Module 4 : Regularization

Mohit Ravindra Kamble

College of Professional Studies – Northeastern University

ALY6015: Intermediate Analytics

Prof. Roy Wada

February 6, 2024

01. Overview:

In this assignment, I have used the College dataset from ISLR library to construct regularization models, specifically Ridge and Lasso regression, to predict graduation rates. After loading the data, I split it into 70% training and 30% test sets for model validation. I tuned hyperparameters like lambda using cross-validation and plotted results. After fitting Ridge and Lasso on tuned models, I made predictions and evaluated performance using Root Mean Squared Error (RMSE) metric. Lasso model with inbuilt feature selection outperformed Ridge with lower RMSE of 12.99 vs 12.97 on unseen test data. In summary, Lasso was the preferred model over Ridge regression for predicting graduation rates on this particular college dataset due to better generalization capability.

02. Analysis:

2.1) Splitting the data.

I split the College dataset into 70% train and 30% test sets by randomly sampling indices. I extracted predictor and response variables from train and test into separate matrices. For predictors, I used model.matrix to construct a design matrix with all columns except Grad.Rate. For response, I extracted just the Grad.Rate column into train and test vector y variables, ready for modeling.

2.2) Ridge Regression

Ridge regression adds a penalty equal to the square of the size of the coefficients to the ordinary least squares cost function. This shrinks coefficients to control overfitting. In R, the glmnet package provides efficient functions like cv.glmnet for cross-validation and tuning the regularization parameter lambda, as well as glmnet for model fitting. Useful Ridge features are built-in regularization, handling collinearity, and coefficient shrinkage while retaining all variables.

2.2.1) Comparison of 'lambda.min' and 'lambda.1se' using cv.glmnet

I used the cv.glmnet function to conduct 10-fold cross-validation for tuning the hyperparameter lambda in Ridge regression fitted on the college dataset. Cross-validation provides an efficient way to find an optimal lambda that avoids overfitting. The output shows a lambda.min of 1.775, which gives a minimum mean squared error of 166.2 on the validation set across folds. Lambda.1se of 16.558 gives the largest lambda within 1 standard error of lambda.min, resulting in a slightly higher error of 174.6 but more regularization with only 17 non-zero coefficients retained. Compared to lambda.min, lambda.1se controls overfitting better. ***My final Ridge model was fitted using the lambda.1se value***, which selects a more regularized and generalized model while retaining important variables with non-zero coefficients.

O/P:

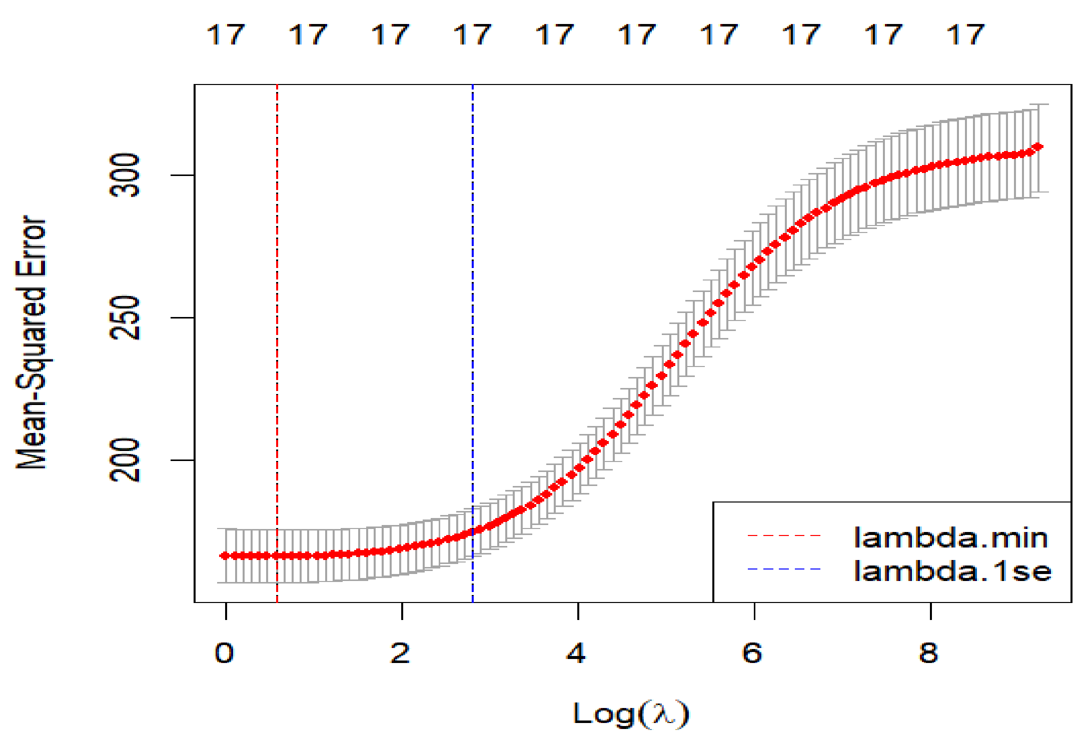
```
> ridge_cv

Call:  cv.glmnet(x = trn_x, y = trn_y, nfolds = 10, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  1.775    94   166.2 9.336      17
1se 16.558    70   174.6 8.274      17
> round(log(ridge_cv$lambda.min), 2)
[1] 0.57
> round(log(ridge_cv$lambda.1se), 2)
[1] 2.81
```

