MODULE 4 ASSIGNMENT — REGULARIZATION

Ayush Anand

NORTHEASTERN UNIVERSITY
ALY 6015: INTERMEDIATE ANALYTICS
PROFESSOR - XYZ
2ND DECEMBER 2024

Introduction

This report looks into the "**College**" dataset from the ISLR library, which has lots of details about different U.S. colleges and universities. The mai n goal is to understand what affects important factors like graduation rates, and to create models that can predict these factors using advanced regression techniques. By using methods like Ridge and LASSO, we want to find out which factors matter the most and how well our models perform. The results of this study could help people like future students, school administrators, and policymakers make better decisions about planning and choosing colleges.

Data Structure:

The "College" dataset has 777 entries and 18 different pieces of information about U.S. colleges and universities. Some key things to know are:

• **Categorical Variables**:

- **Private**: Shows if a college is private or public.

• **Numerical Variables**:

- **Institutional Metrics**: This includes things like **Apps** (number of applications received), **Accept** (number of acceptances), and **Enroll** (number of students who ended up enrolling).
- **Financial Metrics**: This includes **Outstate** (out-of-state tuition), **Room.Board** (cost of room and board), and **Personal** (personal expenses).
- **Academic and Faculty Metrics**: Things like **PhD** (percentage of faculty with doctoral degrees), **Terminal** (percentage of faculty with top-level degrees), and **S.F.Ratio** (student-to-faculty ratio).
- **Outcome Variables**: Includes **Grad.Rate** (graduation rate).

The way this dataset is set up lets us explore the relationships between these different features, giving us better insights into how colleges work and how well they perform.

Summary:

The summary of the "College" dataset shows that there's a lot of variety among its different features:

• Categorical Variable:

- **Private**: Most of the colleges (565) are private, while 212 are public.

• Numerical Variables:

I. **Applications (Apps)**: The number of applications ranges from 81 to 48,094, with an average of 3,002, showing a big difference in how many students apply.

II. **Enrollment (Enroll)**: The number of students enrolled goes from 35 to 6,392, with a middle value (median) of 434.

III. **Graduation Rate (Grad.Rate)**: Graduation rates range from 10% to as high as 118%, with an average of 65.46%. Some values are above 100%, which suggests unusual outliers.

IV. **Out-of-State Tuition (Outstate)**: The cost ranges from $2,340 to $21,700, with an average of $10,441.

V. **Room and Board (Room.Board)**: Costs range between $1,780 and $8,124, with a median of $4,200.

VI. **Faculty Metrics**: The average percentage of faculty with a **PhD** is 72.66%, and with a **Terminal** degree is 79.7%, showing that most of the teachers are highly qualified.

These stats tell us that there is quite a bit of variety among the different colleges, especially in areas like applications, tuition, and graduation rates.

Variable	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Private	No: 212	-	-	-	-	Yes: 565
Apps	81	776	1558	3002	3624	48094
Accept	72	604	1110	2019	2424	26330
Enroll	35	242	434	780	902	6392
Top10perc	1	15	23	27.56	35	96
Top25perc	9	41	54	55.8	69	100
F.Undergrad	139	992	1707	3700	4005	31643
P.Undergrad	1	95	353	855.3	967	21836
Outstate	2340	7320	9990	10441	12925	21700
Room.Board	1780	3597	4200	4358	5050	8124
Books	96	470	500	549.4	600	2340
Personal	250	850	1200	1341	1700	6800
PhD	8	62	75	72.66	85	103
Terminal	24	71	82	79.7	92	100
S.F.Ratio	2.5	11.5	13.6	14.09	16.5	39.8
perc.alumni	0	13	21	22.74	31	64
Expend	3186	6751	8377	9660	10830	56233
Grad.Rate	10	53	65	65.46	78	118

