Report

Regularization

ALY6015 Intermediate Analytics

Module 4 Assignment

Author: Jainam Patel

Faculty: Prof. Richard He

Date: 05/02/2025

Introduction:

The dataset used here is from the package ISLR having data frame College for which has data collected from 1995 issue of US News and World Report.

The data frame consists of 777 observations and 18 features having following information:

- Private - A factor with levels No and Yes indicating private or public university
- Apps - Number of applications received
- Accept - Number of applications accepted
- Enroll - Number of new students enrolled
- Top10perc - Pct. new students from top 10% of H.S. class
- Top25perc - Pct. new students from top 25% of H.S. class
- F.Undergrad - Number of fulltime undergraduates
- P.Undergrad - Number of parttime undergraduates
- Outstate - Out-of-state tuition
- Room.Board - Room and board costs
- Books - Estimated book costs
- Personal - Estimated personal spending
- PhD - Pct. of faculty with Ph.D.'s
- Terminal - Pct. of faculty with terminal degree
- S.F.Ratio - Student/faculty ratio
- perc.alumni - Pct. alumni who donate
- Expend - Instructional expenditure per student
- Grad.Rate - Graduation rate

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Private*	1	777	1.73	0.45	2.0	1.78	0.00	1.0	2.0	1.0	-1.02	-0.96	0.02
Apps	2	777	3001.64	3870.20	1558.0	2193.01	1463.33	81.0	48094.0	48013.0	3.71	26.52	138.84
Accept	3	777	2018.80	2451.11	1110.0	1510.29	1008.17	72.0	26330.0	26258.0	3.40	18.75	87.93
Enroll	4	777	779.97	929.18	434.0	575.95	354.34	35.0	6392.0	6357.0	2.68	8.74	33.33
Top10perc	5	777	27.56	17.64	23.0	25.13	13.34	1.0	96.0	95.0	1.41	2.17	0.63
Top25perc	6	777	55.80	19.80	54.0	55.12	20.76	9.0	100.0	91.0	0.26	-0.57	0.71
F.Undergrad	7	777	3699.91	4850.42	1707.0	2574.88	1441.09	139.0	31643.0	31504.0	2.60	7.61	174.01
P.Undergrad	8	777	855.30	1522.43	353.0	536.36	449.23	1.0	21836.0	21835.0	5.67	54.52	54.62
Outstate	9	777	10440.67	4023.02	9990.0	10181.66	4121.63	2340.0	21700.0	19360.0	0.51	-0.43	144.32
Room.Board	10	777	4357.53	1096.70	4200.0	4301.70	1005.20	1780.0	8124.0	6344.0	0.48	-0.20	39.34
Books	11	777	549.38	165.11	500.0	535.22	148.26	96.0	2340.0	2244.0	3.47	28.06	5.92
Personal	12	777	1340.64	677.07	1200.0	1268.35	593.04	250.0	6800.0	6550.0	1.74	7.04	24.29
PhD	13	777	72.66	16.33	75.0	73.92	17.79	8.0	103.0	95.0	-0.77	0.54	0.59
Terminal	14	777	79.70	14.72	82.0	81.10	14.83	24.0	100.0	76.0	-0.81	0.22	0.53
S.F.Ratio	15	777	14.09	3.96	13.6	13.94	3.41	2.5	39.8	37.3	0.66	2.52	0.14
perc.alumni	16	777	22.74	12.39	21.0	21.86	13.34	0.0	64.0	64.0	0.60	-0.11	0.44
Expend	17	777	9660.17	5221.77	8377.0	8823.70	2730.95	3186.0	56233.0	53047.0	3.45	18.59	187.33
Grad.Rate	18	777	65.46	17.18	65.0	65.60	17.79	10.0	118.0	108.0	-0.11	-0.22	0.62

According to the dataset, there are **565 Private colleges** and **212 Public colleges**.

Above image describes the dataset from number of variables, mean, median, standard deviation, minimum and maximum values etc for each variables.