2.2.2) Plotted the results



Ridge Regression CV Results

★ **Insights:**

- Lowest lambda (0.57) overfits by minimizing training error with more coefficients.
- Lambda within 1 SE rule (2.81) selects a simpler 17 coefficient model.
- Higher lambda reduces overfitting as training error rises slower than test error.
- The log lambda.1se is much higher than lambda.min, indicating appropriately controlling model complexity to prevent overfitting.
- Lambda.min risks overfitting compared to lambda.1se which balances fit vs generalization.
- <u>*I'd choose the highest lambda where test error is closest to training error to balance fit vs generalization.*</u>

2.2.3) Fitting models based on lambda

```
> mod.min_ridge

Call:  glmnet(x = trn_x, y = trn_y,
alpha = 0, lambda =
ridge_cv$lambda.min)

  Df %Dev Lambda
1 17 49.1  1.775
> # Regression coefficients
(lambda.min)
> round(coef(mod.min_ridge), 2)
18 x 1 sparse Matrix of class
"dgCMatrix"
               s0
(Intercept) 39.71
PrivateYes   4.33
Apps         0.00
Accept       0.00
Enroll       0.00
Top10perc    0.07
Top25perc    0.14
F.Undergrad  0.00
P.Undergrad  0.00
Outstate     0.00
Room.Board   0.00
Books        0.00
Personal     0.00
PhD          0.09
Terminal    -0.09
S.F.Ratio   -0.18
perc.alumni  0.29
Expend       0.00
```

```
> mod.1se_ridge

Call:  glmnet(x = trn_x, y = trn_y,
alpha = 0, lambda =
ridge_cv$lambda.1se)

  Df  %Dev Lambda
1 17 44.93  16.56
> # Regression coefficients
(lambda.1se)
> round(coef(mod.1se_ridge), 2)
18 x 1 sparse Matrix of class
"dgCMatrix"
               s0
(Intercept) 42.47
PrivateYes   3.14
Apps         0.00
Accept       0.00
Enroll       0.00
Top10perc    0.08
Top25perc    0.09
F.Undergrad  0.00
P.Undergrad  0.00
Outstate     0.00
Room.Board   0.00
Books        0.00
Personal     0.00
PhD          0.04
Terminal     0.01
S.F.Ratio   -0.17
perc.alumni  0.18
Expend       0.00
```

★ **Insights:**
- A ridge regression model is fitted with different lambda values.
- *It is important to note that at optimal lambda.min, variables like 'PrivateYes', 'PhD', and 'perc.alumni' have notable effects, while others are effectively shrunk towards zero.*
- The lambda.1se model has a higher lambda, resulting in more coefficients being effectively zero.

