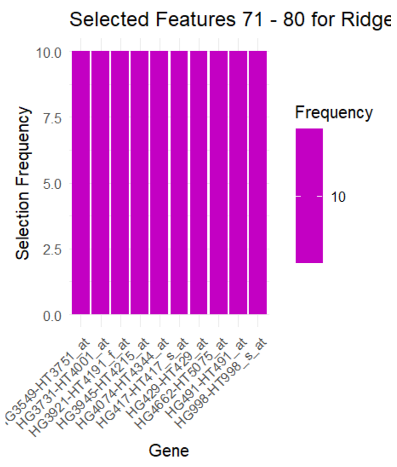
(b) Group 5



(c) Group 6



(a) Group 7



(b) Group 8

Figure 11: Stable Ridge genes (groups of 10).

Prediction and Performance Metrics

The models were first evaluated without cross-validation to assess their ability to classify leukemia subtypes (ALL vs. AML) using key performance metrics, including accuracy, precision, sensitivity, and specificity. Table 4 summarizes the results. The Lasso and Elastic Net models achieved a perfect classification performance on the test data, with an accuracy, precision, sensitivity, and specificity of 100%. These models correctly classified all samples with no false positives (AML cases incorrectly predicted as ALL) or false negatives (ALL cases incorrectly predicted as AML). This performance highlights their capability to balance precision and sensitivity effectively, ensuring robust predictions for both ALL and AML subtypes.

The Ridge model, while highly effective in identifying ALL cases with a sensitivity of 100%, showed some limitations in specificity. It achieved an overall accuracy of 91%, with a precision of 88% and specificity of 71%. This reduced specificity is attributed to two false positives, where AML samples were incorrectly classified as ALL. Despite these false positives, Ridge maintained a strong ability to correctly classify ALL cases, achieving perfect sensitivity. Figure 13 show the methods and their performance.

Method	**Counts**				**Metrics**			
	TN	**FP**	**FN**	**TP**	**Accuracy (%)**	**Precision (%)**	**Sensitivity (%)**	**Specificity (%)**
Elastic Net	7	0	0	15	100	100	100	100
Ridge	5	2	0	15	90.9	88.2	100	71.4
Lasso	7	0	0	15	100	100	100	100

Table 4: Comparison of Elastic Net, Ridge, and Lasso Prediction Results.

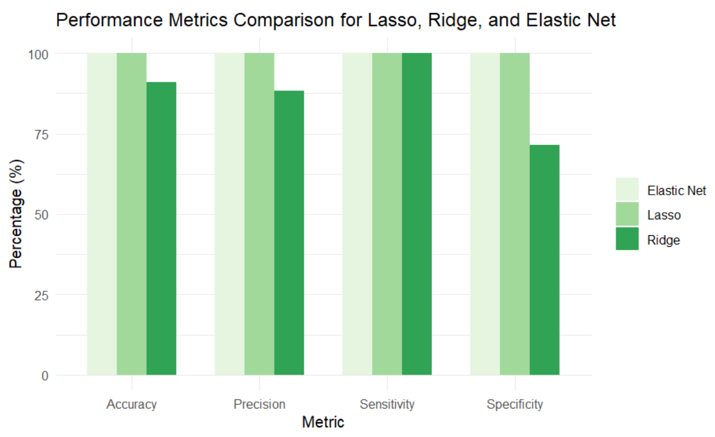


Figure 12: Performance metrics for Lasso, Ridge, and Elastic Net models for stable genes.

To ensure robustness, the final models were evaluated using the stable genes obtained through cross-validation. The results confirmed the initial findings, both Lasso and Elastic Net maintaining perfect performance (100% accuracy, precision, sensitivity, and specificity). Ridge regression, also maintained 100% performance. These findings underscore the ability of Lasso and Elastic Net to provide robust and reliable predictions using the most informative and stable genes, as shown in Table 5 and Figure 12.

Method	Counts				Metrics			
	TN	FP	FN	TP	Accuracy %	Precision %	Sensitivity %	Specificity %
LASSO & Elastic & Ridge	7	0	0	15	100	100	100	100

Table 5: Comparison of LASSO & Elastic Net and Ridge Prediction Results.

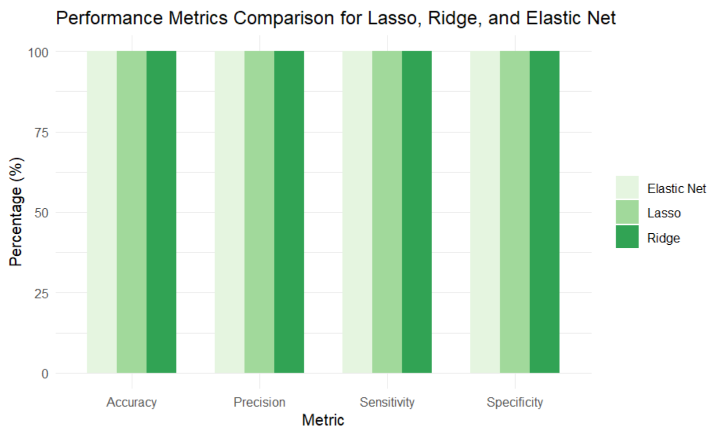


Figure 13: Performance metrics for Lasso, Ridge, and Elastic Net models.

Lasso demonstrated exceptional performance in classifying leukemia subtypes, achieving perfect accuracy while selecting the sparsest set of genes, making it highly interpretable. Elastic Net also achieved perfect classification but retained more genes, balancing sparsity and handling correlated genes effectively. Ridge, while managing multicollinearity, struggled with specificity and retained all genes, limiting its interpretability and practical applicability. The stability analysis confirmed Lasso as the most robust model, consistently selecting a smaller, stable feature set. Lasso is the recommended approach, with Elastic Net as a viable alternative when additional feature retention is desired.