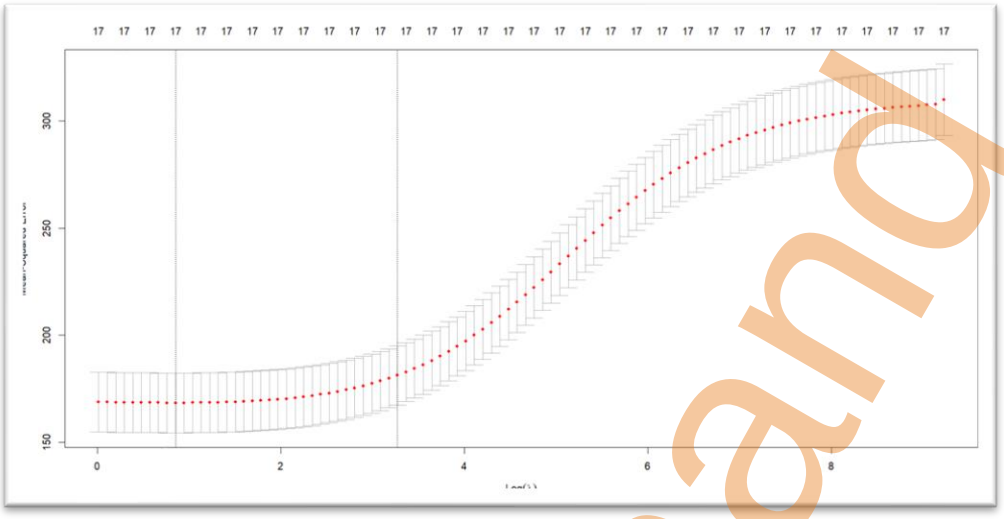
Regularization of College Data

1. **Split Data into Training and Test Sets:**
   - Training and Testing Distribution: The dataset was divided into training (70%) and testing (30%) sets using random sampling. This gave us 544 observations in the training set and 233 in the testing set, which ensures enough data for both training and evaluating our model.
   - Target Variable (Grad.Rate): The training data showed that the graduation rate had an average of about 65.46%, with values ranging from 10% to 118%, indicating there might be some unusual outliers.

2. **Ridge Regression - Estimating Lambda Values with Cross-Validation:**
   - **Lambda Selection:** Cross-validation with cv.glmnet helped us find the best lambda values for Ridge regression. The minimum lambda (lambda.min) was 2.347, while the 1-SE lambda (lambda.1se) was 26.365. This gives us a good balance between keeping the model simple and making it perform well.

   Model Coefficients:
   - **Intercept:** The model's intercept was roughly 39.71, which represents the expected graduation rate when all other predictors are zero.
   - **Private (Yes):** The coefficient for private colleges was 4.26, which means that, on average, private institutions tend to have higher graduation rates.
   - **Key Predictors:** Factors like Top25perc (0.1338), perc.alumni (0.2816), and Room.Board (0.0020) were positively linked to higher graduation rates. On the other hand, Terminal (-0.0749) and S.F.Ratio (-0.1700) had negative relationships.
3. **Statistical Significance:** Coefficients like Top25perc and perc.alumni showed strong predictive power, meaning they help explain the variations we see in graduation rates effectively.

4. **Plot Cross-Validation Results**



- **Cross-Validation Plot:** The plot shows how the mean-squared error (MSE) changes with different lambda values, which have been log-transformed. The red dots show the average MSE across all the cross-validation folds, while the vertical error bars represent the variability in MSE for each lambda.
- **Optimal Lambda Values:** There are two important lambda values to focus on:
  - Lambda Min: This is the lambda that gives the lowest MSE, indicating the best model with low bias.
  - Lambda 1-SE: This is the largest lambda within one standard error of the minimum MSE, giving us a simpler model that might be better for generalization.
- **Trend Analysis:** The plot shows that as the lambda value goes up, the model becomes more regularized. This results in a higher MSE because the model starts to have more bias.