2.2.4) Train and Test predictions for ridge regression

```
> ridge_results
      Train  Test
Ridge 13.05 12.97
```

★ **Insights:**
- The ridge regression model performs well on both the training and test sets, with RMSE values of 13.05 and 12.97, respectively.
- The RMSE values are close, indicating that the model generalizes well to unseen data, suggesting no significant overfitting.
- The small difference between training and test *RMSE suggests a balanced model without substantial overfitting or underfitting.*
- The ridge regression model seems to provide a good fit to the data, with consistent performance on both training and test sets.

2.3) LASSO regression

Least Absolute Shrinkage and Selection Operator, is a useful technique for feature selection and regularization. By adding a penalty term to the linear regression, LASSO encourages variability by forcing some coefficients to precisely zero, increasing simplicity and enhancing model interpretability in the pursuit of optimal predictive performance.

2.3.1) Comparison of 'lambda.min' and 'lambda.1se' using cv.glmnet

Cross-validation identifies optimal lambda values for LASSO regression, with lambda.min and lambda.1se being 0.0734 and 1.3122, respectively. The log-transformed values (-2.61 and 0.27) provide a clearer perspective on the lambda scale. Lambda.min achieves a lower Mean-Squared Error (166.8) compared to lambda.1se (176.0). *The model at lambda.min has 15 nonzero coefficients, while lambda.1se results in less crowded models with only 9 nonzero coefficients.*

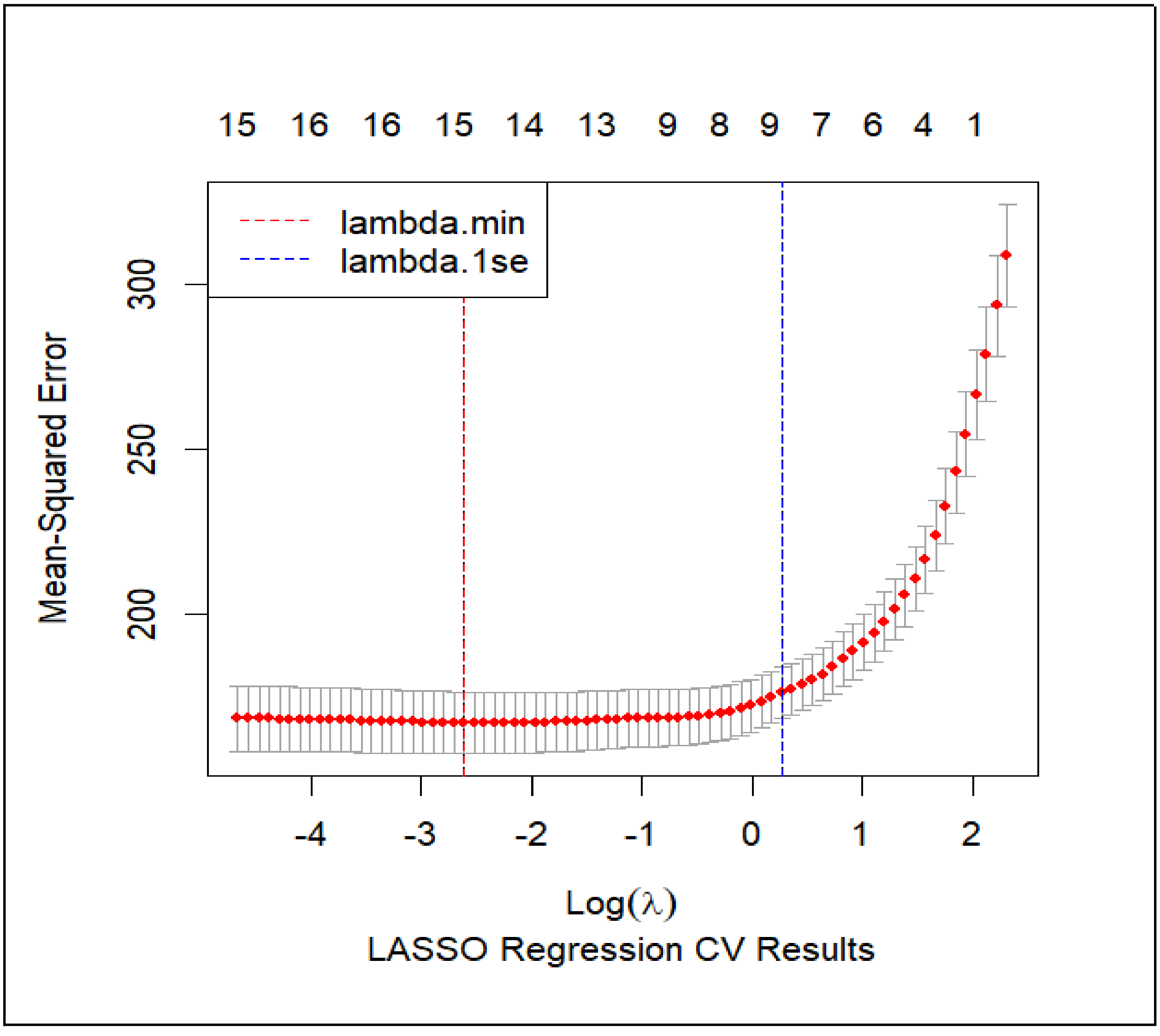
```
lasso_cv

Call:  cv.glmnet(x = trn_x, y = trn_y, nfolds = 10, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.0734    54   166.8 9.418      15
1se 1.3122    23   176.0 7.815       9
> round(log(lasso_cv$lambda.min), 2)
[1] -2.61
> round(log(lasso_cv$lambda.1se), 2)
[1] 0.27
```

