**MODULE: 04**

# **“Regularization”**

**Course: ALY 6015 CRN 71547 Intermediate Analytics**

**Instructor: Dr. Paromita Guha (Paro)**

**Winter 2025 – February 12**

**College of Professional Studies**

**A black background with a black square

Description automatically generated with medium confidence**

**Content:**

1. Introduction
2. What is Regularization
3. Types of Regularization
4. Problem Statement
5. Assignment Overview
6. Data Processing & Preparation
7. Data Slitting
8. Ridge Regression
9. Model Performance
10. Lasso Regression
11. Model Performance
12. Visualization
13. Model & Feature Selection
14. Conclusion
15. Reference
16. Appendix

**Introduction**

In modern predictive modeling, regularization techniques like **Ridge Regression** and **LASSO Regression** are essential tools for handling multicollinearity, improving model interpretability, and enhancing predictive performance. These techniques impose penalties on the magnitude of regression coefficients to prevent overfitting and improve generalization. This assignment applies **regularized regression models** using the **glmnet** package in R to analyze the **College dataset** from the **ISLR library**. The objective is to build predictive models for **graduation rates (Grad.Rate)** by leveraging **Ridge and LASSO regressions** over varying values of the **regularization parameter (lambda)**, optimizing model selection through **cross-validation**, and evaluating model performance using **Root Mean Square Error (RMSE).**

**What is Regularization?**

**Regularization** is a technique used in machine learning and statistical modeling to **prevent overfitting** by adding a penalty to the model's complexity. It is especially useful in regression models where multicollinearity (high correlation between independent variables) or high dimensionality can lead to unstable and overly complex models that fail to generalize well to unseen data.

**Why is Regularization Needed?**

In standard regression models like **Ordinary Least Squares (OLS)**, the goal is to minimize the sum of squared errors. However, when there are **too many features** or **high correlations among variables**, the model may fit the training data too well, capturing noise rather than true patterns. This results in **overfitting**, where the model performs well on training data but poorly on test data.

Regularization helps by **constraining the model**, reducing the impact of less important features, and improving its **generalization** to new data.

**Types of Regularization**

1. **Ridge Regression (L2 Regularization)**
   * Adds a **penalty proportional to the square of the coefficients**:
   * Shrinks coefficients towards **zero** but **does not eliminate any variables**.
   * Works well when **all variables contribute some predictive power**.
2. **LASSO Regression (L1 Regularization)**
   * Adds a **penalty proportional to the absolute values of the coefficients**:
   * Can **shrink some coefficients to exactly zero**, effectively performing **feature selection**.
   * Useful when we expect **only a subset of features to be important**.
3. **Elastic Net Regression (Combination of L1 and L2)**
   * Combines both **Ridge (L2)** and **LASSO (L1)** penalties.
   * Useful when we have **high-dimensional data with correlated variables**.

**Problem Statement**

The primary challenge in regression-based predictive modeling arises from **multicollinearity and high-dimensionality**, which can lead to overfitting, unstable parameter estimates, and reduced interpretability. Traditional **Ordinary Least Squares (OLS) regression** fails to address these challenges, necessitating **regularization techniques** to achieve optimal bias-variance tradeoff. This assignment seeks to answer:

* How do Ridge and LASSO regression models perform in predicting **graduation rates** based on multiple predictor variables?
* How do the models differ in terms of coefficient shrinkage and variable selection?
* Which model generalizes better to unseen data based on **RMSE comparison**?
* Does Ridge or LASSO suffer from overfitting or underfitting in this dataset?

