Exercise 4

for Advanced Methods for Regression and Classification

Dzhamilia Kulikieva

13.11.2024

```
load("/Users/djem/Downloads/building.RData")
```

```
set.seed(2024)
sample_index <- sample(1:nrow(df), size = floor(2/3 * nrow(df)))
train_data <- df[sample_index, ]
test_data <- df[-sample_index, ]
```

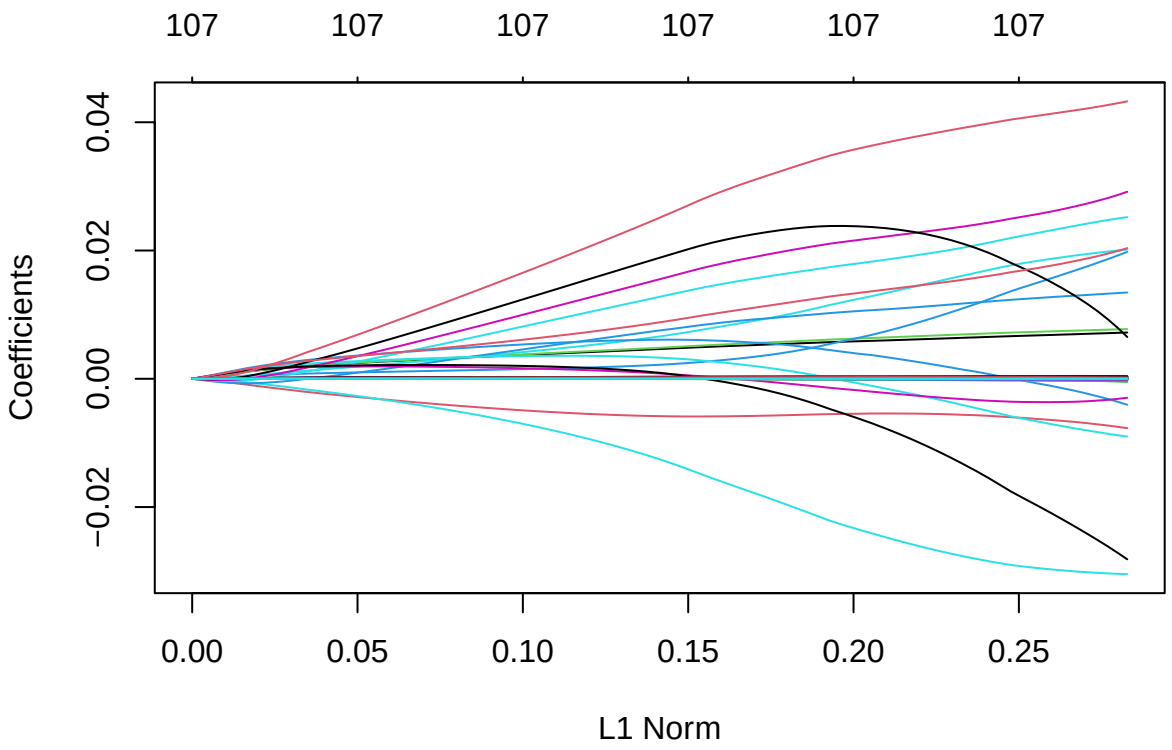
1. Ridge Regression

1a. Ridge Regression with alpha = 0

```
# Training feature matrix and response vector
X_train <- as.matrix(train_data[, -1])  # Features (all columns except the first)
y_train <- train_data[, 1]  # Response (first column)

# Build the Ridge regression model with alpha = 0
ridge_model <- glmnet(X_train, y_train, alpha = 0)

# Plot the model
plot(ridge_model)
```



The plot shows the relationship between the coefficients and the values of lambda. The X-axis represents the log(lambda) values, and the Y-axis displays the coefficient values. We see how the coefficients shrink as lambda increases.

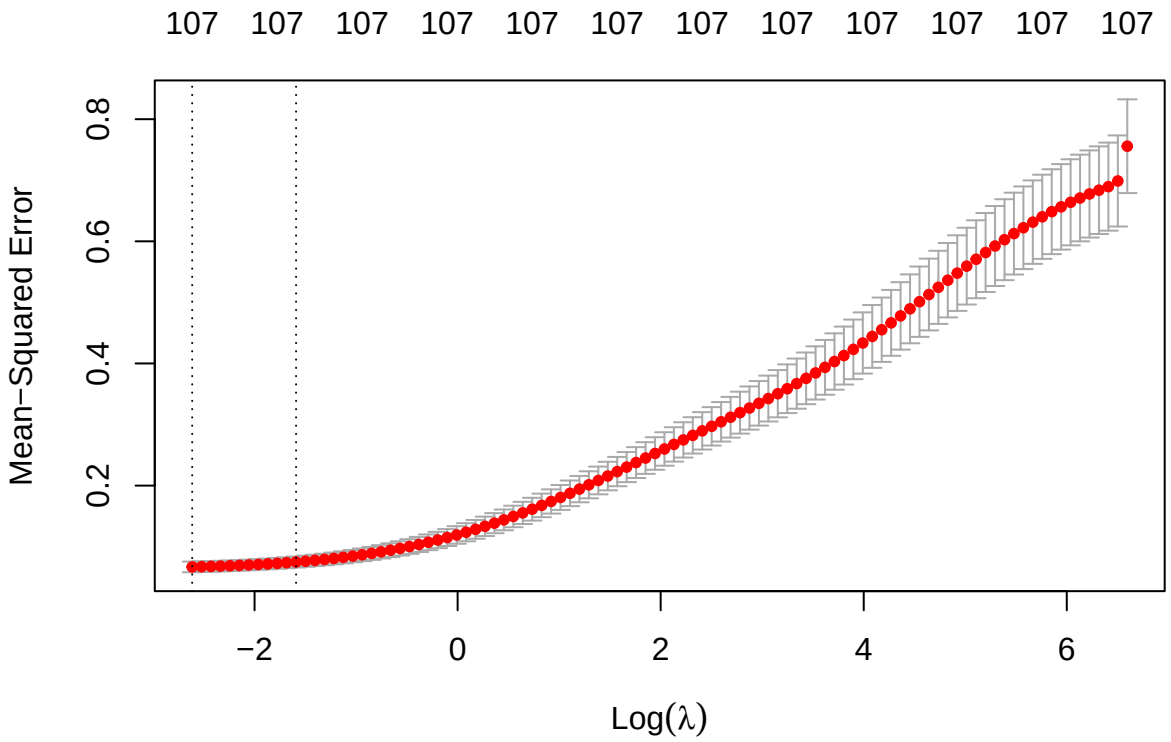
Parameter lambda is the regularization parameter that controls the degree of shrinkage for the coefficients. The higher the lambda, the stronger the regularization, leading to smaller coefficient values. By default, the gmnet function does't specify a specific value for lambda, and instead a range of values of lambda is used.