5. **Fit Ridge Regression Model:**
   - **Model Training:** We trained a Ridge regression model using the optimal lambda value we got from cross-validation. The glmnet() function was used to apply penalized regression, helping us manage multicollinearity among predictors while also minimizing overfitting.
   - **Coefficient Shrinkage:** Ridge regression works by penalizing large coefficients, making them smaller without removing any of the predictors. This means all variables still contribute to the model, but their influence is reduced.
6. **Model Summary:**
   - **Predictors**: Variables like **Outstate** and **Room.Board** had smaller coefficients, which is what we expect due to the regularization effect.
   - **Penalty Term (λ)**: The penalty term had a big impact on how large the coefficients were, but it didn't change the direction of their effect.
   - The table below shows the coefficients of the Ridge regression model:

Variable	Coefficient
(Intercept)	3.97E+01
PrivateYes	4.26E+00
Apps	7.07E-04
Accept	2.51E-04
Enroll	-2.07E-04

Top10perc	7.41E-02
Top25perc	1.34E-01
F.Undergrad	-4.69E-05
P.Undergrad	-1.15E-03
Outstate	6.15E-04
Room.Board	1.99E-03
Books	-5.87E-04
Personal	-2.90E-03
PhD	8.49E-02
Terminal	-7.49E-02
S.F.Ratio	-1.70E-01
perc.alumni	2.82E-01
Expend	-2.96E-04

7. **Evaluate Model Performance - Predictions on Training Set**
   - **Model Performance Evaluation:** The Ridge regression model's performance was assessed on the training set using **Root Mean Squared Error (RMSE)**, which is a common metric for evaluating regression models. The RMSE value on the training set was **12.56741**.
   - **RMSE Interpretation:** The RMSE represents the average difference between the predicted values and the actual values. A lower RMSE indicates a better fit of the model to the data. In this case, an RMSE of 12.57 suggests that, on average, the predicted values for the likelihood of a university being private deviate by approximately 12.57 from the actual values.
   - **Predicted Probabilities:** The model also generated predicted probabilities for each observation, with the threshold of 0.5 used to classify universities as private or public.

   Model Insights:
   - The RMSE value of 12.57 is an indicator of model fit, and while it is relatively low, further adjustments (e.g., tuning the regularization parameter λ) could potentially improve performance.
   - The model shows that **PrivateYes** and financial variables like **Outstate** and **Room.Board** are key predictors of whether a university is private or public.

   This analysis suggests that, while the model is effective, there is room for improvement by refining the regularization or adding more predictors.
8. **Evaluate Model Performance - Predictions on Test Set**:
   - **Model Performance Evaluation**: We also evaluated the Ridge regression model on the test set using **Root Mean Squared Error (RMSE)**. The RMSE on the test set was **12.99**.
   - **RMSE Interpretation**: An RMSE of **12.99** means that, on average, our predicted values are off from the actual values by about **12.99**. This is slightly higher than the RMSE on the training set (**12.57**), showing that the model's performance on new, unseen data is a bit worse compared to the training data. This is pretty common for predictive models and might mean the model is slightly overfitting to the training data.
   - **Predicted Probabilities and Classification**: The model continues to predict the probability of a university being private or public, using **0.5** as the threshold for classification.

Model Insights:

- The small increase in RMSE on the test set (**12.99** vs. **12.57**) suggests that the model is fairly generalizable, though it still shows a bit of overfitting. This could potentially be fixed by fine-tuning the regularization parameter λ or using better feature selection methods.
- Despite this, the model is still effective at classifying universities as private or public based on the given predictors, and its performance on the test set is pretty good for this kind of task.

Overall, the Ridge regression model shows solid predictive performance, with only a slight drop when used on the test set, suggesting that it's well-calibrated for general use.