**Assignment Overview**

This assignment is structured into several key phases to systematically implement and evaluate Ridge and LASSO regression:

1. **Data Preprocessing and Splitting:** The College dataset is cleaned, standardized, and split into **training** and **test sets** to ensure unbiased model evaluation.
2. **Ridge Regression Implementation:** Using **cv.glmnet()**, optimal **lambda values (lambda.min and lambda.1se)** are determined, and Ridge regression is fitted on the training set. The model coefficients are analyzed, visualized, and evaluated using **RMSE on training and test sets**.
3. **LASSO Regression Implementation:** Similar to Ridge, **cv.glmnet()** is used to determine lambda values for LASSO. The model is fitted, and coefficients are examined for **sparsity (feature selection effect)**. The performance is measured using RMSE.
4. **Model Performance Evaluation:** A comparative analysis of Ridge and LASSO is conducted using **visualizations (coefficient bar charts, RMSE comparison plots)** and RMSE values. The impact of **bias-variance tradeoff** is discussed.
5. **Interpretation & Conclusion:** The findings are synthesized to determine the most effective model for predicting graduation rates, with insights into the suitability of Ridge vs. LASSO in different predictive scenarios.

By the end of this assignment, we will gain an advanced understanding of **regularization techniques, cross-validation tuning, model generalization, and feature selection strategies**, all critical for robust data science applications in high-dimensional predictive modeling

**Data Preprocessing and Preparation for Regularization Models**

In this assignment, we aim to implement **Ridge Regression** and **LASSO Regression** using the glmnet package in R. Before applying these regularization techniques, we must **preprocess the data**, ensuring it is in a suitable format for modeling. This section explains the data preparation process in detail, from loading the dataset to splitting it into training and test sets.

1. **Loading Required Libraries**

To execute regularized regression models and visualize results, we load essential R libraries:

* **ISLR** – Provides access to the **College dataset** from the "Introduction to Statistical Learning" book.
* **glmnet** – Enables Ridge and LASSO regression through efficient coordinate descent algorithms.
* **caret** – Supports data partitioning, resampling, and model training workflows.
* **ggplot2 & gridExtra** – Used for advanced data visualization and comparison of models.

1. **Loading the College Dataset**

The **College dataset** contains statistics for U.S. colleges, including **Graduation Rate (Grad.Rate)**, **Private/Public status**, **Acceptance Rate**, **Expenditures per Student**, and other institutional metrics. Since this dataset is available in the ISLR package, we load it directly:

# Load the College dataset

data(College)

1. **Converting Categorical Variables to Numeric**

The dataset contains a categorical variable:

**Private** (indicating whether a college is private or public).

Since Ridge and LASSO regression require **numeric inputs**, we convert this factor variable into a **binary numeric format** (1 = Private, 0 = Public):

1. **Setting Seed for Reproducibility**

To ensure that results remain **consistent** each time the script runs, we set a **random seed**:

# Set seed for reproducibility

set.seed(123)

This ensures that the random sampling process (such as train-test splitting) produces the same results every time the code is executed.

1. **Splitting the Data into Training and Testing Sets**

To evaluate model performance, we **split the dataset into training and test sets**:

**Training Set (70%)** – Used to train Ridge and LASSO models.

**Test Set (30%)** – Used to evaluate the model's generalization ability.

We use the caret’s createDataPartition() function to stratify the dataset based on **Graduation Rate (Grad.Rate)**, ensuring balanced representation:

# Split data into training and testing sets (70% train, 30% test)

trainIndex <- createDataPartition(College$Grad.Rate, p = 0.7, list = FALSE)

trainData <- College[trainIndex, ] # Training set (70%)

testData <- College[-trainIndex, ] # Test set (30%)

By ensuring a **70-30 split**, we create efficient data distribution for training and evaluation.

1. **Preparing Model Matrices for Ridge & LASSO Regression**

Unlike ordinary regression models, Ridge and LASSO regression require matrix input formats rather than standard data frames. The model.matrix() function helps transform the dataset into a numerical design matrix, automatically handling categorical variables.