Parameter alpha controls the type of regression. When alpha = 0, it uses Ridge regression (L2 regularization). This means we limit the magnitude of the coefficients but do not force them to become zero, as in Lasso.

1b. Cross-Validation with cv.glmnet()

```
# Use cv.glmnet() to perform cross-validation on the training set
cv_ridge_model <- cv.glmnet(X_train, y_train, alpha = 0)

# Plot the results of the cross-validation
plot(cv_ridge_model)
```



cv.glmnet() performs k-fold cross-validation to find the optimal value of lambda. The plot displays the mean cross-validation error for each value of lambda. Dashed vertical lines indicate the optimal lambda values: one for the minimum error (lambda.min) and one for the one-standard-error rule (lambda.1se).The standard approach is to select the lambda that yields the minimum MSE (lambda.min represented by first dashed line) ~ -2,70. However, this can sometimes lead to overfitting because the model is fine-tuned to the training data with minimal error. To avoid overfitting and still get a good fit, the second dashed line (lambda.1se) might be the better choice ~ -1,80. It provides a balance between bias and variance by regularizing the model slightly more than lambda.min.

```
# Get the optimal lambda value
optimal_lambda <- cv_ridge_model$lambda.min
cat("Optimal lambda: ", optimal_lambda, "\n")
```

```
## Optimal lambda:  0.07321013
```

```
# Get the coefficients of the model for the optimal lambda
ridge_coefficients <- coef(cv_ridge_model, s = "lambda.min")
print(ridge_coefficients)
```

```
## 108 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)         2.075413e+00
## START.YEAR          7.216330e-03
## START.QUARTER      -7.699095e-03
## COMPLETION.YEAR     7.750186e-03
## COMPLETION.QUARTER  1.979104e-02
## PhysFin1           -3.047520e-02
## PhysFin2            1.995949e-05
## PhysFin3           -8.149266e-05
## PhysFin4           -1.530385e-05
## PhysFin5           -5.237417e-04
## PhysFin6            2.390426e-04
```

```
## PhysFin7            2.014544e-02
## PhysFin8            3.328881e-04
## Econ1               2.054152e-05
## Econ2               9.988636e-05
## Econ3               3.612409e-04
## Econ4              -4.046502e-03
## Econ5               7.820681e-09
## Econ6               2.334352e-06
## Econ7              -7.680694e-05
## Econ8               1.717902e-04
## Econ9               3.829768e-07
## Econ10              1.345601e-02
## Econ11              1.895444e-06
## Econ12              3.435932e-08
## Econ13             -2.221930e-06
## Econ14              1.901774e-05
## Econ15              3.165517e-04
## Econ16              1.426799e-04
## Econ17              5.990567e-06
## Econ18              1.394065e-06
## Econ19              2.156524e-08
## Econ1.lag1          1.208885e-05
## Econ2.lag1          2.414123e-05
## Econ3.lag1         -1.673694e-06
## Econ4.lag1         -9.008647e-03
## Econ5.lag1          1.277996e-08
## Econ6.lag1         -6.123924e-06
## Econ7.lag1         -1.562088e-04
## Econ8.lag1          4.531828e-04
## Econ9.lag1          5.670355e-07
## Econ10.lag1         2.520077e-02
## Econ11.lag1        -1.091185e-05
## Econ12.lag1        -2.590893e-06
## Econ13.lag1         1.537920e-06
## Econ14.lag1         1.553668e-05
## Econ15.lag1         2.088735e-04
## Econ16.lag1         1.135879e-04
## Econ17.lag1         5.269623e-06
## Econ18.lag1        -8.765402e-07
## Econ19.lag1         9.373343e-09
## Econ1.lag2         -4.566438e-06
## Econ2.lag2          5.633831e-05
## Econ3.lag2         -9.669879e-05
## Econ4.lag2         -2.960351e-03
## Econ5.lag2          1.269500e-08
## Econ6.lag2         -8.594717e-06
## Econ7.lag2         -3.031165e-04
## Econ8.lag2          1.767350e-04
## Econ9.lag2          2.668900e-06
## Econ10.lag2         2.916108e-02
## Econ11.lag2         1.539916e-05
## Econ12.lag2         1.425151e-06
## Econ13.lag2        -4.773676e-06
## Econ14.lag2         2.198237e-05
```

```
## Econ15.lag2         2.301603e-04
## Econ16.lag2         2.000893e-04
## Econ17.lag2         2.391582e-06
## Econ18.lag2         1.117879e-06
## Econ19.lag2         4.361459e-10
## Econ1.lag3          2.711651e-05
## Econ2.lag3          1.161711e-04
## Econ3.lag3         -1.156885e-04
## Econ4.lag3         -2.814680e-02
## Econ5.lag3          5.970402e-09
## Econ6.lag3         -5.716593e-06
## Econ7.lag3         -3.580537e-04
## Econ8.lag3          3.158032e-04
## Econ9.lag3          1.413245e-06
## Econ10.lag3         6.516476e-03
## Econ11.lag3         1.644174e-05
## Econ12.lag3         1.776264e-05
## Econ13.lag3        -1.263195e-06
## Econ14.lag3         1.888547e-05
## Econ15.lag3         3.423841e-04
## Econ16.lag3         2.525526e-04
## Econ17.lag3         2.464910e-06
## Econ18.lag3         4.301482e-07
## Econ19.lag3         4.028548e-09
## Econ1.lag4          3.086248e-05
## Econ2.lag4          2.300935e-04
## Econ3.lag4          2.822564e-04
## Econ4.lag4          2.037569e-02
## Econ5.lag4         -1.029853e-09
## Econ6.lag4          7.391936e-06
## Econ7.lag4          7.139856e-05
## Econ8.lag4         -1.365011e-04
## Econ9.lag4         -9.705639e-07
## Econ10.lag4         4.326047e-02
## Econ11.lag4         7.925080e-06
## Econ12.lag4         2.611849e-05
## Econ13.lag4        -3.335861e-07
## Econ14.lag4         9.365211e-06
## Econ15.lag4         3.929262e-04
## Econ16.lag4         1.986826e-04
## Econ17.lag4         6.571761e-06
## Econ18.lag4         1.429256e-06
## Econ19.lag4         2.785812e-09
```

The function gives us the precise lambda value that minimizes the MSE, which appears slightly different from the visual estimate on the plot.