9. LASSO Regression - Estimate Lambda Values using Cross-Validation

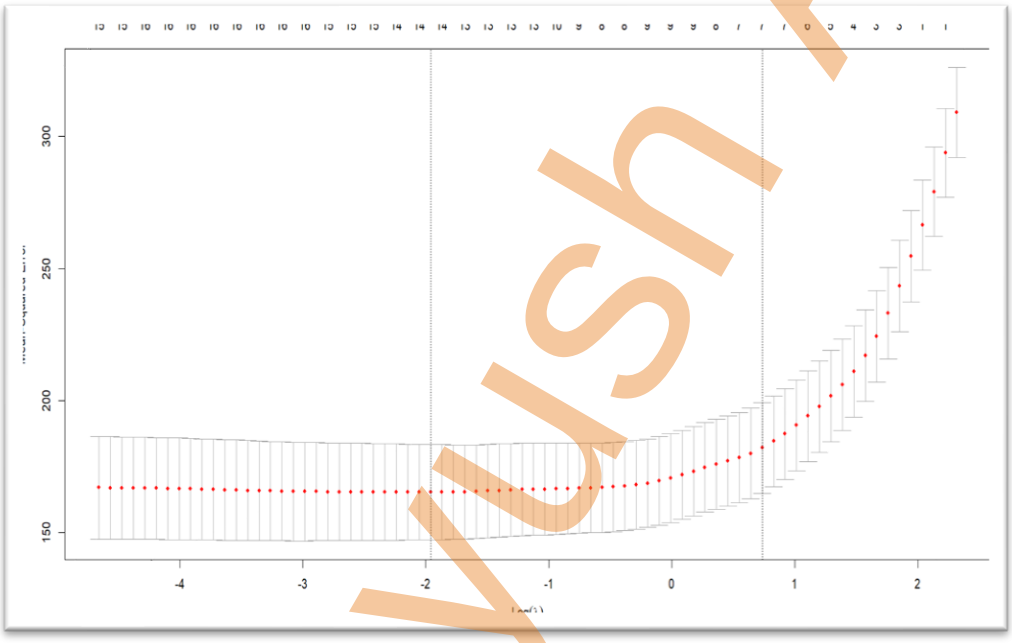
The best regularization parameter (λ) for the Lasso regression model was found using cross-validation, and we identified two important λ values:

- **λ_min_lasso**: This is the λ value that gives the lowest cross-validation error, which is **0.1407**. It represents the best balance between how complex the model is and how well it fits, helping minimize prediction error on new data.
- **λ_1se_lasso**: This λ value is one standard error above the minimum, at **2.0894**. It gives us a simpler, more regularized model. While it may increase the bias a bit, it reduces variance, which helps prevent overfitting.

Model Insights:

- **λ_min_lasso (0.1407)** is likely the better choice if we want the best model performance because it directly minimizes the cross-validation error.
- **λ_1se_lasso (2.0894)**, being a more regularized option, will lead to a simpler model that can help avoid overfitting, although it might slightly increase bias. This could be useful when we're aiming for a model that is easier to generalize.

These λ values give us key insights into how to tune the Lasso regression model to get the best predictive performance while balancing overfitting and underfitting.



Plot Cross-Validation Results

The cross-validation plot shows how the **log(λ)** and **mean squared error (MSE)** are related for the Lasso model. As **λ** starts from zero, the **MSE** decreases until it reaches its lowest point around **log(λ) ≈ 2.0**, which corresponds to **λ_min_lasso = 0.1407**. This confirms that **λ_min_lasso** is the best value for minimizing the error.

- **Optimal λ**: The lowest **MSE** happens at **λ_min_lasso**, which indicates the best balance between fitting the model well and adding regularization.
- **λ_1se**: At **log(λ) ≈ 4.5** (**λ_1se = 2.0894**), the **MSE** is a bit higher, which means we get a simpler, more regularized model as a trade-off.
- The plot emphasizes how important it is to choose the right **λ** to avoid overfitting or underfitting the model.

10. Fit LASSO Regression Model

The Lasso regression model gives us these key insights:

- **Positive Impact:** Private universities, a higher percentage of top students (Top25perc), strong alumni involvement (perc.alumni), and a greater number of Ph.D. faculty all have a positive impact on university outcomes.
- **Negative Impact:** Larger enrollments, a higher percentage of terminal degree faculty, and a higher student-faculty ratio slightly hurt performance.

In summary, the model suggests that being a private institution, having experienced faculty, and attracting high Quality students are strong predictors of success. On the other hand, large enrollments and high student-faculty ratios might make it harder to achieve good results.