The goal is to **predict Graduation Rate (Grad.Rate**) and find out best method out of Ridge, Lasso, Elastic Net(alpha = 0.5) and Stepwise selection models.

Analysis:

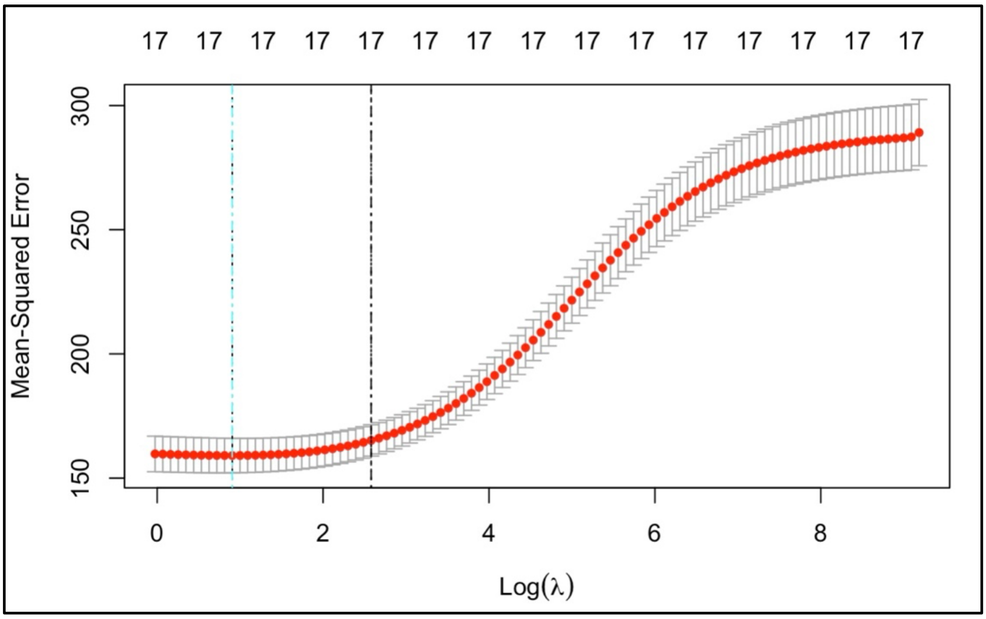
- The **1se lambda** indicates smallest error within standard error of the minimum cross-validation error.
- The **minimum lambda** provides lowest cross-validation error while the regularization process works.

```
## Splitting the dataset into training and testing
trainIndex <- sample(1:nrow(College), size = 0.8 * nrow(College), replace = FALSE)
caret_train <- College[trainIndex, ]
caret_test <- College[-trainIndex, ]

## install.packages("glmnet")
library(glmnet)
train_x <- model.matrix(Grad.Rate ~ ., data = caret_train)[,-1]
train_y <- caret_train$Grad.Rate

test_x <- model.matrix(Grad.Rate ~ ., data = caret_test)[,-1]
test_y <- caret_test$Grad.Rate
```

- The above image displays the splitting of training and testing data where 80% part of data is given for training and 20% is for testing data.
- caret_train and caret_test are actual training and testing datasets.
- **model.matrix()** function creates a matrix in which every row represents college and columns represent predictors/features and subtracting 1 removes intercept column.
- train_x and train_y extracts the target variable which is **Grad.Rate**.
- **glmnet()** function fits **Generalized Linear Models** where alpha value is set to give elastic net regularization.
- While **cv.glmnet()** function applies lambda to cross-validation as lambda controls the strength of the regularization.

Ridge Regularization:

- Ridge regularization, also known as **L2 regularization**, adds a penalty equal to the square of the magnitude of coefficients.
- All the coefficients are shrunk by the same factor but none are eliminated.