We exclude the response variable (Grad.Rate) from the predictor matrix and remove the interception column ([, -1]) since glmnet automatically includes one:

# Prepare model matrices (excluding response variable)

x\_train <- model.matrix(Grad.Rate ~ ., trainData)[, # Remove intercept

y\_train <- trainData$Grad.Rate # Response variable for training set

x\_test <- model.matrix(Grad.Rate ~ ., testData)[, -1] # Remove intercept for test set

y\_test <- testData$Grad.Rate # Response variable for test set

1. **Why Use model.matrix()?**

* It **converts categorical variables** into dummy variables automatically.
* It creates a **numerical representation of predictors** for glmnet(), which requires matrix inputs.
* It **removes the intercept column** to prevent redundant calculations in regularized regression.

“**RIDGE REGRESSION”**

1. **Ridge Regression: Cross Validation.**

**A graph with a red line

Description automatically generated**

* 1. The red points represent the mean cross-validation errors, while the gray bars indicate the standard errors.
  2. The two vertical dashed lines correspond to lambda.min (the value of λ that gives the minimum cross-validation error) and lambda.1se (the most regularized model within one standard error of the minimum).
  3. The optimal λ (lambda.min) is the one that results in the lowest prediction error, while lambda.1se favors a simpler model by adding slightly more regularization.

Key Insights:

The MSE initially remains stable for small values of Log(λ) but starts increasing as Log(λ) grows beyond a certain point. This indicates that too much regularization leads to underfitting, as important predictors are shrunk excessively.

1. **Ridge Regression Coefficient Path.**

**A graph of a curve

Description automatically generated**

The plot "Ridge Regression Coefficient Paths," visualizes how different feature coefficients evolve as the regularization strength (Log Lambda) increases.

* 1. The y-axis represents the magnitude of the coefficients.
  2. The x-axis represents Log(λ), increasing from left to right.
  3. Each line corresponds to a predictor variable, showing how its coefficient shrinks as regularization increases.

Key Observations:

* + 1. At low values of Log(λ), the coefficients have larger magnitudes, meaning the model is less constrained.
    2. As Log(λ) increases, the coefficients shrink towards zero but never exactly reach zero (unlike LASSO regression).
    3. The coefficient for **"Private"** remains the dominant predictor with the highest absolute magnitude, indicating it has the strongest influence on predicting **Graduation Rate**.

**Impact:**  
This plot shows that Ridge Regression keeps all predictors in the model but shrinks their magnitudes, reducing multicollinearity and improving generalization.

1. **Ridge Regression Co-Efficient Bar Chart**

**A graph with a bar graph

Description automatically generated**

The third plot, titled "Ridge Regression Coefficients," presents a ranked visualization of the absolute magnitudes of the estimated coefficients.

1. The **"Private"** variable has the highest coefficient (~5.03), indicating that whether a college is private significantly influences the **Graduation Rate**.
2. Other notable predictors include **"perc.alumni"** (percentage of alumni donating), **"S.F.Ratio"** (student-to-faculty ratio), and **"Top25perc"** (percentage of new students from the top 25% of their high school class).
3. Many other predictors, such as **"Expend" (expenditures per student), "Books," and "Personal" expenses**, have near-zero coefficients, meaning their influence is minimal.

**Interpretation:**  
This plot confirms that Ridge Regression assigns importance to relevant predictors while suppressing less useful ones. However, unlike LASSO, Ridge does not eliminate any variables completely.

1. **Purpose and Impact of Ridge Regression Predictions**

**Purpose of Predictions**

The goal of these computations is to evaluate the predictive performance of Ridge Regression, a regularized linear regression method that reduces overfitting by penalizing large coefficients. By using the predict() function, we estimate graduation rates for both the training dataset and test dataset, providing insight into how well the model generalizes to unseen data.

**Interpretation of Predictions**

The printed outputs display Ridge Regression predictions for the first few universities. These values represent the estimated graduation rates based on various predictor variables such as student-faculty ratio, alumni donations, acceptance rates, and whether the institution is private or public.

A screen shot of a computer

Description automatically generated

A white background with black text

Description automatically generated

For example:

* Adelphi University is predicted to have a graduation rate of 62.69%,
* Agnes Scott College is predicted at 78.48%,
* Alaska Pacific University at 54.03%, etc.