Discussion of Results

Lasso and Elastic Net models achieved perfect classification performance on the test data, demonstrating both high predictive accuracy and effective feature selection. Ridge, while achieving 91% accuracy, struggled with specificity due to a few false positives. The stability analysis confirmed that Lasso consistently selected a small, robust set of genes, whereas Ridge retained a larger number of genes. The overlaps in selected genes (illustrated by the Venn diagrams) reinforce the importance of a core set of genes in distinguishing between ALL and AML.

Overall, the results provide a comprehensive comparative analysis of penalized regression methods, with robust feature selection and performance evaluation. Further enhancements in model diagnostics and biological interpretation could elevate the project even more.

Potential Pitfalls

The perfect classification performance reported for Lasso and Elastic Net (100% accuracy, sensitivity, and specificity) is a critical concern. Achieving such flawless results in biomedical data is highly unusual and often indicative of potential issues such as data leakage, overfitting, or improper train-test split methodology. Further scrutiny is necessary to ensure the validity of these findings.

Additionally, the test set comprises only 22 samples (15 ALL and 7 AML), which is both small and imbalanced. This combination raises concerns about the robustness and representativeness of the reported performance metrics. A small test set limits the ability to generalize these results to larger populations reliably. Leukemia datasets are typically small, exacerbating the risk of overfitting and undermining the stability of feature selection. Additional evaluation with larger or independent datasets is recommended to confirm the robustness of these results.

Another challenge lies in the disparity in the number of stable genes selected by the models (Lasso: 10, Elastic Net: 133, Ridge: 806). While Lasso's aggressive sparsity enhances interpretability, it risks excluding potentially important genes. Conversely, Ridge retains an excessively large number of genes, reducing its practicality and interpretability. The threshold for defining stable genes (genes appearing in at least 80% of cross-validation folds) and 1×10^{-3} may also require further validation to confirm its suitability and relevance for this analysis.

Appendix

```r
library(glmnet)
library(tidyr)
library(dplyr)
library(ggplot2)
library(caret)
library(kableExtra)
library(RColorBrewer)
```

Listing 1: R libraries

```r
data(Golub_Merge)

expression_data = exprs(Golub_Merge)

sample_labels = pData(Golub_Merge)$ALL.AML

summary(sample_labels)

dataset_summary = as.data.frame(table(sample_labels))
colnames(dataset_summary) = c("Leukemia_Type", "Count")


table_visual <- dataset_summary %>%
  kable("html", col.names = c("Leukemia Type", "Sample Count"),
     align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "
     condensed", "responsive")) %>%
  row_spec(0, bold = TRUE, background = "#f2f2f2") %>%
  add_header_above(c("Dataset Breakdown" = 2))
```

```r

table_visual


# A bar chart for dataset breakdown

ggplot(dataset_summary, aes(x = Leukemia_Type, y = Count, fill =
   Leukemia_Type)) +
  geom_bar(stat = "identity", width = 0.5) +
  theme_minimal() +
  scale_fill_manual(values = c("ALL" = "purple", "AML" = "
     turquoise")) +
  labs(
    title = "Dataset Breakdown by Leukemia Type",
    x = "Leukemia Type",
    y = "Sample Count"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.text.x = element_text(face = "bold"),
    axis.text.y = element_text(face = "bold")
  )


  expression_df= as.data.frame(t(expression_data[1:5, ]))
colnames(expression_df) = paste0("Gene", 1:5)
expression_df$LeukemiaType = sample_labels


long_data = pivot_longer(expression_df, cols = starts_with("Gene"
   ),
                          names_to = "Gene", values_to = "
                             Expression")

# Boxplots with colors for leukemia types
ggplot(long_data, aes(x = Gene, y = Expression, fill =
   LeukemiaType)) +
  geom_boxplot() +
  scale_fill_manual(values = c("ALL" = "purple", "AML" = "
     turquoise")) +
  theme_minimal() +
  labs(title = "Gene Expression by Leukemia Type",
       x = "Gene",
       y = "Expression Level",
       fill = "Leukemia Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


scaled_expression_data = scale(expression_data)


```

```r
## Encoding the response variable as 0 for AML and 1 for ALL

response = numeric(length(sample_labels))

for (i in seq_along(sample_labels)){
  if (sample_labels[i] == "ALL"){

  response[i] = 1
  }
  else {
  response[i] = 0
  }
}

## : Split the dataset(sample) into training (70%) and testing
    (30%) sets for model.

n_samples = ncol(Golub_Merge)

set.seed(4321)

train_indices = sample(1:n_samples, size = 0.7*n_samples)


training_data = Golub_Merge[,train_indices]

testing_data = Golub_Merge[,-train_indices]

response_train = response[train_indices]

response_test = response[-train_indices]

expression_train = as.data.frame(t(exprs(training_data)))

expression_train$response = response_train

X = subset(expression_train, select = -response)

X_mat = as.matrix(X)

X_mat = scale(X_mat)

expression_test = as.data.frame(t(exprs(testing_data)))
```

Listing 2: Data Processing and Exploration

```r

## Linear Regression

model_lm =
lm(response~., data = expression_train)