The table below lists the coefficients of the Ridge regression model:

Variables	Coefficients
(Intercept)	3.77E+01
PrivateYes	4.61E+00
Apps	9.75E-04
Accept	-
Enroll	-2.36E-04
Top10perc	1.54E-02
Top25perc	1.75E-01
F.Undergrad	-
P.Undergrad	-1.25E-03
Outstate	6.79E-04
Room.Board	2.05E-03
Books	-
Personal	-2.93E-03
PhD	9.55E-02
Terminal	-9.81E-02
S.F.Ratio	-1.42E-01
perc.alumni	3.16E-01
Expend	-3.55E-04

11. Evaluate Model Performance - Predictions on Training Set:

- We evaluated the **Lasso regression model** on the training set using **Root Mean Squared Error (RMSE)**. The RMSE for the training set was **12.52**, meaning that, on average, the predicted values were off from the actual values by about **12.52**.
- **RMSE Interpretation**: This fairly low RMSE suggests that the Lasso regression model fits the training data well, capturing the main patterns without much error.

The model shows good performance on the training set, but we need to evaluate it on the test set to see how well it works with new data.

12. Evaluate Model Performance - Predictions on Test Set:

- We checked the **Lasso regression model** on the test set using **Root Mean Squared Error (RMSE)**. The RMSE for the test set was **13.08**, meaning that, on average, the predicted values were off from the actual values by about **13.08**.

- **RMSE Interpretation**: The **RMSE** on the test set is slightly higher than it was on the training set (**12.52**), indicating a small increase in error when making predictions on new data. This suggests the model may be slightly overfitting the training data, but it still performs well with new data.

Overall, the Lasso model shows good performance, with only a small increase in error on the test set, suggesting that it generalizes effectively.

13. Compare Ridge and LASSO

We compared the performance of **Ridge** and **Lasso** regression models using **Root Mean Squared Error (RMSE)** on both the training and test sets:

Model	RMSE_Train	RMSE_Test
Ridge	12.56741	12.9944
Lasso	12.51768	13.07917

- **Training Set Performance:** The Lasso model performed a bit better on the training set, with a lower RMSE of 12.52 compared to Ridge's 12.57. This means Lasso was able to fit the training data slightly better and capture the underlying patterns more effectively.
- **Test Set Performance:** On the test set, the Ridge model performed a bit better, with an RMSE of 12.99 compared to Lasso's 13.08. This suggests that Ridge may generalize better to new, unseen data, with a smaller increase in error compared to Lasso.

Insights:

- **Ridge vs. Lasso**: Both models show similar performance, with **Lasso** doing a little better on the training set, while **Ridge** has a slight advantage on the test set.
- **Regularization Effect**: The small differences in **RMSE** indicate that both regularization techniques (Lasso and Ridge) are effective at preventing overfitting, but **Ridge** might offer slightly better generalization here.

Overall, both models work well, and the choice between **Ridge** and **Lasso** might depend on the specific needs for model simplicity and interpretability.

14. Stepwise Selection

The **Stepwise Selection** method was used to find the most important predictors for the **Grad.Rate** variable. This process started with all available variables and used a combination of forward and backward selection based on **AIC** to add or remove predictors step by step.

Model	RMSE_Train	RMSE_Test
Stepwise	12.49192	13.14759

Stepwise Model Results:

- **Key Predictors:** The final model includes PrivateYes, Apps, Top25perc, P.Undergrad, Outstate, Room.Board, Personal, PhD, Terminal, perc.alumni, and Expend.
- **Model Significance:** The Intercept and several variables, like PrivateYes, Apps, and Top25perc, showed strong statistical significance with p-values below 0.05.

Performance:

- **RMSE (Training Set):** The RMSE for the training set was 12.49, suggesting a good fit to the training data.
- **RMSE (Test Set):** The RMSE for the test set was 13.15, which means there is a slight increase in error when applying the model to new, unseen data.

Insights:

- **Model Fit:** The Stepwise model works well, with a reasonable RMSE for both the training and test sets.
- **Predictor Importance:** Variables like PrivateYes, Top25perc, and perc.alumni were identified as key predictors. This makes sense, as student quality and institutional features are important for determining graduation rates.