These estimates provide a way to gauge expected student outcomes based on institutional characteristics.

1. **Evaluating Model Performance: RMSE Computation**

The **Root Mean Squared Error (RMSE)** is calculated for both the training and test datasets. RMSE measures the **average deviation between predicted values and actual graduation rates**, with lower values indicating better predictive accuracy.

A close-up of numbers

Description automatically generated

* **Training RMSE:** **12.98**
* **Test RMSE:** **12.04**

The relatively close values of **training and test RMSE** suggest that the Ridge Regression model is not overfitting and is generalizing well to new data.

**“LASSO Regression”**

1. **Lasso Regression-Cross Validation**

**A graph with red dots

Description automatically generated**

**Observation:**

* The plot shows the mean squared error (MSE) as a function of the logarithm of the regularization parameter (λ).
* The red dots indicate the MSE values for different λ values.
* The vertical dotted lines likely represent the optimal λ values based on cross-validation.

**Insights:**

* As λ increases, the MSE first stabilizes and then rises, indicating that too much regularization leads to underfitting.
* The optimal λ balances bias and variance to achieve the best predictive performance.

**Purpose:**

* The purpose is to determine the best λ value for LASSO regression, ensuring that the model avoids overfitting and underfitting.

**Impact:**

* Help in selecting an appropriate level of regularization to enhance model generalization.

1. **Lasso Regression Co-efficient Path**

**A graph of a log-lamda

Description automatically generated**

**Observation:**

* This plot visualizes the evolution of regression coefficients as λ changes.
* At high λ values, most coefficients shrink toward zero.
* As the number of features decreases, more features start getting nonzero coefficients.

**Insights:**

* LASSO eliminates less important features by shrinking their coefficients to zero.
* The most significant predictors maintain nonzero coefficients for a larger range of λ values.

**Purpose:**

* To visualize how LASSO performs feature selection by shrinking coefficients.

**Impact:**

1. Provides a clear understanding of which variables are most influential in the predictive model.
2. **Lasso Regression Coefficients**

**Insights:**

* LASSO has effectively performed feature selection by keeping only the most relevant predictors.
* The variable at the top has the strongest effect on the dependent variable.

**A graph with red squares

Description automatically generated**

**Observation:**

* A bar chart displaying the final selected LASSO regression coefficients.
* Only a few features have significant nonzero coefficients, while others are reduced to zero.
* The most dominant feature (at the top) has a much larger coefficient than the others.

**Purpose:**

* To display the final selected features and their impact on the model.

**Impact:**

* Helps in model interpretability by highlighting the key drivers of predictions.

1. **Explanation of PMSE Calculation for LASSO Regression**

Root Mean Squared Error (RMSE) measures the model’s prediction accuracy by calculating the square root of the average squared differences between actual and predicted values. It is a common metric for evaluating regression models.

**Breaking Down the Code**

lasso\_rmse\_train <- sqrt(mean((y\_train - lasso\_train\_pred)^2))

lasso\_rmse\_test <- sqrt(mean((y\_test - lasso\_test\_pred)^2))

* **y\_train**: The actual target values in the training set.
* **lasso\_train\_pred**: Predicted values from the LASSO model for the training set.
* **y\_test**: The actual target values in the test set.
* **lasso\_test\_pred**: Predicted values from the LASSO model for the test set.

**Steps:**

1. Compute the **difference** between actual (y\_train or y\_test) and predicted values (lasso\_train\_pred or lasso\_test\_pred).
2. Square these differences.
3. Compute the **meaning** of the squared differences.
4. Take the **square root** of this mean → **RMSE value**.

**Results Interpretation**

print(lasso\_rmse\_train)

[1] 12.94191

print(lasso\_rmse\_test)

[1] 11.98714

* Training RMSE = 12.94
* Test RMSE = 11.98

Since RMSE measures the average prediction error, these values indicate that the average error in predictions is around 12.94 for training data and 11.98 for test data.