```

```r
summary(model_lm)

## Generalised Linear Regression

model_glm =
glm(response˜., data = expression_train, family=binomial(link=
   logit))

summary(model_glm)


## Deviance goodness-of-fit test

residual_deviance = model_glm$deviance

df = model_glm$df.residual

p_value = 1 - pchisq(residual_deviance, df)

residual_deviance

p_value


deviance_residuals = residuals(model_glm, type = "deviance")
fitted_values = fitted(model_glm)

plot(fitted_values,deviance_residuals,
     xlab = "Fitted Values",
     ylab = "Deviance Residuals",
     main = "Deviance Residuals vs Fitted Values",
     pch = 19,
     col = "turquoise")

# Test on a simulated binary classification dataset
set.seed(42)
X_test = X_mat
y_test = expression_train$response

# Fit Lasso, Ridge, and Elastic Net
fit_lasso <- glmnet(X_test, y_test, family = "binomial", alpha =
   1)
fit_ridge <- glmnet(X_test, y_test, family = "binomial", alpha =
   0)
fit_elastic <- glmnet(X_test, y_test, family = "binomial", alpha
   = 0.25)

# Calculate AIC/BIC
aic_bic_lasso = calculate_aic_bic(fit_lasso, fit_lasso$lambda
   [10], X_test, y_test)
aic_bic_ridge = calculate_aic_bic(fit_ridge, fit_ridge$lambda
   [10], X_test, y_test)
aic_bic_elastic = calculate_aic_bic(fit_elastic, fit_elastic$
   lambda[10], X_test, y_test)

cat("Test Lasso AIC:", aic_bic_lasso$AIC, "BIC:", aic_bic_lasso$
   BIC, "\n")
cat("Test Ridge AIC:", aic_bic_ridge$AIC, "BIC:", aic_bic_ridge$
   BIC, "\n")
cat("Test Elastic Net AIC:", aic_bic_elastic$AIC, "BIC:", aic_bic
   _elastic$BIC, "\n")
```

Listing 3: Model Checking

```

## mean deviance for lasso regression

set.seed(4321)

fit_cv_lasso = cv.glmnet(
  x = X_mat,
  y = expression_train$response,
 family = "binomial",
  alpha = 1,
  nfolds = 10,
  #type.measure = "class"
  )

plot(fit_cv_lasso)
title("LASSO", line = 2.5)

list(c( "lambda.min.lasso" = fit_cv_lasso$lambda.min,

"lambda.1se.lasso" = fit_cv_lasso$lambda.1se))

## mean deviance for ridge regression

set.seed(4321)

fit_cv_ridge = cv.glmnet(
  x = X_mat,
  y = expression_train$response,
 family = "binomial",   # logistic regression
  alpha = 0,             # lasso penalty
  nfolds = 10,           # 10-fold CV
  #type.measure = "class" # classification error for binomial
  )

plot(fit_cv_ridge)
title("RIDGE", line = 2.5)

list(c( "lambda.min.ridge" = fit_cv_ridge$lambda.min,

```

```r
"lambda.1se.ridge" = fit_cv_ridge$lambda.1se))

set.seed(4321)

alpha_values = c(0, 0.25, 0.5, 0.75, 1)

results = data.frame(alpha = numeric(),
                      lambda.min = numeric(),
                      lambda.1se = numeric(),
                      cv.error.min = numeric(),
                      cv.error.1se = numeric(),
                      stringsAsFactors = FALSE)

for (a in alpha_values) {

  # Perform cross-validation for given alpha

  cv_fit = cv.glmnet(X_mat, expression_train$response, family = "
     binomial", alpha = a, type.measure = "class")

 results = rbind(results,
                   data.frame(
                     alpha = a,
                     lambda.min = cv_fit$lambda.min,
                     lambda.1se = cv_fit$lambda.1se,
                     cv.error.min = min(cv_fit$cvm),
                     cv.error.1se = cv_fit$cvm[which(cv_fit$
                        lambda == cv_fit$lambda.1se)]
                   ))
}

print(results)

best <- results[which.min(results$cv.error.min), ]
cat("Best alpha:", best$alpha,
    "with lambda.min:", best$lambda.min,
    "achieved a cross-validation error of:", best$cv.error.min, "
       \n")

## mean deviance for elastic regression

set.seed(4321)

fit_cv_elastic = cv.glmnet(
  x = X_mat,
  y = expression_train$response,
 family = "binomial",
  alpha = 0.25,
  nfolds = 10,
  #type.measure = "class" # classification error for binomial
  )
```

```

plot(fit_cv_elastic)
title("ELASTIC", line = 2.5)

list(c( "lambda.min.elastic" = fit_cv_elastic$lambda.min,

"lambda.1se.elastic" = fit_cv_elastic$lambda.1se))
```

Listing 4: Mean Deviance

```


rlas = glmnet(X_mat, expression_train$response,  family = "
   binomial", alpha = 1,  lambda = fit_cv_lasso$lambda.min)

rrid = glmnet(X_mat, expression_train$response,  family = "
   binomial", alpha = 0,  lambda = fit_cv_ridge$lambda.min)

renet = glmnet(X_mat, expression_train$response, family = "
   binomial", alpha = .25, lambda = fit_cv_elastic$lambda.min)
```