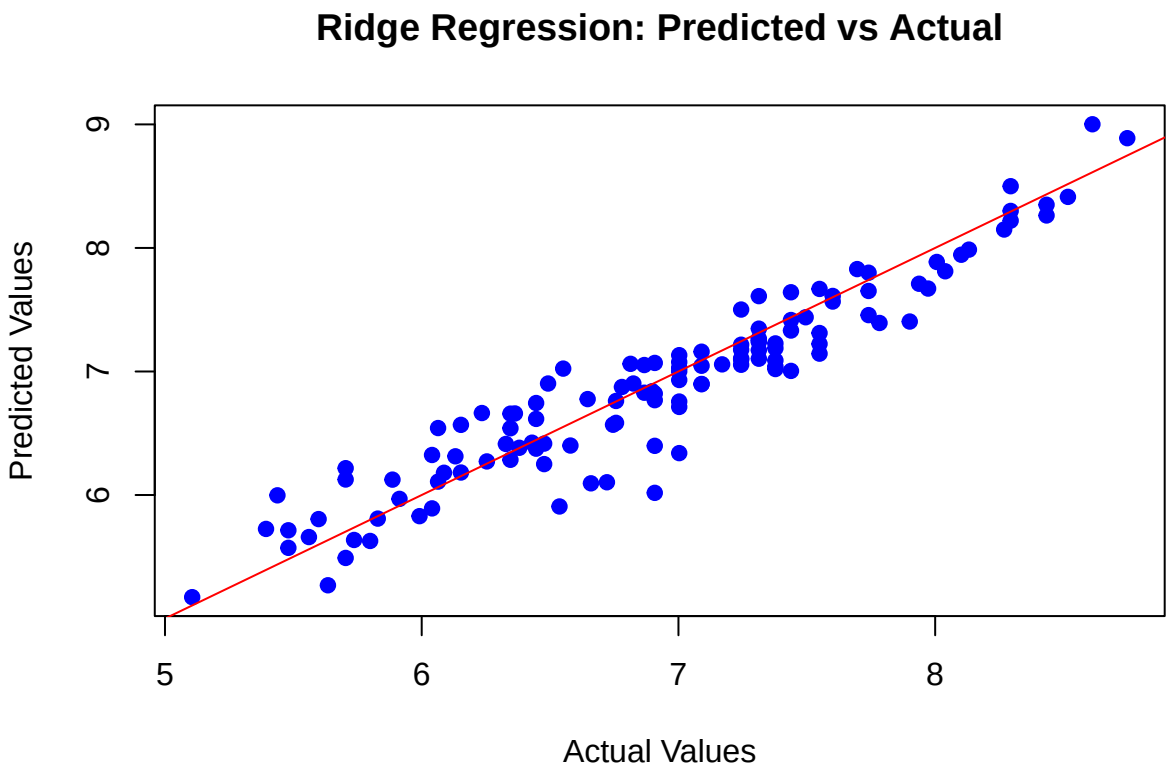
1c.Predictions and Calculation of RMSE on the test_data

```
# Prepare the test data
X_test <- as.matrix(test_data[, -1])  # Features (all columns except the first)
y_test <- test_data[, 1]  # Response (first column)

# Make predictions using the optimal Ridge model
```

```r
predictions_ridge <- predict(cv_ridge_model, newx = X_test, s = "lambda.min")

# Plot the predicted vs actual values
plot(y_test, predictions_ridge, main = "Ridge Regression: Predicted vs Actual",
     xlab = "Actual Values", ylab = "Predicted Values", pch = 19, col = "blue")
abline(a = 0, b = 1, col = "red")  # Add a 45-degree line for comparison
```




```r
# Calculate RMSE (Root Mean Squared Error)
rmse_ridge <- sqrt(mean((predictions_ridge - y_test)^2))
cat("RMSE for Ridge Regression: ", rmse_ridge, "\n")
```

```
## RMSE for Ridge Regression:  0.2586422
```

The points generally follow a diagonal trend along the red line, which represents a good prediction line where predicted = observed.

The calculated RMSE of 0.2586 is relatively low, indicating that the average prediction error is small, and the model performs well on the test data. There is some scattering around the line, but most points are clustered closely, which suggests that the model captures the underlying pattern of the data effectively, with only minor deviations.