**Insights & Impact**

1. **Lower Test RMSE than Train RMSE**
   * The model **generalizes well** because the test RMSE (11.98) is slightly lower than the training RMSE (12.94).
   * This suggests the model is **not overfitting** with the training data.
2. **LASSO’s Regularization Effect**

LASSO helps in **feature selection** by eliminating irrelevant predictors.

The small gap between training and test RMSE indicates that the model is **less prone to overfitting** compared to an ordinary least squares regression model.

1. **Prediction Visualization**
   1. Ridge Regression Plot:

 The purpose of this plot is to visualize how well the Ridge Regression model predicts the actual values. Ridge Regression is a technique used to analyze multiple regression data that suffer from multicollinearity. It adds a degree of bias to the regression estimates, which often results in lower mean squared error.

A graph with blue dots

Description automatically generated

* The plot shows a comparison between the predicted and actual values. The points are colored blue, and a dashed line with a slope of 1 is included to represent the ideal scenario where predicted values perfectly match actual values.
* If the blue points are closely aligned with the dashed line, it indicates that the Ridge model’s predictions are close to the actual values. Any significant deviation from the line suggests prediction errors.
* The model seems to perform well if the points are clustered around the dashed line. This suggests that the Ridge model is effectively handling multicollinearity and providing reliable predictions.
* The results suggest that the Ridge model is robust in handling data with multicollinearity, providing predictions that are close to the actual values.
* The impact of using Ridge Regression is a more stable and reliable model in the presence of multicollinearity, leading to better generalization on new data.
  1. Lasso Regression:

This plot aims to show the predictive performance of the LASSO Regression model. LASSO (Least Absolute Shrinkage and Selection Operator) not only helps in reducing overfitting but also performs feature selection by shrinking some coefficients to zero.

This plot compares predicted and actual values with points colored red. The dashed line again represents the ideal scenario

A graph with red dots

Description automatically generated

* The red points’ proximity to the dashed line indicates the accuracy of the LASSO model. Points far from the line indicate discrepancies between predicted and actual values.
* The clustering of red points around the dashed line indicates that the LASSO model is not only predicting accurately but also potentially simplifying the model by eliminating less important features.
* The results indicate that the LASSO model is effective in both prediction and feature selection, which can be particularly useful in high-dimensional datasets.
* The impact of using LASSO Regression is a more interpretable model due to feature selection, which can lead to simpler models that are easier to understand and deploy.

1. Displaying Plots:
   1. **grid.arrange(ridge\_coef\_plot, lasso\_coef\_plot, ncol = 2)**:
   2. This line arranges and displays the coefficient plots for Ridge and LASSO regression side by side in a grid layout with two columns.
   3. **Ridge Coefficient Plot**: Visualizes the magnitude of coefficients in the Ridge model. Ridge regression shrinks coefficients towards zero but does not set them exactly to zero, which helps in reducing model complexity and multicollinearity.
   4. **LASSO Coefficient Plot**: Shows the coefficients in the LASSO model. LASSO not only shrinks coefficients but can also set some of them to zero, effectively performing feature selection.

A comparison of a graph

Description automatically generated

* 1. **grid.arrange(ridge\_pred\_plot, lasso\_pred\_plot, ncol = 2)**:

This line displays the predicted versus actual value plots for both Ridge and LASSO models side by side.

* + 1. **Ridge Predicted vs. Actual Plot**: Compares the predicted values from the Ridge model against the actual values. The closer the points are to the 45-degree line, the better the model's predictions.
    2. **LASSO Predicted vs. Actual Plot**: Similarly, compares the predicted values from the LASSO model against the actual values. The proximity of points to the 45-degree line indicates prediction accuracy.
  1. **grid.arrange(ridge\_residual\_plot, lasso\_residual\_plot, ncol = 2)**:

This line arranges and displays the residual distribution plots for both models.