- The above image is the plot for Ridge regression cross validation plot.
- The x-axis represents tuning parameter for Ridge regression while y-axis represents mean-squared error.
- Since, mean-squared is low and stable when the lambda values is between 0 and 3, thus indicating good performance with minimal regularization.
- As the values on x-axis increases, mean-squared error increases sharply indicating underfitting as model becomes simple to capture data patterns.
- The red dots represent average mean-squared error for each lambda and grey bars represent variability i.e., shorter bars mean more consistent performance.
- **The blue dashed line is the optimal minimum lambda value which approximately equal to 1 and black line is optimal 1se lambda value which is approximately between 2 and 3.**

Table: Ridge Regression RMSE and Lambda Values

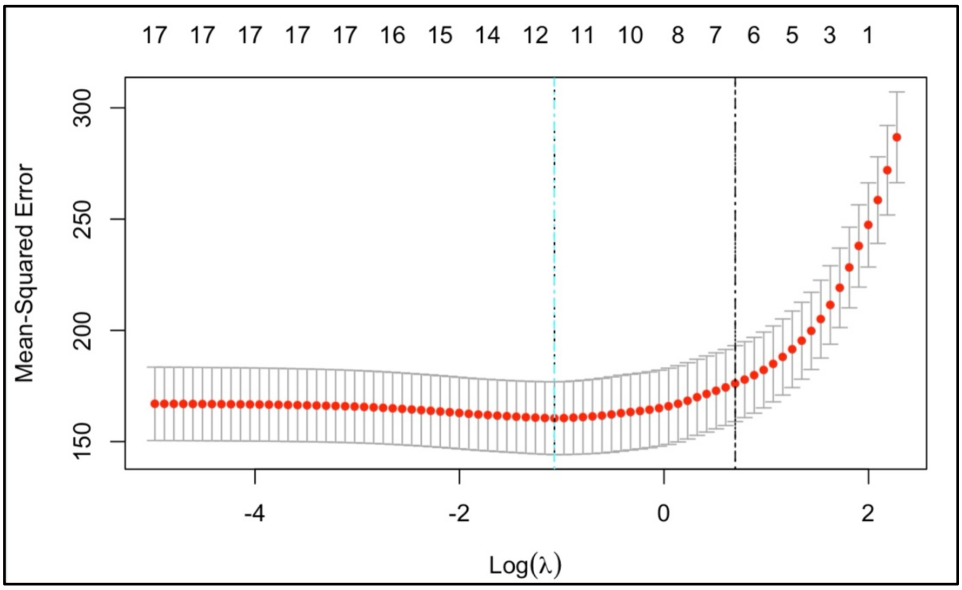
Metric	Value
Minimum Lambda	2.472268
1se Lambda	13.193752
Training RMSE	12.340646
Testing RMSE	14.084288

- The above image is the result for Ridge regression for training and testing data.
- Here, minimum lambda is 2.472 which represents that model reached the point 2.472 as lowest possible error during cross-validation.
- The value 13.193 is the error within one standard error of the minimum cross-validation. It shows that model is simple, i.e., more regularized.

- Since, the testing root mean squared error is a bit higher than training RMSE, we can interpret that **minor overfitting** was observed but the model can be accepted as the difference isn't significant.

LASSO Regularization:

- **L1(LASSO) regularization** adds a penalty equal to the absolute value of the magnitude of coefficients and limits the size of the coefficients.
- LASSO can eliminate the variables/predictors by reducing them to zero.



- For the above image, it is a plot for Lasso regression.
- Error remains relatively stable for lower lambda but beyond a certain point it increases significantly as more coefficients are shrunk.