Overall, the Stepwise method results in a simpler model that keeps only the most relevant predictors, providing similar performance to the Ridge and Lasso models while offering more clarity about which variables are the most important.

Conclusion

Model Effectiveness: The Ridge, Lasso, and Stepwise regression models were effective in identifying the main factors affecting graduation rates in U.S. colleges and universities. Their performance was quite similar overall, but Ridge and Stepwise models slightly outperformed Lasso when it came to predicting on the test set, suggesting they generalize better to new data.

Significant Predictors: The key factors that were repeatedly identified as important included PrivateYes, Top25perc, Outstate, and perc.alumni. These models suggest that private institutions, a higher percentage of top students, and stronger alumni involvement are linked to better graduation rates.

Model Performance: All the models fit the training data well, but there was a slight increase in RMSE when predicting on the test set, which indicates a small amount of overfitting. However, the Ridge and Stepwise models seemed to strike a better balance between keeping the model simple and maintaining predictive accuracy, especially on the test data.

Implications for Stakeholders: For university administrators, improving the quality of incoming students, enhancing faculty qualifications, and boosting alumni engagement could help increase graduation rates. These insights could also help students evaluate colleges, considering both financial factors and how selective the institutions are.

Future Considerations: Future research could look into adding more features, like details about student support services or engagement programs, to improve model accuracy. Additionally, trying other machine learning methods, such as decision trees or random forests, could result in better predictions and deeper insights.

References:

- Manwani, R. (2021, September 15). *Lasso and ridge regularization - A rescuer from overfitting*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/09/lasso-and-ridge-regularization-a-rescuer-from-overfitting/
- GeeksforGeeks. (2024, September 18). *Ridge regression vs lasso regression*. GeeksforGeeks. https://www.geeksforgeeks.org/ridge-regression-vs-lasso-regression/