Listing 5: Fitting the methods

```


## non-zero coefficients for lasso

coef_matrix_lasso = coef(rlas)

coef_values_lasso = coef_matrix_lasso[-1,,drop = FALSE]

nonzero_idx_lasso = which(coef_values_lasso != 0)

nonzero_coefs_lasso = coef_values_lasso[nonzero_idx_lasso, , drop
    = FALSE]



## non-zero coefficients for ridge


coef_matrix_ridge = coef(rrid)

coef_values_ridge = coef_matrix_ridge[-1,,drop = FALSE]

nonzero_idx_ridge = which(coef_values_ridge != 0)

nonzero_coefs_ridge = coef_values_ridge[nonzero_idx_ridge, , drop
    = FALSE]


## non-zero coefficients for elastic
```

```r

coef_matrix_elastic = coef(renet)

coef_values_elastic = coef_matrix_elastic[-1,,drop = FALSE]

nonzero_idx_elastic = which(coef_values_elastic != 0)

nonzero_coefs_elastic = coef_values_elastic[nonzero_idx_elastic, , drop = FALSE]

# Remove intercept row for each method
coef_values_lasso_noIntercept = coef_values_lasso[-1,, drop=FALSE]
coef_values_ridge_noIntercept = coef_values_ridge[-1,, drop=FALSE]
coef_values_elastic_noIntercept = coef_values_elastic[-1,, drop=FALSE]

# Threshold


# Lasso
threshold_idx_lasso = which(abs(coef_values_lasso_noIntercept) > threshold)
threshold_coefs_lasso = coef_values_lasso_noIntercept[threshold_idx_lasso, , drop=FALSE]
selected_genes_lasso = rownames(coef_values_lasso_noIntercept)[threshold_idx_lasso]

# Ridge
threshold_idx_ridge = which(abs(coef_values_ridge_noIntercept) > threshold)
threshold_coefs_ridge = coef_values_ridge_noIntercept[threshold_idx_ridge, , drop=FALSE]
selected_genes_ridge = rownames(coef_values_ridge_noIntercept)[threshold_idx_ridge]

# Elastic Net
threshold_idx_elastic = which(abs(coef_values_elastic_noIntercept) > threshold)
threshold_coefs_elastic = coef_values_elastic_noIntercept[threshold_idx_elastic, , drop=FALSE]
selected_genes_elastic = rownames(coef_values_elastic_noIntercept)[threshold_idx_elastic]

# Compare coeff selected
list(
  lasso   = length(threshold_coefs_lasso),
  ridge   = length(threshold_coefs_ridge),
  elastic = length(threshold_coefs_elastic)
)
```

```


df_lasso <- data.frame(
  Step        = seq_len(nrow(threshold_coefs_lasso)),
  Coefficient = threshold_coefs_lasso[, 1],
  Method      = "Lasso"
)

df_elastic <- data.frame(
  Step        = seq_len(nrow(threshold_coefs_elastic)),
  Coefficient = threshold_coefs_elastic[, 1],
  Method      = "Elastic Net"
)

df_ridge <- data.frame(
  Step        = seq_len(nrow(threshold_coefs_ridge)),
  Coefficient = threshold_coefs_ridge[, 1],
  Method      = "Ridge"
)


df_all <- rbind(df_lasso, df_elastic, df_ridge)
```

Listing 6: Selection of non-zero coefficients

```


y_min = min(df_all$Coefficient, na.rm = TRUE)
y_max = max(df_all$Coefficient, na.rm = TRUE)

ggplot(df_all, aes(x = Step, y = Coefficient)) +

  geom_line(color = "blue") +
  geom_point(color = "blue") +

  geom_hline(yintercept = 0, linetype = "dashed", color = "red")
    +

  facet_wrap(~ Method, scales = "free_x") +

  coord_cartesian(ylim = c(y_min, y_max)) +
  labs(
    title = "Thresholded Coefficients by Method",
    x = "Step",
    y = "Coefficients"
  ) +
  theme_minimal()
```

Listing 7: Plot Threshold Coefficients by Method

```


```

```r
variable_names = colnames(X_mat)

# Common coefficients between LASSO and Ridge
common_coefficients1 = Reduce(intersect, list(threshold_idx_lasso
    , threshold_idx_ridge))
common_vars_lasso_ridge = variable_names[common_coefficients1]
cat("Common variables between LASSO and Ridge:\n", common_vars_
    lasso_ridge, "\n")

# Common coefficients between LASSO and Elastic Net
common_coefficients2 = Reduce(intersect, list(threshold_idx_lasso
    , threshold_idx_elastic))
common_vars_lasso_elastic = variable_names[common_coefficients2]
cat("Common variables between LASSO and Elastic Net:\n", common_
    vars_lasso_elastic, "\n")

# Common coefficients between Elastic Net and Ridge
common_coefficients3 = Reduce(intersect, list(threshold_idx_
    elastic, threshold_idx_ridge))
common_vars_elastic_ridge = variable_names[common_coefficients3]
cat("Common variables between Elastic Net and Ridge:\n", common_
    vars_elastic_ridge, "\n")

# Common coefficients across all three methods
common_coefficients4 = Reduce(intersect, list(threshold_idx_lasso
    , threshold_idx_ridge, threshold_idx_elastic))
common_vars_all = variable_names[common_coefficients4]
cat("Common variables across all three methods:\n", common_vars_
    all, "\n")

### venn diagram of associated genes

venn.plot =  venn.diagram(
  x = list(
    Lasso = threshold_idx_lasso,
    Ridge = threshold_idx_ridge,
    ElasticNet = threshold_idx_elastic
  ),
  filename = NULL,
  fill = c("red", "blue", "green"),
  alpha = 0.7,
  main = "Overlap of Genes Across Methods"
)
grid.newpage()
grid.draw(venn.plot)
```

Listing 8: Associated Genes

```r


# Extract genes from test data and convert to matrix