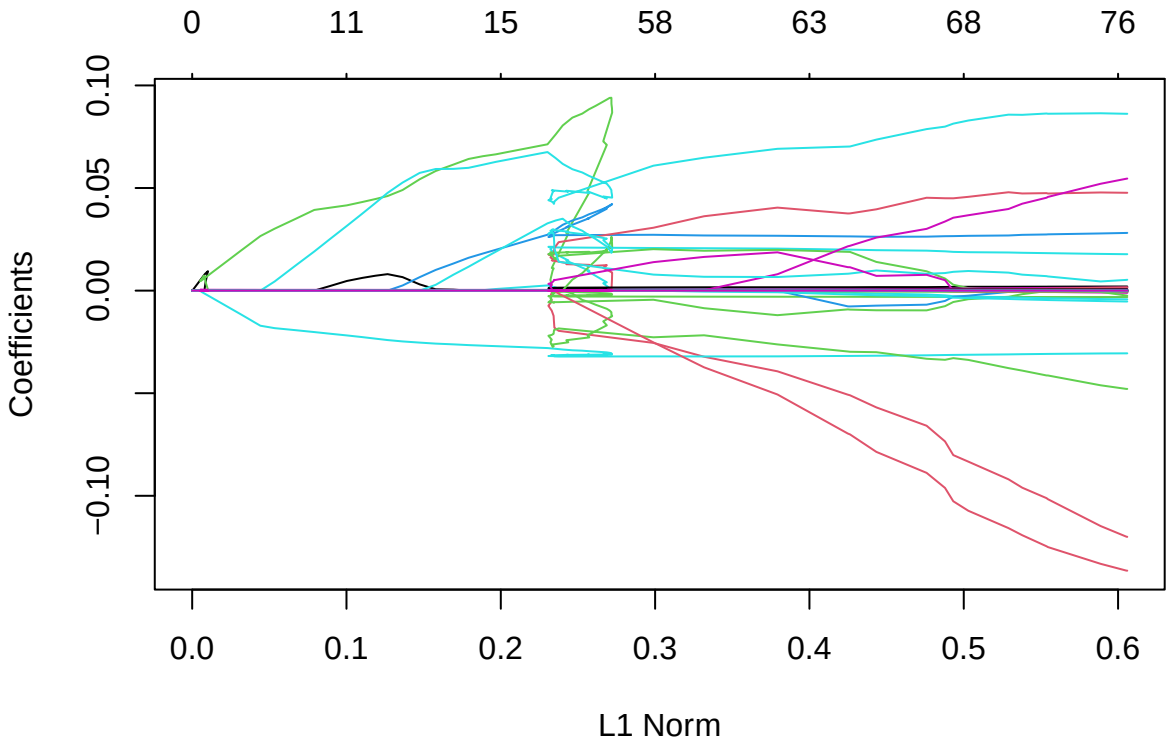
The RMSE results for the PLS and PCR prediction models from the previous exercise (0.2519375) are almost the same as the RMSE for Ridge Regression (0.2586422).This similarity in RMSE values suggests that all three models—PLS, PCR, and Ridge Regression—are performing comparably in terms of predictive accuracy on this dataset.

2. Lasso Regression:

2a. Lasso Regression with alpha = 1

```
# Build the Lasso model with alpha = 1 (Lasso regression)
lasso_model <- glmnet(X_train, y_train, alpha = 1)

# Plot the model coefficients against log(lambda)
plot(lasso_model)
```

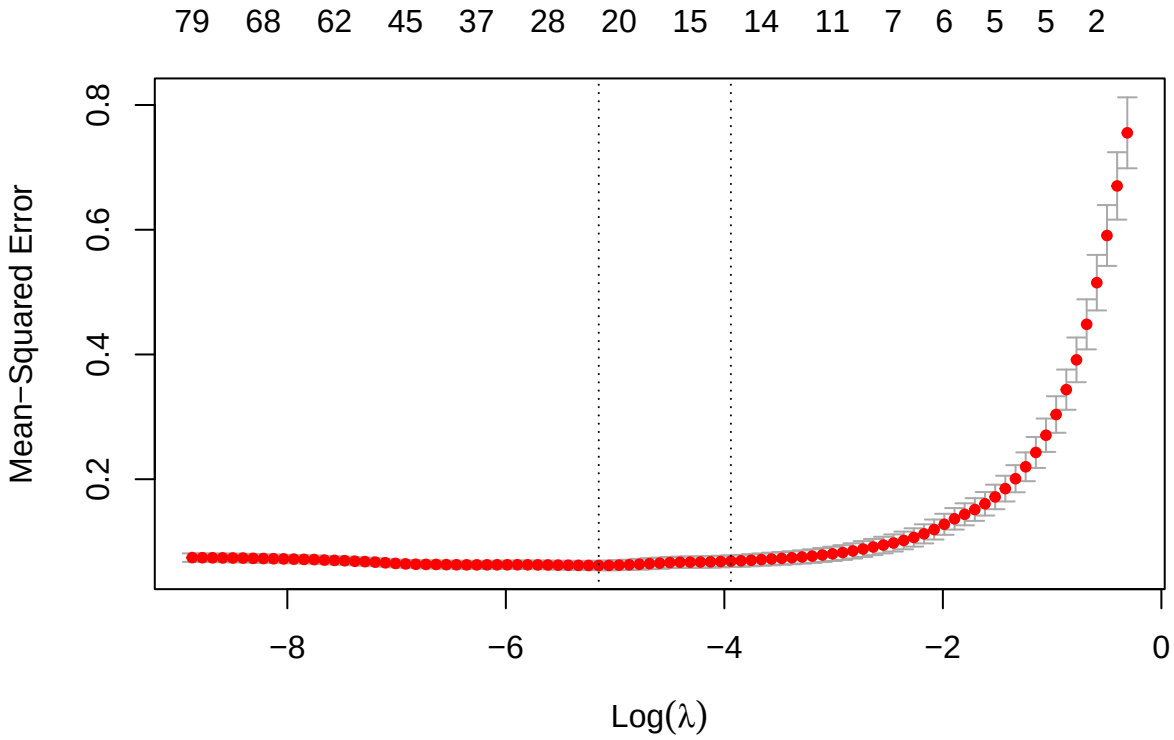


At lower values of lambda (left side), there is minimal regularization, and most coefficients are non-zero, meaning the model includes almost all features. As lambda increases (moving right), many coefficients shrink toward zero and eventually become exactly zero, effectively removing those features from the model.

2b. Cross-Validation with cv.glmnet() for the Lasso model

```
# Perform cross-validation to select the optimal lambda for Lasso
cv_lasso_model <- cv.glmnet(X_train, y_train, alpha = 1)

# Plot the results of the cross-validation
plot(cv_lasso_model)
```



The value of lambda with the minimum cross-validation error is approximately -5.3. This lambda is likely to result in more features with non-zero coefficients and may provide the most accurate predictions. The largest lambda within one standard error of the minimum error is approximately -3.99. This value will apply stronger regularization, potentially leading to fewer non-zero coefficients and offering a simpler model that balances accuracy and interpretability.