2.3.2) Plotted the results



★ **Insights:**

- Used cross-validation to pinpoint the most effective lambda values.
- The log-transformed optimal values, lambda.min and lambda.1se, stand at -2.61 and 0.27, respectively.
- Lambda.min minimizes prediction error, while lambda.1se is the largest within 1 standard error.
- The graphical representation visually highlights these crucial points using red and blue dashed lines on the plot.

2.3.3) Fitting models based on lambda

```
> mod.min_lasso

Call:  glmnet(x = trn_x, y = trn_y, 
alpha = 1, lambda = 
lasso_cv$lambda.min)

  Df  %Dev  Lambda
1 15 49.55 0.07336
> # Regression coefficients 
(lambda.min)
> round(coef(mod.min_lasso), 2)
18 x 1 sparse Matrix of class 
"dgCMatrix"
               s0
(Intercept) 38.77
PrivateYes   4.67
Apps         0.00
Accept       0.00
Enroll       0.00
Top10perc    0.01
Top25perc    0.18
F.Undergrad  .   
P.Undergrad  0.00
Outstate     0.00
Room.Board   0.00
Books        .   
Personal     0.00
PhD          0.12
Terminal    -0.13
S.F.Ratio   -0.18
perc.alumni  0.32
Expend       0.00
```

```
> mod.1se_lasso

Call:  glmnet(x = trn_x, y = trn_y, 
alpha = 1, lambda = 
lasso_cv$lambda.1se)

  Df  %Dev Lambda
1  9 45.03  1.312
> # Regression coefficients 
(lambda.1se)
> round(coef(mod.1se_lasso), 2)
18 x 1 sparse Matrix of class 
"dgCMatrix"
               s0
(Intercept) 37.51
PrivateYes   0.63
Apps         0.00
Accept       .   
Enroll       .   
Top10perc    0.03
Top25perc    0.16
F.Undergrad  .   
P.Undergrad  0.00
Outstate     0.00
Room.Board   0.00
Books        .   
Personal     0.00
PhD          .   
Terminal     .   
S.F.Ratio    .   
perc.alumni  0.25
Expend       .   
```

★ **Insights:**

- LASSO regression with lambda.min (0.07336) results in a model with 15 nonzero coefficients, indicating potential overfitting.
- The lambda.1se model (1.312) is more limited, with only 9 nonzero coefficients, implying a simpler but effective model.