```

```r
X_test = as.matrix(expression_test)

## Predict probabilities on test data; (threshold = 0.5)

pred_probs_lasso = predict(lasso_model, newx = X_test, type = "
   response")

pred_class_lasso = ifelse(pred_probs_lasso > 0.5, 1, 0)

## Model performance on test data

confusion_matrix_lasso = table(Predicted = pred_class_lasso,
   Actual = response_test)
print(confusion_matrix_lasso)


X_test = as.matrix(expression_test)

## Predict probabilities on test data; (threshold = 0.5)
pred_probs_ridge = predict(ridge_model, newx = X_test, type = "
   response")


pred_class_ridge = ifelse(pred_probs_ridge > 0.5, 1, 0)

## Model performance on test data

confusion_matrix_ridge = table(Predicted = pred_class_ridge,
   Actual = response_test)
print(confusion_matrix_ridge)

X_test = as.matrix(expression_test)

## Predict probabilities on test data; (threshold = 0.5)
pred_probs_elastic = predict(elastic_model, newx = X_test, type =
    "response")

pred_class_elastic = ifelse(pred_probs_elastic > 0.5, 1, 0)

# Model performance on test data

confusion_matrix_elastic = table(Predicted = pred_class_elastic,
   Actual = response_test)
print(confusion_matrix_elastic)
```

Listing 9: Prediction and Classification

```r

## Model Performance for Lasso

TN_lasso = confusion_matrix_lasso[1,1]

```

```
FN_lasso = confusion_matrix_lasso[1,2]

FP_lasso = confusion_matrix_lasso[2,1]

TP_lasso = confusion_matrix_lasso[2,2]

all_samples_lasso = TN_lasso + FN_lasso + FP_lasso + TP_lasso


### Accuracy

Accuracy_lasso = ((TN_lasso + TP_lasso)/ all_samples_lasso)*100




### Sensitivity

Sensitivity_lasso = ((TP_lasso)/ (TP_lasso + FN_lasso))*100




### Specificity

Specificity_lasso = ((TN_lasso)/ (TN_lasso + FP_lasso))*100




### Precision

Precision_lasso = ((TP_lasso)/ (TP_lasso + FP_lasso))*100




list(c(Accuracy_lasso = Accuracy_lasso, Sensitivity_lasso =
   Sensitivity_lasso, Specificity_lasso = Specificity_lasso,
   Precision_lasso = Precision_lasso ))


## Model performance for Ridge

TN_ridge = confusion_matrix_ridge[1,1]

FN_ridge = confusion_matrix_ridge[1,2]

FP_ridge = confusion_matrix_ridge[2,1]

TP_ridge = confusion_matrix_ridge[2,2]
```

```




### Accuracy

Accuracy_ridge = ((TN_ridge + TP_ridge)/ (TN_ridge + TP_ridge +
   FN_ridge + FP_ridge) )*100




### Sensitivity

Sensitivity_ridge = ((TP_ridge)/ (TP_ridge + FN_ridge))*100




### Specificity

Specificity_ridge = ((TN_ridge)/ (TN_ridge + FP_ridge))*100




### Precision

Precision_ridge = ((TP_ridge)/ (TP_ridge + FP_ridge))*100




list(c(Accuracy_ridge = Accuracy_ridge, Sensitivity_ridge =
   Sensitivity_ridge, Specificity_ridge = Specificity_ridge,
   Precision_ridge = Precision_ridge ))


## Model Performance for Elastic Methods

TN_elastic = confusion_matrix_elastic[1,1]

FN_elastic = confusion_matrix_elastic[1,2]

FP_elastic = confusion_matrix_elastic[2,1]

TP_elastic = confusion_matrix_elastic[2,2]




```

```r
### Accuracy

Accuracy_elastic = ((TN_elastic + TP_elastic)/ (TN_elastic + TP_elastic + FN_elastic + FP_elastic) )*100




### Sensitivity

Sensitivity_elastic = ((TP_elastic)/ (TP_elastic + FN_elastic))*100




### Specificity

Specificity_elastic = ((TN_elastic)/ (TN_elastic + FP_elastic))*100




### Precision

Precision_elastic = ((TP_elastic)/ (TP_elastic + FP_elastic))*100


list(c(Accuracy_elastic = Accuracy_elastic, Sensitivity_elastic = Sensitivity_elastic, Specificity_elastic = Specificity_elastic, Precision_elastic = Precision_elastic ))

metrics_df = data.frame(
  Method = rep(c("Lasso", "Ridge", "Elastic Net"), each = 4),
  Metric = rep(c("Accuracy", "Sensitivity", "Specificity", "Precision"), times = 3),
  Value  = c(Accuracy_lasso, Sensitivity_lasso, Specificity_lasso, Precision_lasso,
             Accuracy_ridge, Sensitivity_ridge, Specificity_ridge, Precision_ridge,
             Accuracy_elastic, Sensitivity_elastic, Specificity_elastic, Precision_elastic)
)

ggplot(metrics_df, aes(x = Metric, y = Value, fill = Method)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  ylim(0, 100) +
  labs(title = "Performance Metrics Comparison for Lasso, Ridge, and Elastic Net",
       x = "Metric",
      y = "Percentage (%)") +
  scale_fill_brewer(palette = "Greens") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

Listing 10: Model Performance

```


y_min = min(df_all$Coefficient, na.rm = TRUE)
y_max = max(df_all$Coefficient, na.rm = TRUE)

ggplot(df_all, aes(x = Step, y = Coefficient)) +

  geom_line(color = "blue") +
  geom_point(color = "blue") +

  geom_hline(yintercept = 0, linetype = "dashed", color = "red")
     +

  facet_wrap(~ Method, scales = "free_x") +

  coord_cartesian(ylim = c(y_min, y_max)) +
  labs(
    title = "Thresholded Coefficients by Method",
    x = "Step",
    y = "Coefficients"
  ) +
  theme_minimal()
```

Listing 11: Plot Threshold Coefficients by Method

```


selected_genes_lasso = list()
selected_genes_ridge = list()
selected_genes_elastic = list()