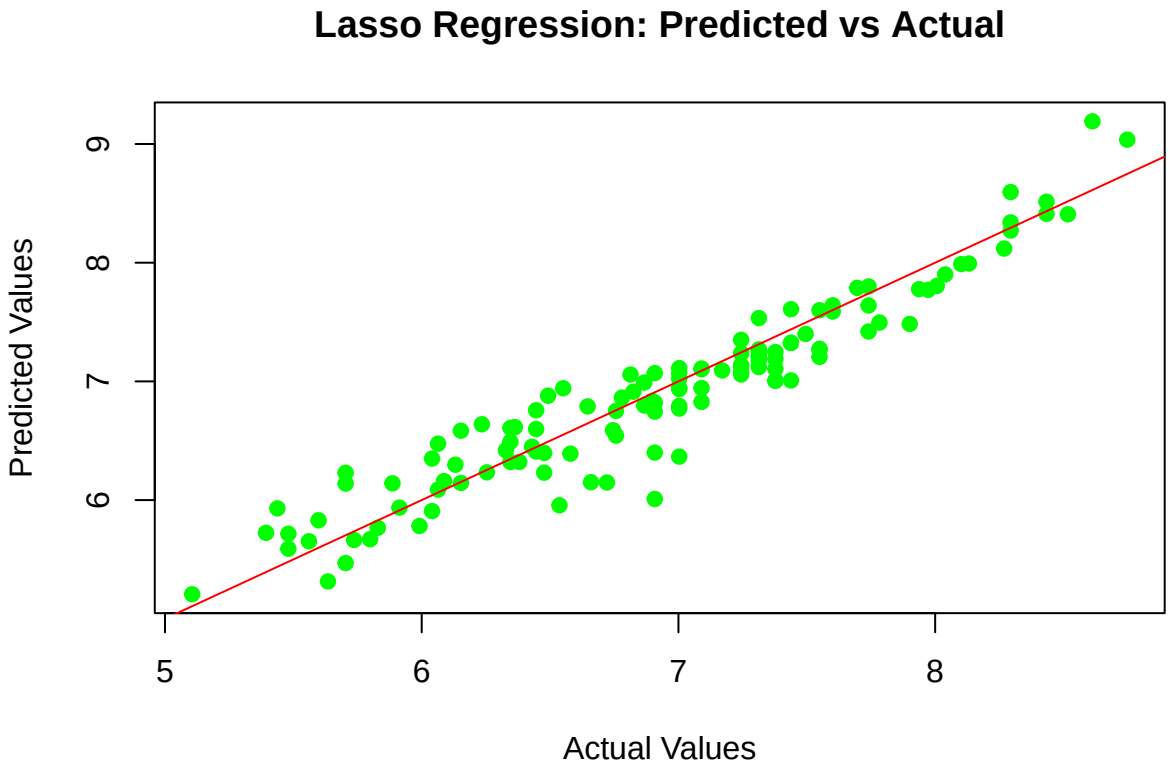
2c. Predictions and Calculation of RMSE for the Lasso model

```
# Get the optimal lambda value for Lasso regression
optimal_lambda_lasso <- cv_lasso_model$lambda.min
cat("Optimal lambda for Lasso: ", optimal_lambda_lasso, "\n")
```

```
## Optimal lambda for Lasso:  0.005801778
```

```
# Make predictions on the test data using the optimal Lasso model
predictions_lasso <- predict(cv_lasso_model, newx = X_test, s = "lambda.min")

# Plot the predicted vs actual values for Lasso regression
plot(y_test, predictions_lasso, main = "Lasso Regression: Predicted vs Actual",
     xlab = "Actual Values", ylab = "Predicted Values", pch = 19, col = "green")
abline(a = 0, b = 1, col = "red")  # Add a 45-degree line for comparison
```




```
# Calculate the RMSE for the Lasso model
rmse_lasso <- sqrt(mean((predictions_lasso - y_test)^2))
cat("RMSE for Lasso Regression: ", rmse_lasso, "\n")
```

```
## RMSE for Lasso Regression:  0.2495514
```

In Lasso regression, the optimal lambda of 0.0058 applies mild regularization, keeping more features in the model but potentially setting some coefficients to zero for feature selection. In Ridge regression, the optimal lambda of 0.0732 applies stronger regularization, shrinking all coefficients but retaining them, thus controlling multicollinearity without eliminating features.

The plot for the Lasso model is very similar to the plot for the Ridge regression model, showing the relationship between the predicted and actual values.

In summary, both models(Ridge regression and Lasso) perform similarly in terms of RMSE, with Lasso slightly outperforming Ridge.

3. Adaptive Lasso Regression:

3a. Constructing Weights Based on Ridge Regression

We will create weights by taking the inverse of the absolute values of the Ridge model coefficients (excluding zeros to avoid division by zero).

```
# Extract Ridge model coefficients, excluding the intercept
ridge_coefficients <- coef(cv_ridge_model, s = "lambda.min")[-1, ]  # Remove the intercept

# Create weights by inverting the absolute values of the coefficients
adaptive_weights <- 1 / abs(ridge_coefficients)
adaptive_weights[is.infinite(adaptive_weights)] <- 0  # Set infinite values to zero
```

```r
# Check the length to ensure it matches the number of variables in X_train
print(length(adaptive_weights))
```

```
## [1] 107
```

```r
print(ncol(X_train))
```

```
## [1] 107
```

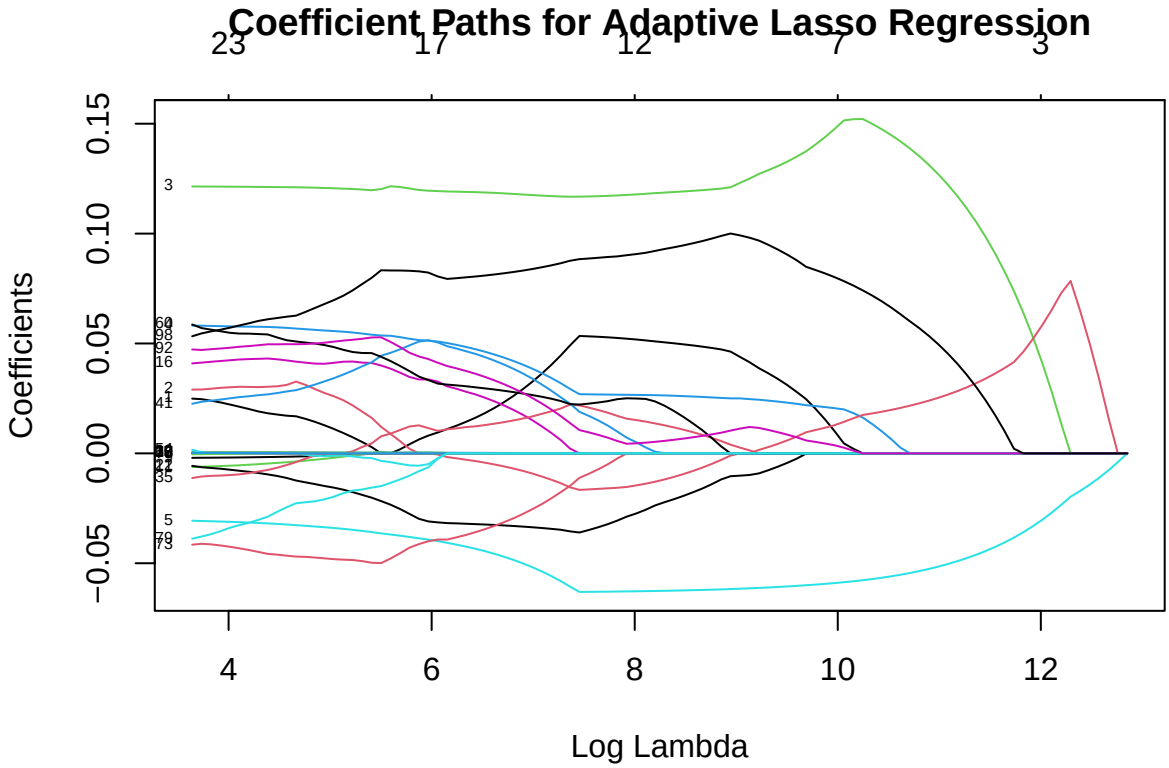
Since both values are equal to 107, this means that the number of features in the data (X_train) corresponds to the number of adaptive weights, which confirms that the adaptive weights were correctly calculated for all features.