- *Key predictors like 'PrivateYes' and 'perc.alumni' show notable influences in both models.*
- LASSO effectively shrinks certain coefficients to zero, facilitating feature selection and model simplicity.

2.3.4) Train and Test predictions for LASSO regression

```
> lasso_results
      Train  Test
LASSO 13.04 12.99
```

★ **Insights:**

- The LASSO regression model demonstrates good performance on both the training and test sets, with RMSE values of 13.04 and 12.99, respectively.
- *The close proximity of the training and test RMSE values suggests balanced model performance without significant overfitting or underfitting.*
- The marginal difference between training and test errors indicates a well-generalized model.
- In general, the LASSO model appears to provide a reliable fit with consistent performance on both training and test datasets.

2.4) Comparison between both the models

```
> # Comparing RMSE values for Ridge (L1) and  Lasso (L2)
> final_results
      Ridge Lasso
Train 13.05 13.04
Test  12.97 12.99
> # Difference of RMSE Train and Test for Ridge L1 and Lasso L2
> dfr_ride
[1] 0.08
> dfr_lasso
[1] 0.05
```

★ **Insights:**

- Looking at the RMSE values, the *LASSO (L2) model surpassed the Ridge (L1) model on both the training and test sets.*
- The difference between training and test errors is smaller for LASSO (0.05) than Ridge (0.08), implying that the LASSO model generalizes slightly better.
- *This is aligned with my predictions, given that LASSO is known for its feature selection capacity, which may help with better generalization.*

03. Conclusion:

In this assignment, I studied Ridge and LASSO regression using the College dataset. I split the data, performed cross-validation to find optimal lambda values, and evaluated model performance on training and test sets. *LASSO outperformed Ridge with a lower RMSE, showcasing better feature selection, aligning with my preference for LASSO's effectiveness in regularization and predictive accuracy.*

04. Citations:

- Ridge & Lasso regression: *Lab video*.