* + 1. **Ridge Residuals Distribution**: Shows the distribution of residuals (differences between actual and predicted values) for the Ridge model. Ideally, residuals should be normally distributed around zero.
    2. **LASSO Residuals Distribution**: Displays the distribution of residuals for the LASSO model. Like Ridge, a normal distribution around zero indicates good model performance.

**Printing RMSE:**

* **RMSE (Root Mean Squared Error)**: RMSE is a measure of the differences between predicted and actual values. Lower RMSE values indicate better model performance.
  + **Ridge RMSE (Train)**: 12.98
    - This value represents the RMSE for the Ridge model on the training data. It indicates the average deviation of the predicted values from the actual values in the training set.
  + **Ridge RMSE (Test)**: 12.04
    - This value represents the RMSE for the Ridge model on the test data. It shows how well the model generalizes unseen data.
  + **LASSO RMSE (Train)**: 12.94
    - This value represents the RMSE for the LASSO model on the training data.
  + **LASSO RMSE (Test)**: 11.99
    - This value represents the RMSE for the LASSO model on the test data.

**Insights and Interpretations:**

* **Model Performance**: Both Ridge and LASSO models show similar RMSE values on the training and test datasets, indicating comparable performance. However, LASSO has a slightly lower RMSE on the test data, suggesting it may generalize slightly better.
* **Residual Analysis**: The residual distributions for both models should be checked for normality. If residuals are normally distributed around zero, it suggests that the models are capturing the underlying patterns well without systematic errors.
* **Feature Selection**: LASSO's ability to set some coefficients to zero can lead to a more interpretable model by selecting only the most important features.

1. **Conclusion of the Assignment**

Based on the analysis and results, we can conclude the following:

**1. LASSO is the Selected Model**

* After evaluating different models, **LASSO Regression was chosen** because it performs both **feature selection** and **regularization**, reducing overfitting while maintaining prediction accuracy.
* The **RMSE values** for LASSO on training and test data indicate that the model generalizes well to unseen data:
  + **Training RMSE:** 12.94191
  + **Test RMSE:** 11.98714
  + Since the test RMSE is not significantly higher than the training RMSE, the model is not overfitting.

**2. Important Features Were Selected**

* LASSO automatically removed irrelevant features by shrinking their coefficients to zero.
* The selected important features (predictors) are the most influential in determining the target variable.
* This reduces model complexity while maintaining predictive performance.

**3. Impact of LASSO Feature Selection**

a) **Improved Interpretability**: The final model includes only the most relevant features, making it easier to understand.  
b) **Reduced Overfitting**: By eliminating unnecessary features, the model avoids noise and improves generalization.  
c) **Computational Efficiency**: Fewer features mean the model trains faster and is more efficient.  
d) **Better Predictions**: The RMSE values suggest a well-balanced model that performs reliably on new data.

**Reference:**

**Regulate Your Regression Model With Ridge, LASSO and ElasticNet**

<https://towardsdatascience.com/regulate-your-regression-model-with-ridge-lasso-and-elasticnet-92735e192e34/>