3b. Building the Adaptive Lasso Model with penalty.factor

We will use the weights in the penalty.factor parameter for the Adaptive Lasso model, setting alpha = 1.

```r
# Build Adaptive Lasso model using the corrected weights
adaptive_lasso_model <- glmnet(X_train, y_train, alpha = 1, penalty.factor = adaptive_weights)

# Plot the Adaptive Lasso model
plot(adaptive_lasso_model, xvar = "lambda", label = TRUE)
title("Coefficient Paths for Adaptive Lasso Regression")
```



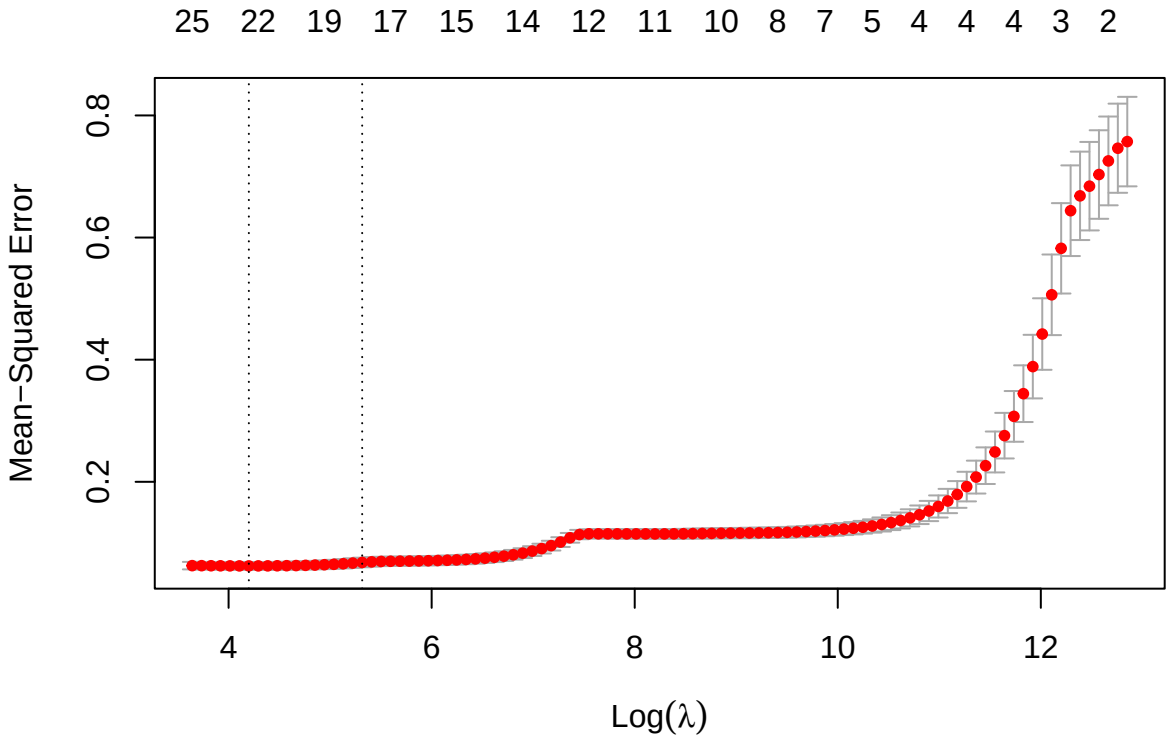
We see that with increasing lambda, the coefficients tend to zero, and fewer features remain in the model with non-zero coefficients. Adaptive Lasso adjusts different features differently, based on their previous coefficients (compared to the usual Lasso), which allows us to focus on more important features.

3c. Cross-Validation for Adaptive Lasso

Let's run cross-validation to select the optimal lambda parameter for Adaptive Lasso

```r
# Cross-validation for Adaptive Lasso
cv_adaptive_lasso_model <- cv.glmnet(X_train, y_train, alpha = 1, penalty.factor = adaptive_weights)

# Visualize cross-validation results
plot(cv_adaptive_lasso_model)
```




```r
# Get the optimal lambda value
optimal_lambda_adaptive <- cv_adaptive_lasso_model$lambda.min
cat("Optimal lambda for Adaptive Lasso: ", optimal_lambda_adaptive, "\n")
```

```
## Optimal lambda for Adaptive Lasso:  66.65166
```