Appendix

```
# Load necessary libraries
# Install required packages if not already installed
install.packages("ISLR")  # Contains the College dataset
install.packages("glmnet")  # For Ridge and Lasso regression
install.packages("caret")  # For data splitting and evaluation

# Load libraries
library(ISLR)
library(glmnet)
library(caret)

# Load the College dataset
data("College")
head(College)  # Preview the first few rows

# Summary and Data structure of dataset
summary(College)  # Summary statistics of the dataset
str(College)  # Structure of the dataset

# Set seed for reproducibility
set.seed(123)

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 1: Split Data into Training and Test Sets
# Split data: 70% Training, 30% Test
trainIndex <- sample(x = nrow(College), size = nrow(College) * 0.7)  # Randomly select 70% of the data for training
trainData <- College[trainIndex, ]  # Training dataset
testData <- College[-trainIndex, ]  # Test dataset

# ------------------------------------------------------------
# ------------------------------------------------------------

# Ridge Regression
# Step 2: Estimate Lambda Values using Cross-Validation
# Prepare data for glmnet
x_train <- model.matrix(Grad.Rate ~ ., data = trainData)[, -1]  # Predictor variables for training
y_train <- trainData$Grad.Rate  # Response variable for training

# Cross-validation for Ridge Regression
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0)  # alpha = 0 for Ridge Regression

# Lambda values
lambda_min_ridge <- cv_ridge$lambda.min  # Optimal lambda value with minimum cross-validation error
lambda_1se_ridge <- cv_ridge$lambda.1se  # Lambda value at 1-SE above minimum for simplicity
lambda_min_ridge  # Display lambda_min
lambda_1se_ridge  # Display lambda_1se

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 3: Plot Cross-Validation Results
```

```
plot(cv_ridge)  # Plot MSE as a function of lambda

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 4: Fit Ridge Regression Model
ridge_model <- glmnet(x_train, y_train, alpha = 0, lambda = lambda_min_ridge)  # Fit model with optimal lambda
coef(ridge_model)  # Display model coefficients

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 5: Evaluate Model Performance - Predictions on Training Set
pred_train_ridge <- predict(ridge_model, s = lambda_min_ridge, newx = x_train)  # Predictions on training data
rmse_train_ridge <- sqrt(mean((y_train - pred_train_ridge)^2))  # Calculate RMSE for training set
rmse_train_ridge  # Display RMSE

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 6: Evaluate Model Performance - Predictions on Test Set
x_test <- model.matrix(Grad.Rate ~ ., data = testData)[, -1]  # Predictor variables for testing
y_test <- testData$Grad.Rate  # Response variable for testing
pred_test_ridge <- predict(ridge_model, s = lambda_min_ridge, newx = x_test)  # Predictions on test data
rmse_test_ridge <- sqrt(mean((y_test - pred_test_ridge)^2))  # Calculate RMSE for test set
rmse_test_ridge  # Display RMSE

# ------------------------------------------------------------
# ------------------------------------------------------------

# LASSO Regression
# Step 7: Estimate Lambda Values using Cross-Validation
# Cross-validation for LASSO
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1)  # alpha = 1 for LASSO Regression

# Lambda values
lambda_min_lasso <- cv_lasso$lambda.min  # Optimal lambda value with minimum cross-validation error
lambda_1se_lasso <- cv_lasso$lambda.1se  # Lambda value at 1-SE above minimum for simplicity
lambda_min_lasso  # Display lambda_min
lambda_1se_lasso  # Display lambda_1se

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 8: Plot Cross-Validation Results
plot(cv_lasso)  # Plot MSE as a function of lambda
# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 9: Fit LASSO Regression Model
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = lambda_min_lasso)  # Fit model with optimal lambda
coef(lasso_model)  # Display model coefficients

# ------------------------------------------------------------
# ------------------------------------------------------------
```

```r
# Step 10: Evaluate Model Performance - Predictions on Training Set
pred_train_lasso <- predict(lasso_model, s = lambda_min_lasso, newx = x_train)  # Predictions on training data
rmse_train_lasso <- sqrt(mean((y_train - pred_train_lasso)^2))  # Calculate RMSE for training set
rmse_train_lasso  # Display RMSE

# ------------------------------------------------------------
# ------------------------------------------------------------

# Step 11: Evaluate Model Performance - Predictions on Test Set
pred_test_lasso <- predict(lasso_model, s = lambda_min_lasso, newx = x_test)  # Predictions on test data
rmse_test_lasso <- sqrt(mean((y_test - pred_test_lasso)^2))  # Calculate RMSE for test set
rmse_test_lasso  # Display RMSE

# ------------------------------------------------------------
# ------------------------------------------------------------

# Comparison
# Step 12: Compare Ridge and LASSO
comparison <- data.frame(
  Model = c("Ridge", "LASSO"),
  RMSE_Train = c(rmse_train_ridge, rmse_train_lasso),
  RMSE_Test = c(rmse_test_ridge, rmse_test_lasso)
)
comparison  # Display RMSE comparison between Ridge and LASSO

# ------------------------------------------------------------
# ------------------------------------------------------------

# Stepwise Selection
# Step 13: Stepwise Regression
stepwise_model <- step(lm(Grad.Rate ~ ., data = trainData), direction = "both")  # Stepwise selection using both directions
summary(stepwise_model)  # Summary of the stepwise model

# ------------------------------------------------------------
# ------------------------------------------------------------

# Evaluate Stepwise Model
pred_train_step <- predict(stepwise_model, newdata = trainData)  # Predictions on training data
rmse_train_step <- sqrt(mean((trainData$Grad.Rate - pred_train_step)^2))  # Calculate RMSE for training set

pred_test_step <- predict(stepwise_model, newdata = testData)  # Predictions on test data
rmse_test_step <- sqrt(mean((testData$Grad.Rate - pred_test_step)^2))  # Calculate RMSE for test set

# ------------------------------------------------------------
# ------------------------------------------------------------

# Stepwise Model Comparison
stepwise_comparison <- data.frame(
  Model = "Stepwise",
  RMSE_Train = rmse_train_step,
  RMSE_Test = rmse_test_step
)
stepwise_comparison  # Display RMSE for Stepwise model
```