1. **Appendix**

**# Load necessary libraries**

**library(ISLR)**

**library(glmnet)**

**library(caret)**

**library(ggplot2)**

**library(gridExtra)**

**# Load the College dataset**

**data(College)**

**# Convert categorical variables to numeric**

**College$Private <- as.numeric(College$Private) # Convert factor to numeric**

**# Set seed for reproducibility**

**set.seed(123)**

**# Split data into training and testing sets (70% train, 30% test)**

**trainIndex <- createDataPartition(College$Grad.Rate, p = 0.7, list = FALSE)**

**trainData <- College[trainIndex, ]**

**testData <- College[-trainIndex, ]**

**# Prepare model matrices (excluding response variable)**

**x\_train <- model.matrix(Grad.Rate ~ ., trainData)[, -1] # Remove intercept**

**y\_train <- trainData$Grad.Rate**

**x\_test <- model.matrix(Grad.Rate ~ ., testData)[, -1]**

**y\_test <- testData$Grad.Rate**

**# ---------------- RIDGE REGRESSION ----------------**

**cv\_ridge <- cv.glmnet(x\_train, y\_train, alpha = 0) # alpha = 0 for Ridge**

**lambda\_min\_ridge <- cv\_ridge$lambda.min**

**lambda\_1se\_ridge <- cv\_ridge$lambda.1se**

**# Plot Ridge cross-validation error**

**plot(cv\_ridge, main = "Ridge Regression: Cross-Validation")**

**# Fit Ridge regression model using lambda.min**

**ridge\_model <- glmnet(x\_train, y\_train, alpha = 0)**

**# Ridge Coefficient Paths**

**plot(ridge\_model, xvar = "lambda", label = TRUE, main = "Ridge Regression Coefficient Paths")**

**# Get coefficients for Ridge at lambda.min**

**ridge\_coeffs <- coef(glmnet(x\_train, y\_train, alpha = 0, lambda = lambda\_min\_ridge))**

**# Convert to a dataframe for visualization**

**ridge\_coef\_df <- data.frame(Feature = rownames(ridge\_coeffs), Coefficient = as.vector(ridge\_coeffs))[-1, ] # Remove intercept**

**ridge\_coef\_df <- ridge\_coef\_df[order(abs(ridge\_coef\_df$Coefficient), decreasing = TRUE), ]**

**# Ridge Coefficient Bar Chart**

**print(ridge\_coef\_df)**

**ridge\_coef\_df$Coefficient <- as.numeric(ridge\_coef\_df$Coefficient)**

**library(ggplot2)**

**ridge\_coef\_plot <- ggplot2::ggplot(ridge\_coef\_df, aes(x = reorder(Feature, Coefficient), y = Coefficient)) +**

**geom\_bar(stat = "identity", fill = "pink", color = "black") +**

**coord\_flip() +**

**ggtitle("Ridge Regression Coefficients") +**

**theme\_minimal()**

**print(ridge\_coef\_plot)**

**# Predict on training and test data**

**ridge\_train\_pred <- predict(glmnet(x\_train, y\_train, alpha = 0, lambda = lambda\_min\_ridge), newx = x\_train)**

**ridge\_test\_pred <- predict(glmnet(x\_train, y\_train, alpha = 0, lambda = lambda\_min\_ridge), newx = x\_test)**

**print(head(ridge\_train\_pred))**

**print(head(ridge\_test\_pred))**

**# Compute RMSE for Ridge**

**ridge\_rmse\_train <- sqrt(mean((y\_train - ridge\_train\_pred)^2))**

**ridge\_rmse\_test <- sqrt(mean((y\_test - ridge\_test\_pred)^2))**

**print(ridge\_rmse\_train)**

**print(ridge\_rmse\_test)**

**# ---------------- LASSO REGRESSION ----------------**

**cv\_lasso <- cv.glmnet(x\_train, y\_train, alpha = 1) # alpha = 1 for LASSO**

**lambda\_min\_lasso <- cv\_lasso$lambda.min**

**lambda\_1se\_lasso <- cv\_lasso$lambda.1se**

**# Plot LASSO cross-validation error**

**plot(cv\_lasso, main = "LASSO Regression: Cross-Validation")**

**# Fit LASSO regression model using lambda.min**

**lasso\_model <- glmnet(x\_train, y\_train, alpha = 1)**

**# LASSO Coefficient Paths**

**plot(lasso\_model, xvar = "lambda", label = TRUE, main = "LASSO Regression Coefficient Paths")**

**# Get coefficients for LASSO at lambda.min**

**lasso\_coeffs <- coef(glmnet(x\_train, y\_train, alpha = 1, lambda = lambda\_min\_lasso))**

**# Convert to a dataframe for visualization**

**lasso\_coef\_df <- data.frame(Feature = rownames(lasso\_coeffs), Coefficient = as.vector(lasso\_coeffs))[-1, ] # Remove intercept**