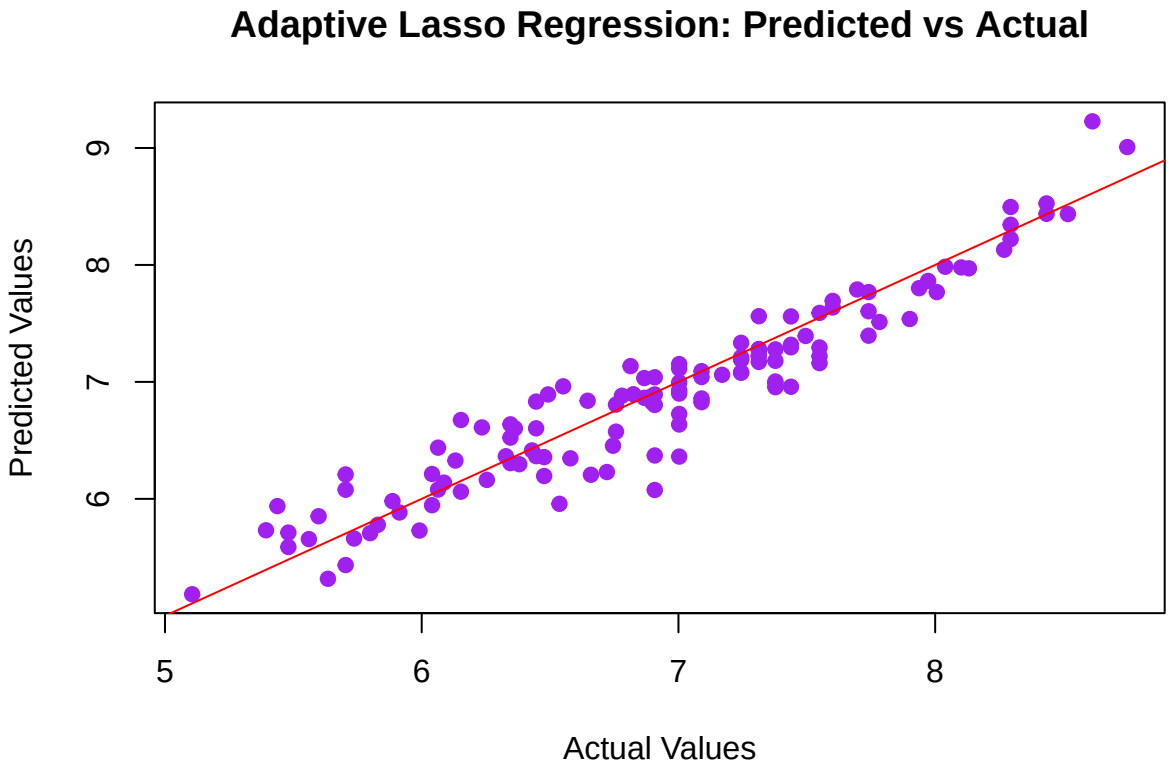
A lambda of 66.65166 indicates that the model is applying a relatively strong regularization compared to the standard Lasso or Ridge Regression models, which are often smaller and more directly chosen to reduce overfitting. A larger lambda in Adaptive Lasso lead to a simpler model with fewer non-zero coefficients, especially for features that are less significant according to the Ridge model.

3d. Prediction and RMSE Calculation for the Adaptive Lasso model

We will use the optimal Adaptive Lasso model to make predictions on the test data and compare the RMSE with previous models.

```r
# Predictions on the test data using Adaptive Lasso
predictions_adaptive_lasso <- predict(cv_adaptive_lasso_model, newx = X_test, s = "lambda.min")

# Plot predicted vs actual values
plot(y_test, predictions_adaptive_lasso, main = "Adaptive Lasso Regression: Predicted vs Actual",
     xlab = "Actual Values", ylab = "Predicted Values", pch = 19, col = "purple")
abline(a = 0, b = 1, col = "red")  # Add y=x line for comparison
```




```
# Calculate RMSE for Adaptive Lasso
rmse_adaptive_lasso <- sqrt(mean((predictions_adaptive_lasso - y_test)^2))
cat("RMSE for Adaptive Lasso Regression: ", rmse_adaptive_lasso, "\n")
```

```
## RMSE for Adaptive Lasso Regression:  0.2525235
```

Lasso Regression(RMSE=0.2495514) appears to perform slightly better than Adaptive Lasso Regression based on the RMSE. This suggests that, for this dataset, the simpler regularization provided by standard Lasso might be more effective than the more complex adaptive regularization applied in the Adaptive Lasso.

The plot for the Adaptive Lasso model closely mirrors that of the Lasso model, it suggests that both models have very similar predictive performance on the test data.