# Cross-validation
set.seed(4321)
k = 10
folds = createFolds(expression_train$response, k = k, list = TRUE
    )


fit_cv_lasso = cv.glmnet(X_mat, expression_train$response,
                         family = "binomial",
                         alpha = 1)

fit_cv_ridge = cv.glmnet(X_mat, expression_train$response,
                         family = "binomial",
                         alpha = 0)
```

```r

fit_cv_elastic = cv.glmnet(X_mat, expression_train$response,
                           family = "binomial",
                           alpha = 0.5)


for (i in seq_along(folds)) {
    train_indices = unlist(folds[-i])
    valid_indices = folds[[i]]

    X_train = X_mat[train_indices, ]
    y_train = expression_train$response[train_indices]

    y_train = as.vector(y_train)

    # Train Lasso model

    fit_lasso = glmnet(X_train, y_train,
                       family = "binomial",
                       alpha = 1,
                       lambda = fit_cv_lasso$lambda.min)

    coef_lasso = coef(fit_lasso, s = fit_cv_lasso$lambda.min)

    coef_lasso_matrix = as.matrix(coef_lasso)

    selected_genes_lasso = rownames(coef_lasso_matrix)[abs(coef_
       lasso_matrix[, 1]) > threshold]

    selected_genes_lasso[[i]] = selected_genes_lasso

    # Train Ridge model

    fit_ridge = glmnet(X_train, y_train,
                       family = "binomial",
                       alpha = 0,
                       lambda = fit_cv_ridge$lambda.min)

    coef_ridge = coef(fit_ridge, s = fit_cv_ridge$lambda.min)

    coef_ridge_matrix = as.matrix(coef_ridge)

    selected_genes_ridge = rownames(coef_ridge_matrix)[abs(coef_
       ridge_matrix[, 1]) > threshold]

    selected_genes_ridge[[i]] = selected_genes_ridge

    # Train Elastic Net model

    fit_elastic = glmnet(X_train, y_train,
                         family = "binomial",
                         alpha = 0.25,
                         lambda = fit_cv_elastic$lambda.min)

    coef_elastic = coef(fit_elastic, s = fit_cv_elastic$lambda.
       min)

    coef_elastic_matrix = as.matrix(coef_elastic)

    selected_genes_elastic = rownames(coef_elastic_matrix)[abs(
       coef_elastic_matrix[, 1]) > threshold]

    selected_genes_elastic[[i]] = selected_genes_elastic
}

list(c("lasso genes" = fit_lasso$df, "lasso Deviance" = fit_
    elastic$dev.ratio,
       "ridge genes" = fit_ridge$df,
       "ridge Deviance" = fit_ridge$dev.ratio,
       "elastic genes" = fit_elastic$df, "elastic Deviance" = fit
          _elastic$dev.ratio))


#  All selected genes from all folds

selected_genes_all_lasso = unlist(selected_genes_lasso)
selected_genes_all_elastic = unlist(selected_genes_elastic)
selected_genes_all_ridge = unlist(selected_genes_ridge)


selected_genes_summary_lasso = sort(table(selected_genes_all_
    lasso), decreasing = TRUE)
selected_genes_summary_elastic = sort(table(selected_genes_all_
    elastic), decreasing = TRUE)
selected_genes_summary_ridge = sort(table(selected_genes_all_
    ridge), decreasing = TRUE)


lasso_stability = as.data.frame(selected_genes_summary_lasso)
colnames(lasso_stability) = c("Gene", "Frequency")
lasso_stability$Method = "Lasso"

elastic_stability = as.data.frame(selected_genes_summary_elastic)
colnames(elastic_stability) = c("Gene", "Frequency")
elastic_stability$Method = "Elastic Net"

ridge_stability = as.data.frame(selected_genes_summary_ridge)
colnames(ridge_stability) = c("Gene", "Frequency")
ridge_stability$Method = "Ridge"


combined_stability = rbind(lasso_stability, elastic_stability,
```

```r
    ridge_stability)


head(combined_stability)

## genes that appear in >= 8 out of 10 folds.

#  Lasso:
stable_lasso = names(selected_genes_summary_lasso[
                         selected_genes_summary_lasso >= 8
                      ])

#  Ridge:
stable_ridge = names(selected_genes_summary_ridge[
                         selected_genes_summary_ridge >= 8
                      ])

#  Elastic Net:
stable_elastic = names(selected_genes_summary_elastic[
                         selected_genes_summary_elastic >= 8
                       ])


list(c( stable_lasso = length(stable_lasso), stable_ridge =
   length(stable_ridge), stable_elastic = length(stable_elastic))
   )

# Venn diagram with stable genes

venn.plot = venn.diagram(
    x = list(
        Lasso = stable_lasso,
        Ridge = stable_ridge,
        ElasticNet = stable_elastic
    ),
    filename = NULL,
    fill = c("red", "blue", "green"),
    alpha = 0.7,
    main = "Overlap of Stable Genes Across Methods"
)
grid.newpage()
grid.draw(venn.plot)