Table: Lasso Regression RMSE and Lambda Values

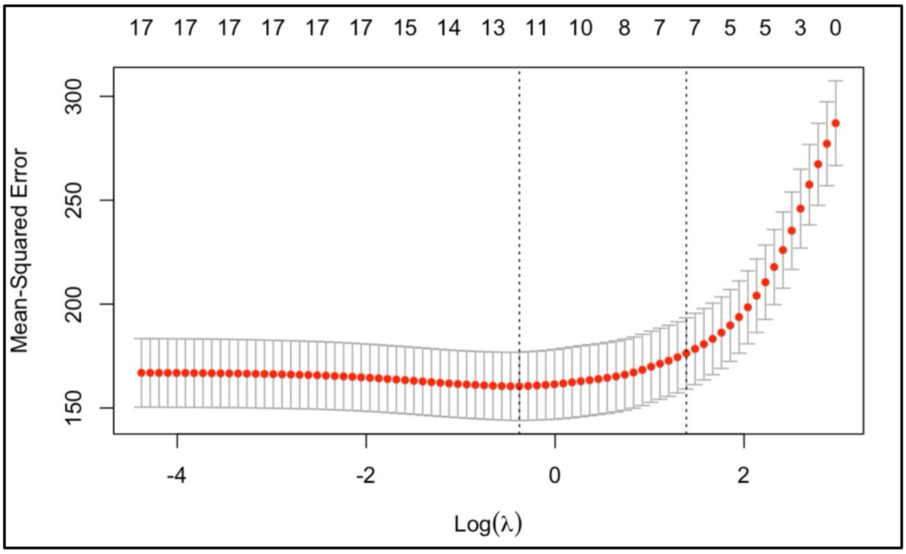
Metric	Value
Minimum Lambda	0.3423811
1se Lambda	2.0053330
Training RMSE	12.3820192
Testing RMSE	14.1081044

- The above image is the result for Lasso regression for training and testing data.
- The model reached the point 0.342 as lowest possible error during cross-validation.
- The 1se lambda value 2.0 is the error within one standard error of the minimum cross-validation. It shows that model is simple, i.e., more regularized and reduces overfitting.

- Since, the testing RMSE is a bit higher than training RMSE, we can interpret that **minor overfitting** was observed but the model can be accepted as the difference isn't significant.

Elastic Net (alpha = 0.5):

- The below image is the plot for alpha value at 0.5 which is the case between Ridge and Lasso regression.



- The plot for Elastic Net provides a compromise as high value is increasing mean-squared error and useful when correlated predictors exist.

Table: Alpha at 0.5 Regression RMSE and Lambda Values

Metric	Value
Minimum Lambda	0.6847621
1se Lambda	4.0106659
Training RMSE	12.3945783
Testing RMSE	14.1069516

- The above image is the result of **ElasticNet** on the same training and testing data.
- Here, the value of **alpha is set to 0.5**.
- The minimum lambda value is 0.68 representing that this was the lowest error during cross-validation.
- The 1se lambda value is 4.01 which is quite higher than minimum lambda thus indicates that more regularization was performed on the model.
- The training and testing RMSE respectively are 12.4 and 14.1 and hence isn't a significant difference and thus **minor overfitting** was observed.

Table: Comparison of Ridge, LASSO and ElasticNet

Model	Training RMSE	Testing RMSE	Minimum Lambda	Lambda 1se
Ridge	12.34065	14.08429	2.4722682	13.193752
LASSO	12.38202	14.10810	0.3423811	2.005333
ElasticNet (α=0.5)	12.39458	14.10695	0.6847621	4.010666

- The above image is the comparison for Ridge, Lasso and ElasticNet models.
- In terms of training and testing RMSE, all models show similar performance but Ridge regression slightly has lesser training and testing RMSE and hence, have a bit better performance.
- Ridge has significantly **higher minimum lambda values** and **lower RMSE** and does not reduce coefficients to zero but rather shrink them and typically deals when predictors are highly correlated, therefore, **I would prefer Ridge regressio**n.
- Here, the ElasticNet is between ridge and lasso but did not perform well significantly while balancing ridge and lasso.