**lasso\_coef\_df <- lasso\_coef\_df[order(abs(lasso\_coef\_df$Coefficient), decreasing = TRUE), ]**

**# LASSO Coefficient Bar Chart**

**lasso\_coef\_plot <- ggplot(lasso\_coef\_df, aes(x = reorder(Feature, Coefficient), y = Coefficient)) +**

**geom\_bar(stat = "identity", fill = "red", color = "black") +**

**coord\_flip() +**

**ggtitle("LASSO Regression Coefficients") +**

**theme\_minimal()**

**print(lasso\_coef\_plot)**

**# Predict on training and test data**

**lasso\_train\_pred <- predict(glmnet(x\_train, y\_train, alpha = 1, lambda = lambda\_min\_lasso), newx = x\_train)**

**lasso\_test\_pred <- predict(glmnet(x\_train, y\_train, alpha = 1, lambda = lambda\_min\_lasso), newx = x\_test)**

**print(lasso\_train\_pred)**

**print(lasso\_test\_pred)**

**# Compute RMSE for LASSO**

**lasso\_rmse\_train <- sqrt(mean((y\_train - lasso\_train\_pred)^2))**

**lasso\_rmse\_test <- sqrt(mean((y\_test - lasso\_test\_pred)^2))**

**print(lasso\_rmse\_train)**

**print(lasso\_rmse\_test)**

**# ---------------- Prediction Visualization ----------------**

**ridge\_pred\_plot <- ggplot(data.frame(Actual = y\_test, Predicted = as.vector(ridge\_test\_pred)), aes(x = Actual, y = Predicted)) +**

**geom\_point(color = "blue") +**

**geom\_abline(slope = 1, intercept = 0, linetype = "dashed") +**

**ggtitle("Ridge: Predicted vs. Actual") +**

**theme\_minimal()**

**print(ridge\_pred\_plot)**

**lasso\_pred\_plot <- ggplot(data.frame(Actual = y\_test, Predicted = as.vector(lasso\_test\_pred)), aes(x = Actual, y = Predicted)) +**

**geom\_point(color = "red") +**

**geom\_abline(slope = 1, intercept = 0, linetype = "dashed") +**

**ggtitle("LASSO: Predicted vs. Actual") +**

**theme\_minimal()**

**print(lasso\_pred\_plot)**

**# ---------------- Residual Plots ----------------**

**ridge\_residuals <- data.frame(Residuals = y\_test - as.vector(ridge\_test\_pred))**

**lasso\_residuals <- data.frame(Residuals = y\_test - as.vector(lasso\_test\_pred))**

**ridge\_residual\_plot <- ggplot(ridge\_residuals, aes(x = Residuals)) +**

**geom\_histogram(binwidth = 5, fill = "lavender", color = "black") +**

**ggtitle("Ridge Residuals Distribution") +**

**theme\_minimal()**

**print(ridge\_residual\_plot)**

**lasso\_residual\_plot <- ggplot(lasso\_residuals, aes(x = Residuals)) +**

**geom\_histogram(binwidth = 5, fill = "cyan", color = "black") +**

**ggtitle("LASSO Residuals Distribution") +**

**theme\_minimal()**

**print(lasso\_residual\_plot)**

**# ---------------- Display All Plots ----------------**

**grid.arrange(ridge\_coef\_plot, lasso\_coef\_plot, ncol = 2)**

**grid.arrange(ridge\_pred\_plot, lasso\_pred\_plot, ncol = 2)**

**grid.arrange(ridge\_residual\_plot, lasso\_residual\_plot, ncol = 2)**

**# ---------------- Print RMSE ----------------**

**print(paste("Ridge RMSE (Train):", ridge\_rmse\_train))**

**print(paste("Ridge RMSE (Test):", ridge\_rmse\_test))**

**print(paste("LASSO RMSE (Train):", lasso\_rmse\_train))**

**print(paste("LASSO RMSE (Test):", lasso\_rmse\_test))**

“End of Module 5”