## Lasso Stable genes selected

lasso_stability_df = as.data.frame(selected_genes_summary_lasso)
colnames(lasso_stability_df) <- c("Gene", "Frequency")


stable_lasso_df = subset(lasso_stability_df, Frequency >= 8)

stable_lasso_df = stable_lasso_df[order(stable_lasso_df$Frequency, decreasing = TRUE), ]


stable_lasso_df = stable_lasso_df[order(stable_lasso_df$Frequency, decreasing = TRUE), ]


ggplot(stable_lasso_df, aes(x = reorder(Gene, Frequency), y = Frequency, fill = Frequency)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Stable Lasso genes (Selected in    8 Folds)",
       x = "Gene",
       y = "Selection Frequency") +
  theme_minimal()

elastic_stability_df = as.data.frame(selected_genes_summary_elastic)

colnames(elastic_stability_df) = c("Gene", "Frequency")

stable_elastic_df = subset(elastic_stability_df, Frequency >= 8)

stable_elastic_df = stable_elastic_df[order(stable_elastic_df$Frequency, decreasing = TRUE), ]


stable_elastic_df = stable_elastic_df[order(stable_elastic_df$Frequency, decreasing = TRUE), ]


create_plot = function(data, title) {
  ggplot(data, aes(x = reorder(Gene, -Frequency), y = Frequency, fill = Frequency)) +
    geom_bar(stat = "identity") +
    theme_minimal() +
    labs(
      title = title,
      x = "Gene",
      y = "Selection Frequency"
    ) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_gradient(low = "green", high = "orange")
}


plot_list = list()
for (i in 1:8) {
  subset_data = stable_elastic_df[((i - 1) * 10 + 1):(i * 10), ]
  plot_title = paste("Selected genes", (i - 1) * 10 + 1, "-", i *
      10, "for Elastic Net")
  plot_list[[i]] = create_plot(subset_data, plot_title)
}


for (p in plot_list) {
  print(p)
}

ridge_stability_df = as.data.frame(selected_genes_summary_ridge)

colnames(ridge_stability_df) = c("Gene", "Frequency")


stable_ridge_df = subset(ridge_stability_df, Frequency >= 8)

stable_ridge_df = stable_ridge_df[order(stable_ridge_df$Frequency
    , decreasing = TRUE), ]


stable_ridge_df = stable_ridge_df[order(stable_ridge_df$Frequency
    , decreasing = TRUE), ]


plot_list = list()
for (i in 1:8) {
  subset_data <- stable_ridge_df[((i - 1) * 10 + 1):(i * 10), ]
  plot_title <- paste("Selected genes", (i - 1) * 10 + 1, "-", i
      * 10, "for Ridge Net")
  plot_list[[i]] <- create_plot(subset_data, plot_title)
}

for (p in plot_list) {
  print(p)
}
```

Listing 12: Stability Analysis

```


min_frequency = 8


stable_genes_lasso = names(selected_genes_summary_lasso[selected_
    genes_summary_lasso >= min_frequency])

stable_genes_lasso = setdiff(stable_genes_lasso, "(Intercept)")


X_train_stable_lasso = expression_train[, stable_genes_lasso,
    drop = FALSE]
```

```r

X_test_stable_lasso = expression_test[, stable_genes_lasso, drop
    = FALSE]


if (!identical(colnames(X_train_stable_lasso), colnames(X_test_
    stable_lasso))) {
  stop("Mismatch between training and testing genes.")
}


cv_fit_lasso_final = cv.glmnet(
  x = as.matrix(X_train_stable_lasso),
  y = expression_train$response,
  family = "binomial",
  alpha = 1
)


lasso_model = glmnet(
  x = as.matrix(X_train_stable_lasso),
  y = expression_train$response,
  family = "binomial",
  alpha = 1,
  lambda = cv_fit_lasso_final$lambda.min
)


pred_probs = predict(lasso_model,
                     newx = as.matrix(X_test_stable_lasso),
                     type = "response")


if (length(pred_probs) != length(response_test)) {
  stop("Mismatch between predicted probabilities and actual
      responses.")
}


pred_class = ifelse(pred_probs > 0.5, 1, 0)


confusion_matrix_lasso = table(Predicted = pred_class, Actual =
    response_test)
print(confusion_matrix)


stable_genes_ridge = names(selected_genes_summary_ridge[selected_
    genes_summary_ridge >= min_frequency])

stable_genes_ridge = setdiff(stable_genes_ridge, "(Intercept)")
```

```r


X_train_stable_ridge = expression_train[, stable_genes_ridge,
   drop = FALSE]

X_test_stable_ridge = expression_test[, stable_genes_ridge, drop
   = FALSE]


if (!identical(colnames(X_train_stable_ridge), colnames(X_test_
   stable_ridge))) {
  stop("Mismatch between training and testing genes.")
}


cv_fit_ridge_final = cv.glmnet(
  x = as.matrix(X_train_stable_ridge),
  y = expression_train$response,
  family = "binomial",
  alpha = 0
)


ridge_model = glmnet(
  x = as.matrix(X_train_stable_ridge),
  y = expression_train$response,
  family = "binomial",
  alpha = 0,
  lambda = cv_fit_ridge_final$lambda.min
)


pred_probs = predict(ridge_model,
                     newx = as.matrix(X_test_stable_ridge),
                     type = "response")


if (length(pred_probs) != length(response_test)) {
  stop("Mismatch between predicted probabilities and actual
     responses.")
}


pred_class = ifelse(pred_probs > 0.5, 1, 0)


confusion_matrix_ridge= table(Predicted = pred_class, Actual =
   response_test)
print(confusion_matrix)


```

```
stable_genes_elastic = names(selected_genes_summary_elastic[
   selected_genes_summary_elastic >= min_frequency])

stable_genes_elastic = setdiff(stable_genes_elastic, "(Intercept)
   ")


X_train_stable_elastic = expression_train[, stable_genes_elastic,
    drop = FALSE]