Stepwise Selection:

Table: Comparison of RMSE for Ridge, LASSO, ElasticNet, and Stepwise Selection

Model	Training RMSE	Testing RMSE
Ridge Regression	12.34065	14.08429
LASSO Regression	12.38202	14.10810
ElasticNet (α=0.5)	12.39458	14.10695
Stepwise Selection	12.31481	14.15413

- The above image is the comparison of training and testing RMSE for Ridge, Lasso, ElasticNet and Stepwise selection models.
- The Stepwise selection model had training RMSE slightly lesser than the other 3 models while testing RMSE was slightly larger than the other 3 models.
- This indicates that minor overfitting is present in all the models but stepwise had slightly more overfitting.
- Hence, this model did not perform better than ridge, lasso and elasticnet.
- **I would prefer Ridge regression overall** as per the metrics observed compared to other models and its ability to handle correlated features without reducing irrelevant features and their coefficients to zero.

Conclusion:

1. The dataset on which I worked on was obtained from ISLR library named College dataset having 777 observations and 18 features.
2. The Ridge, LASSO, ElasticNet and Stepwise models were performed to observe their performance and conclude the best model to predict graduation rate (Grad.Rate) for the colleges.
3. All the models showed minor overfitting but it was acceptable.

	Model	Predictions
1	Ridge (Training)	57.08409
2	Ridge (Testing)	92.17908
3	Lasso (Training)	58.41340
4	Lasso (Testing)	94.35502
5	ElasticNet (α=0.5) (Training)	58.65086
6	ElasticNet (α=0.5) (Testing)	94.49596
7	Stepwise (Training)	55.98795
8	Stepwise (Testing)	92.11378

4. From the image it is observed that Ridge regression (testing data) had the 92.18% prediction rate which is slight less than Lasso and ElasticNet (testing data) but it does not reduced irrelevant coefficients to zero and is more than Stepwise (testing data).
5. Ridge considered all variables thus performing better regularization than Lasso and ElasticNet and hence, **I prefer Ridge regression model**.

References:

Books:

- Bluman, A. (2018). Elementary statistics: A step by step approach (10th ed.). McGraw Hill.ISBN 13: 978-1-259-755330.
- R. Kabacoff, *R in Action*, 2nd Edition, Manning Publisher ISBN 978-161-7291-388.

Canvas:

https://northeastern.instructure.com/courses/200351/assignments/2542453

Package:

https://cran.r-project.org/web/packages/ISLR/ISLR.pdf

Appendix:

```
library(Matrix)
library(ISLR)
View(College)

library(psych)
describe(College)
summary(College)

set.seed(20305)
library(caret)

College$Grad.Rate <- as.numeric(College$Grad.Rate)

## Splitting the dataset into training and testing
trainIndex <- sample(1:nrow(College), size = 0.8 * nrow(College), replace = FALSE)
caret_train <- College[trainIndex, ]
caret_test <- College[-trainIndex, ]

## install.packages("glmnet")
library(glmnet)
train_x <- model.matrix(Grad.Rate ~ ., data = caret_train)[,-1]
train_y <- caret_train$Grad.Rate

test_x <- model.matrix(Grad.Rate ~ ., data = caret_test)[,-1]
test_y <- caret_test$Grad.Rate

print(unique(train_y))
```

```
summary(train_y)

## Ridge regression
cv_ridge <- cv.glmnet(train_x, train_y, alpha = 0)

print(paste("Minimum Lambda for Ridge:", ridge_min_lambda <- cv_ridge$lambda.min))
print(paste("Lambda 1se for Ridge:", ridge_lambda_1se <- cv_ridge$lambda.1se))

plot(cv_ridge)
## Ridge plot
abline(v = log(cv_ridge$lambda.min), col = "cyan", lty = 4)
abline(v = log(cv_ridge$lambda.1se), col = "black", lty = 4)

ridge <- glmnet(train_x, train_y, alpha = 0, lambda = ridge_min_lambda)
print(coef_ridge <- coef(ridge))

## Ridge train rmse
predict_ridge_train <- predict(ridge, newx = train_x)
rmse_ridge_train <- sqrt(mean((train_y - predict_ridge_train)^2))
print(paste("RMSE for Ridge Training data:", rmse_ridge_train))

## Ridge test rmse
predict_ridge_test <- predict(ridge, newx = test_x)
rmse_ridge_test <- sqrt(mean((test_y - predict_ridge_test)^2))
print(paste("RMSE for Ridge Testing data:", rmse_ridge_test))

## Showing Ridge result in table
library(knitr)
```

```
ridge_result <- data.frame(
  Metric = c("Minimum Lambda", "1se Lambda", "Training RMSE", "Testing RMSE"),
  Value = c(ridge_min_lambda, ridge_lambda_1se, rmse_ridge_train, rmse_ridge_test)
)

kable(ridge_result, col.names = c("Metric", "Value"), caption = "Ridge Regression RMSE
and Lambda Values")

library(glmnet)

set.seed(20305)
## Lasso regression
cv_lasso <- cv.glmnet(train_x, train_y, alpha = 1)

print(paste("Minimum Lambda for LASSO:", lasso_min_lambda <- cv_lasso$lambda.min))
print(paste("Lambda 1se for LASSO:", lasso_lambda_1se <- cv_lasso$lambda.1se))

plot(cv_lasso)
## Lasso plot
abline(v = log(cv_lasso$lambda.min), col = "cyan", lty = 4)
abline(v = log(cv_lasso$lambda.1se), col = "black", lty = 4)

lasso <- glmnet(train_x, train_y, alpha = 1, lambda = lasso_min_lambda)
print(coef_lasso <- coef(lasso))

## Lasso train rmse
predict_lasso_train <- predict(lasso, newx = train_x)
rmse_lasso_train <- sqrt(mean((train_y - predict_lasso_train)^2))
print(paste("RMSE for Lasso Training data:", rmse_lasso_train))
```

```
## Lasso test rmse
predict_lasso_test <- predict(lasso, newx = test_x)
rmse_lasso_test <- sqrt(mean((test_y - predict_lasso_test)^2))
print(paste("RMSE for LASSO Testing data:", rmse_lasso_test))

## Showing Lasso result in table
library(knitr)

lasso_result <- data.frame(
  Metric = c("Minimum Lambda", "1se Lambda", "Training RMSE", "Testing RMSE"),
  Value = c(lasso_min_lambda, lasso_lambda_1se, rmse_lasso_train, rmse_lasso_test)
)

kable(lasso_result, col.names = c("Metric", "Value"), caption = "Lasso Regression RMSE and
Lambda Values")

library(glmnet)

set.seed(20305)
## Elastic Net regression
alpha_0.5 <- cv.glmnet(train_x, train_y, alpha = 0.5)

print(paste("Minimum Lambda for Elastic Net:", alpha_min_lambda <-
alpha_0.5$lambda.min))
print(paste("Lambda 1se for Elastic Net:", alpha_lambda_1se <- alpha_0.5$lambda.1se))

plot(alpha_0.5)
## Elastic Net plot
abline(v = log(alpha_0.5$lambda.min), col = "cyan", lty = 0.4)
```

```
abline(v = log(alpha_0.5$lambda.1se), col = "black", lty = 0.4)

alpha <- glmnet(train_x, train_y, alpha = 0.5, lambda = alpha_min_lambda)
print(coef_lambda <- coef(alpha))