3e. Comparing Coefficients with Lasso

Retrieve the Adaptive Lasso coefficients for the optimal lambda value and compare them with the regular Lasso coefficients.

```
# Extract Lasso model coefficients (excluding the intercept)
lasso_coefficients <- coef(cv_lasso_model, s = "lambda.min")

# Adaptive Lasso Coefficients
adaptive_lasso_coefficients <- coef(cv_adaptive_lasso_model, s = "lambda.min")

# Comparing Lasso and Adaptive Lasso coefficients
print("Coefficients for Lasso Regression:")
```

```
## [1] "Coefficients for Lasso Regression:"
```

```
print(lasso_coefficients)
```

```
## 108 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
```

```
## (Intercept)        -3.067356e+00
## START.YEAR          .
## START.QUARTER       .
## COMPLETION.YEAR     9.310277e-02
## COMPLETION.QUARTER  4.052517e-02
## PhysFin1           -3.011022e-02
## PhysFin2            .
## PhysFin3           -2.202088e-05
## PhysFin4            .
## PhysFin5           -1.304312e-03
## PhysFin6            3.150409e-04
## PhysFin7            .
## PhysFin8            4.119621e-04
## Econ1               7.643283e-06
## Econ2               .
## Econ3               .
## Econ4               .
## Econ5               .
## Econ6               .
## Econ7               .
## Econ8               7.896147e-05
## Econ9               .
## Econ10              .
## Econ11              .
## Econ12              .
## Econ13              .
## Econ14              .
## Econ15              .
## Econ16              .
## Econ17              .
## Econ18              .
## Econ19              .
## Econ1.lag1          .
## Econ2.lag1          .
## Econ3.lag1          .
## Econ4.lag1          .
## Econ5.lag1          .
## Econ6.lag1          .
## Econ7.lag1          .
## Econ8.lag1          1.406401e-04
## Econ9.lag1          .
## Econ10.lag1         2.006876e-02
## Econ11.lag1         .
## Econ12.lag1         .
## Econ13.lag1         .
## Econ14.lag1         .
## Econ15.lag1         .
## Econ16.lag1         .
## Econ17.lag1         .
## Econ18.lag1         .
## Econ19.lag1         .
## Econ1.lag2          .
## Econ2.lag2          .
## Econ3.lag2          .
```

```
## Econ4.lag2          .
## Econ5.lag2          .
## Econ6.lag2          .
## Econ7.lag2          .
## Econ8.lag2          .
## Econ9.lag2          3.598804e-06
## Econ10.lag2         2.151128e-02
## Econ11.lag2         .
## Econ12.lag2         .
## Econ13.lag2         .
## Econ14.lag2         2.192083e-05
## Econ15.lag2         .
## Econ16.lag2         .
## Econ17.lag2         .
## Econ18.lag2         .
## Econ19.lag2         .
## Econ1.lag3          .
## Econ2.lag3          .
## Econ3.lag3          .
## Econ4.lag3         -8.859362e-03
## Econ5.lag3          .
## Econ6.lag3          .
## Econ7.lag3          .
## Econ8.lag3          8.563931e-06
## Econ9.lag3          8.120116e-07
## Econ10.lag3         .
## Econ11.lag3         .
## Econ12.lag3         .
## Econ13.lag3         .
## Econ14.lag3         4.227666e-05
## Econ15.lag3         .
## Econ16.lag3         .
## Econ17.lag3         .
## Econ18.lag3         .
## Econ19.lag3         .
## Econ1.lag4          2.024201e-05
## Econ2.lag4          .
## Econ3.lag4          .
## Econ4.lag4          .
## Econ5.lag4          .
## Econ6.lag4          .
## Econ7.lag4          .
## Econ8.lag4          .
## Econ9.lag4          .
## Econ10.lag4         5.210548e-02
## Econ11.lag4         .
## Econ12.lag4         .
## Econ13.lag4         .
## Econ14.lag4         .
## Econ15.lag4         .
## Econ16.lag4         .
## Econ17.lag4         3.668237e-08
## Econ18.lag4         .
## Econ19.lag4         .
```

```r
print("Coefficients for Adaptive Lasso Regression:")
```

```
## [1] "Coefficients for Adaptive Lasso Regression:"
```

```r
print(adaptive_lasso_coefficients)
```

```
## 108 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)        -6.734091e+00
## START.YEAR          2.019038e-02
## START.QUARTER       3.024010e-02
## COMPLETION.YEAR     1.212853e-01
## COMPLETION.QUARTER  5.765182e-02
## PhysFin1           -3.147580e-02
## PhysFin2            .
## PhysFin3            .
## PhysFin4            .
## PhysFin5           -1.798053e-03
## PhysFin6            3.718094e-04
## PhysFin7           -5.176000e-03
## PhysFin8            4.242208e-04
## Econ1               .
## Econ2               .
## Econ3               .
## Econ4               4.290152e-02
## Econ5               .
## Econ6               .
## Econ7               .
## Econ8               .
## Econ9               .
## Econ10             -8.372126e-03
## Econ11              .
## Econ12              .
## Econ13              .
## Econ14              .
## Econ15              .
## Econ16              .
## Econ17              .
## Econ18              .
## Econ19              .
## Econ1.lag1          .
## Econ2.lag1          .
## Econ3.lag1          .
## Econ4.lag1         -8.994401e-03
## Econ5.lag1          .
## Econ6.lag1          .
## Econ7.lag1          .
## Econ8.lag1          4.337987e-04
## Econ9.lag1          .
## Econ10.lag1         2.599014e-02
## Econ11.lag1         .
## Econ12.lag1         .
## Econ13.lag1         .
## Econ14.lag1         .
```

```
## Econ15.lag1          .
## Econ16.lag1          .
## Econ17.lag1          .
## Econ18.lag1          .
## Econ19.lag1          .
## Econ1.lag2           .
## Econ2.lag2           .
## Econ3.lag2           .
## Econ4.lag2           .
## Econ5.lag2           .
## Econ6.lag2           .
## Econ7.lag2           .
## Econ8.lag2           1.885565e-05
## Econ9.lag2           .
## Econ10.lag2          5.447822e-02
## Econ11.lag2          .
## Econ12.lag2          .
## Econ13.lag2          .
## Econ14.lag2          .
## Econ15.lag2          .
## Econ16.lag2          .
## Econ17.lag2          .
## Econ18.lag2          .
## Econ19.lag2          .
## Econ1.lag3           .
## Econ2.lag3           .
## Econ3.lag3           .
## Econ4.lag3          -4.396487e-02
## Econ5.lag3           .
## Econ6.lag3           .
## Econ7.lag3          -1.000801e-04
## Econ8.lag3           1.414004e-04
## Econ9.lag3           .
## Econ10.lag3         -3.157317e-02
## Econ11.lag3          .
## Econ12.lag3          .
## Econ13.lag3          .
## Econ14.lag3          .
## Econ15.lag3          .
## Econ16.lag3          .
## Econ17.lag3          .
## Econ18.lag3          .
## Econ19.lag3          .
## Econ1.lag4           .
## Econ2.lag4           .
## Econ3.lag4           .
## Econ4.lag4           4.881735e-02
## Econ5.lag4           .
## Econ6.lag4           .
## Econ7.lag4           .
## Econ8.lag4           .
## Econ9.lag4           .
## Econ10.lag4          5.918904e-02
## Econ11.lag4          .
```

```
## Econ12.lag4         .
## Econ13.lag4         .
## Econ14.lag4         .
## Econ15.lag4         .
## Econ16.lag4         .
## Econ17.lag4         .
## Econ18.lag4         .
## Econ19.lag4         .
```

Lasso Regression Coefficients: The Lasso model has coefficients for some variables that are exactly zero, indicating feature selection. Variables like START.YEAR, PhysFin2, and many Econ variables (e.g., Econ1, Econ3, Econ5, etc.) have zero coefficients, suggesting that Lasso has removed them from the model. For the variables that are not zero, Lasso has shrunk their coefficients significantly. For example, COMPLETION.YEAR has a non-zero coefficient of 0.0931, indicating its contribution to the model.

Adaptive Lasso Regression Coefficients: Adaptive Lasso, influenced by Ridge regression coefficients as penalty factors, shows a different pattern. Some of the coefficients that were zero in Lasso (e.g., PhysFin2, Econ5, Econ9, Econ2.lag1) still have small non-zero values in Adaptive Lasso, indicating that Adaptive Lasso assigns different penalties to variables based on their importance, using Ridge weights. Variables such as START.YEAR, COMPLETION.YEAR, and PhysFin1 still have non-zero coefficients, with some changes in magnitude compared to Lasso.