X_test_stable_elastic = expression_test[, stable_genes_elastic,
   drop = FALSE]


if (!identical(colnames(X_train_stable_elastic), colnames(X_test_
   stable_elastic))) {
  stop("Mismatch between training and testing genes.")
}


cv_fit_elastic_final = cv.glmnet(
  x = as.matrix(X_train_stable_elastic),
  y = expression_train$response,
  family = "binomial",
  alpha = 0.25
)


elastic_model = glmnet(
  x = as.matrix(X_train_stable_elastic),
  y = expression_train$response,
  family = "binomial",
  alpha = 0.25,
  lambda = cv_fit_elastic_final$lambda.min
)


pred_probs = predict(elastic_model,
                     newx = as.matrix(X_test_stable_elastic),
                     type = "response")


if (length(pred_probs) != length(response_test)) {
  stop("Mismatch between predicted probabilities and actual
     responses.")
}


pred_class = ifelse(pred_probs > 0.5, 1, 0)

```

```
confusion_matrix_elastic = table(Predicted = pred_class, Actual =
    response_test)
print(confusion_matrix)
```

Listing 13: Classification and Prediction using Stable genes

```


TN_lasso = confusion_matrix_lasso[1,1]

FN_lasso = confusion_matrix_lasso[1,2]

FP_lasso = confusion_matrix_lasso[2,1]

TP_lasso = confusion_matrix_lasso[2,2]

all_samples_lasso = TN_lasso + FN_lasso + FP_lasso + TP_lasso


### Accuracy

Accuracy_lasso = ((TN_lasso + TP_lasso)/ all_samples_lasso)*100




### Sensitivity

Sensitivity_lasso = ((TP_lasso)/ (TP_lasso + FN_lasso))*100




### Specificity

Specificity_lasso = ((TN_lasso)/ (TN_lasso + FP_lasso))*100




### Precision

Precision_lasso = ((TP_lasso)/ (TP_lasso + FP_lasso))*100




list(c(Accuracy_lasso = Accuracy_lasso, Sensitivity_lasso =
   Sensitivity_lasso, Specificity_lasso = Specificity_lasso,
   Precision_lasso = Precision_lasso ))


```

```
TN_ridge = confusion_matrix_ridge[1,1]

FN_ridge = confusion_matrix_ridge[1,2]

FP_ridge = confusion_matrix_ridge[2,1]

TP_ridge = confusion_matrix_ridge[2,2]




### Accuracy

Accuracy_ridge = ((TN_ridge + TP_ridge)/ (TN_ridge + TP_ridge +
    FN_ridge + FP_ridge) )*100




### Sensitivity

Sensitivity_ridge = ((TP_ridge)/ (TP_ridge + FN_ridge))*100




### Specificity

Specificity_ridge = ((TN_ridge)/ (TN_ridge + FP_ridge))*100




### Precision

Precision_ridge = ((TP_ridge)/ (TP_ridge + FP_ridge))*100




list(c(Accuracy_ridge = Accuracy_ridge, Sensitivity_ridge =
    Sensitivity_ridge, Specificity_ridge = Specificity_ridge,
    Precision_ridge = Precision_ridge ))


TN_elastic = confusion_matrix_ridge[1,1]

FN_elastic = confusion_matrix_ridge[1,2]

FP_elastic = confusion_matrix_ridge[2,1]

```

```
TP_elastic = confusion_matrix_ridge[2,2]




### Accuracy

Accuracy_elastic = ((TN_elastic + TP_elastic)/ (TN_elastic + TP_
    elastic + FN_elastic + FP_elastic) )*100




### Sensitivity

Sensitivity_elastic = ((TP_elastic)/ (TP_elastic + FN_elastic))*
    100




### Specificity

Specificity_elastic = ((TN_elastic)/ (TN_elastic + FP_elastic))*
    100




### Precision

Precision_elastic = ((TP_elastic)/ (TP_elastic + FP_elastic))*100




list(c(Accuracy_elastic = Accuracy_elastic, Sensitivity_elastic =
     Sensitivity_elastic, Specificity_elastic = Specificity_
    elastic, Precision_elastic = Precision_elastic ))


metrics_df = data.frame(
  Method = rep(c("Lasso", "Ridge", "Elastic Net"), each = 4),
  Metric = rep(c("Accuracy", "Sensitivity", "Specificity", "
     Precision"), times = 3),
  Value  = c(Accuracy_lasso, Sensitivity_lasso, Specificity_lasso
     , Precision_lasso,
             Accuracy_ridge, Sensitivity_ridge, Specificity_ridge
                , Precision_ridge,
             Accuracy_elastic, Sensitivity_elastic, Specificity_
                elastic, Precision_elastic)
)

ggplot(metrics_df, aes(x = Metric, y = Value, fill = Method)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  ylim(0, 100) +
  labs(title = "Performance Metrics Comparison for Lasso, Ridge,
     and Elastic Net",
       x = "Metric",
       y = "Percentage (%)") +
  scale_fill_brewer(palette = "Greens") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

Listing 14: Model Performance Evaluation

References

[Greenwood et al., 2020] Christopher J. Greenwood, George J. Youssef, Primrose Letcher, Jacqui A. Macdonald, Lauryn J. Hagg, Ann Sanson, Jenn Mcintosh, Delyse M. Hutchinson, John W. Toumbourou, Matthew Fuller-Tyszkiewicz, and Craig A. Olsson. *A comparison of penalised regression methods for informing the selection of predictive markers*. PLoS ONE, 15(10):e0241570, 2020. doi: 10.1371/journal.pone.0241570.

[Friedman et al., 2010] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. *Regularization Paths for Generalized Linear Models via Coordinate Descent*. Journal of Statistical Software, 33(1):1-22, 2010. doi: 10.18637/jss.v033.i01.

[Golub et al., 1999] Todd R. Golub, Donna K. Slonim, Pablo Tamayo, Christine Huard, Michael Gaasenbeek, Jill P. Mesirov, Hilary Coller, Michelle L. Loh, James R. Downing, Michael A. Caligiuri, Clara D. Bloomfield, and Eric S. Lander. *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*. Science, 286(5439):531–537, 1999. doi: 10.1126/science.286.5439.531.