## Elasticnet train rmse
predict_alpha_train <- predict(alpha, newx = train_x)
rmse_alpha_train <- sqrt(mean((train_y - predict_alpha_train)^2))
print(paste("RMSE for Elastic Net Training data:", rmse_alpha_train))

## Elasticnet test rmse
predict_alpha_test <- predict(alpha, newx = test_x)
rmse_alpha_test <- sqrt(mean((test_y - predict_alpha_test)^2))
print(paste("RMSE for Elastic Net Testing data:", rmse_alpha_test))

## Showing ElasticNet result in table
library(knitr)

alpha_result <- data.frame(
  Metric = c("Minimum Lambda", "1se Lambda", "Training RMSE", "Testing RMSE"),
  Value = c(alpha_min_lambda, alpha_lambda_1se, rmse_alpha_train, rmse_alpha_test)
)

kable(alpha_result, col.names = c("Metric", "Value"), caption = "Alpha at 0.5 Regression
RMSE and Lambda Values")

library(knitr)

combined_results <- data.frame(
```

```
  Model = c("Ridge", "LASSO", "ElasticNet (α=0.5)"),
  Training_RMSE = c(rmse_ridge_train, rmse_lasso_train, rmse_alpha_train),
  Testing_RMSE = c(rmse_ridge_test, rmse_lasso_test, rmse_alpha_test),
  Minimum_Lambda = c(ridge_min_lambda, lasso_min_lambda, alpha_min_lambda),
  Lambda_1se = c(ridge_lambda_1se, lasso_lambda_1se, alpha_lambda_1se)
)

kable(combined_results, col.names = c("Model", "Training RMSE", "Testing RMSE",
"Minimum Lambda", "Lambda 1se"),
      caption= "Comparison of Ridge, LASSO and ElasticNet")

## Stepwise selection
model <- lm(Grad.Rate ~ ., data = caret_train)
stepwise_model <- step(model, direction = "both" , trace = 0)
summary(stepwise_model)

stepwise_train <- predict(stepwise_model, newdata = caret_train)
stepwise_test <- predict(stepwise_model, newdata = caret_test)

## RMSE of Stepwise selection for Training and Testing
rmse_stepwise_train <- sqrt(mean((caret_train$Grad.Rate - stepwise_train)^2))
rmse_stepwise_test <- sqrt(mean((caret_test$Grad.Rate - stepwise_test)^2))

print(paste("Stepwise Training RMSE:", rmse_stepwise_train))
print(paste("Stepwise Testing RMSE:", rmse_stepwise_test))

## Comparing all 4 models RMSE for training and testing
library(knitr)
```

```
rmse <- data.frame(
  Model = c("Ridge Regression", "LASSO Regression", "ElasticNet (α=0.5)", "Stepwise Selection"),
  Training_RMSE = c(rmse_ridge_train, rmse_lasso_train, rmse_alpha_train, rmse_stepwise_train),
  Testing_RMSE = c(rmse_ridge_test, rmse_lasso_test, rmse_alpha_test, rmse_stepwise_test)
)

kable(rmse, col.names = c("Model", "Training RMSE", "Testing RMSE"),
      caption= "Comparison of RMSE for Ridge, LASSO, ElasticNet, and Stepwise Selection")

train_predictions <- data.frame(
  Model = c("Ridge (Training)", "Lasso (Training)", "ElasticNet (α=0.5) (Training)", "Stepwise (Training)"),
  Predictions = c(predict_ridge_train,
                  predict_lasso_train,
                  predict_alpha_train,
                  stepwise_train)
)

test_predictions <- data.frame(
  Model = c("Ridge (Testing)", "Lasso (Testing)", "ElasticNet (α=0.5) (Testing)", "Stepwise (Testing)"),
  Predictions = c(predict_ridge_test,
                  predict_lasso_test,
                  predict_alpha_test,
                  stepwise_test)
)
combined_predictions <- rbind(train_predictions, test_predictions)
```

```
## Comparing all 4 models for training and testing
combined_predictions <- data.frame(
  Model = c("Ridge (Training)", "Ridge (Testing)",
            "Lasso (Training)", "Lasso (Testing)",
            "ElasticNet (α=0.5) (Training)", "ElasticNet (α=0.5) (Testing)",
            "Stepwise (Training)", "Stepwise (Testing)"),
  Predictions = c(tail(predict_ridge_train, 1), tail(predict_ridge_test, 1),
                  tail(predict_lasso_train, 1), tail(predict_lasso_test, 1),
                  tail(predict_alpha_train, 1), tail(predict_alpha_test, 1),
                  tail(stepwise_train, 1), tail(stepwise_test, 1))
)
print(combined_predictions)
```