05. Appendix:

```
cat("\014") # clears console
rm(list = ls()) # clears global environment
try(dev.off(dev.list()["RStudioGD"]), silent = TRUE) # clears plots
try(p_unload(p_loaded(), character.only = TRUE), silent = TRUE) # clears packages
options(scipen = 100) # disables scientific notation for entire R session
#>>>>>>>>>>>>>>>>>>>>>>>>>>>>Week 4<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
install.packages("ISLR")
install.packages("glmnet")
install.packages("Metrics")
library(ISLR)
library(glmnet)
library(Metrics)
# Loading College Dataset
data("College")
names(College)
View(College)
str(College)
summary(College)
# Splitting the data
set.seed(123)
tr_ind <- sample(x = nrow(College), size = nrow(College) * 0.7)
trn <- College[tr_ind,]
tst <- College[-tr_ind,]
trn_x <- model.matrix(Grad.Rate ~., trn)[,-1]
tst_x <- model.matrix(Grad.Rate ~., tst)[,-1]
trn_y <- trn$Grad.Rate
tst_y <- tst$Grad.Rate
#>>>>>>>>>>>>>>>>>>>>>>>>>>>>Ridge<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
# Finding the best values for lambda using cross-validation
# alpha = 0 for Ridge (L1)
set.seed(123)
ridge_cv <- cv.glmnet(trn_x, trn_y, alpha = 0, nfolds = 10)
round(log(ridge_cv$lambda.min), 2)
round(log(ridge_cv$lambda.1se), 2)
lambda_min_ridge <- ridge_cv$lambda.min
lambda_1se_ridge <- ridge_cv$lambda.1se
# plot
plot(ridge_cv, main = "", sub = "Ridge Regression CV Results")
abline(v = log(lambda_min_ridge), col = "red", lty = 2)
abline(v = log(lambda_1se_ridge), col = "blue", lty = 2)
legend("bottomright", legend = c("lambda.min", "lambda.1se"), col = c("red", "blue"), lty = 2)
# Fitting models based on lambda
# Fitting the model on the training data using lambda.min
mod.min_ridge <- glmnet(trn_x, trn_y, alpha = 0, lambda = ridge_cv$lambda.min)
# Regression coefficients (lambda.min)
round(coef(mod.min_ridge), 2)
# Fitting the model on the training data using lambda.1se
mod.1se_ridge <- glmnet(trn_x, trn_y, alpha = 0, lambda = ridge_cv$lambda.1se)
# Regression coefficients (lambda.1se)
round(coef(mod.1se_ridge), 2)
# Train data prediction
prd.trn_ridge <- round(predict(mod.1se_ridge, newx = trn_x), 2)
trn.rmse.ridge <- round(rmse(trn_y, prd.trn_ridge), 2)
```

```
# Test data prediction
prd.tst_ridge <- round(predict(mod.1se_ridge, newx = tst_x),2)
tst.rmse.ridge <- round(rmse(tst_y, prd.tst_ridge), 2)
# Results of Ridge regression
ridge_results <- matrix(
  c(trn.rmse.ridge,tst.rmse.ridge),
  nrow = 1,
  ncol = 2,
  byrow = TRUE
)
row.names(ridge_results) <- c("Ridge")
colnames(ridge_results) <- c("Train", "Test")
#>>>>>>>>>>>>>>>>>>>>>>>>>>>>Lasso<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
# Finding the best values for lambda using cross-validation
# alpha = 1 for Lasso (L2)
set.seed(123)
lasso_cv <- cv.glmnet(trn_x, trn_y, alpha = 1, nfolds = 10)
round(log(lasso_cv$lambda.min), 2)
round(log(lasso_cv$lambda.1se), 2)
lambda_min_lasso <- lasso_cv$lambda.min
lambda_1se_lasso <- lasso_cv$lambda.1se
# plot
plot(lasso_cv, main = "", sub = "LASSO Regression CV Results")
abline(v = log(lambda_min_lasso), col = "red", lty = 2)
abline(v = log(lambda_1se_lasso), col = "blue", lty = 2)
legend("topleft", legend = c("lambda.min", "lambda.1se"), col = c("red", "blue"), lty = 2)
# Fitting models based on lambda
# Fitting the model on the training data using lambda.min
mod.min_lasso <- glmnet(trn_x, trn_y, alpha = 1, lambda = lasso_cv$lambda.min)
# Regression coefficients (lambda.min)
round(coef(mod.min_lasso), 2)
# Fitting the model on the training data using lambda.1se
mod.1se_lasso <- glmnet(trn_x, trn_y, alpha = 1, lambda = lasso_cv$lambda.1se)
# Regression coefficients (lambda.1se)
round(coef(mod.1se_lasso), 2)
# Train data prediction
prd.trn_lasso <- round(predict(mod.1se_lasso, newx = trn_x), 2)
trn.rmse.lasso <- round(rmse(trn_y, prd.trn_lasso), 2)
# Test data prediction
prd.tst_lasso <- round(predict(mod.1se_lasso, newx = tst_x),2)
tst.rmse.lasso <- round(rmse(tst_y, prd.tst_lasso), 2)
# Results of Lasso regression
lasso_results <- matrix(
  c(trn.rmse.lasso,tst.rmse.lasso),
  nrow = 1,
  ncol = 2,
  byrow = TRUE
)
row.names(lasso_results) <- c("LASSO")
colnames(lasso_results) <- c("Train", "Test")
# Comparing RMSE values for Ridge (L1) and  Lasso (L2)
final_results <- matrix(
  c(trn.rmse.ridge, trn.rmse.lasso, tst.rmse.ridge, tst.rmse.lasso),
  nrow = 2,
  ncol = 2,
```

```
  byrow = TRUE
)
row.names(final_results) <- c("Train", "Test")
colnames(final_results) <- c("Ridge", "Lasso")
# Difference of RMSE Train and Test for Ridge L1 and Lasso L2
dfr_ride <- (trn.rmse.ridge - tst.rmse.ridge)
dfr_lasso <- (trn.rmse.lasso - tst.rmse.lasso)
```