Sean McLean

ALY 6015

Module 4 Assignment

Regularization

Introduction

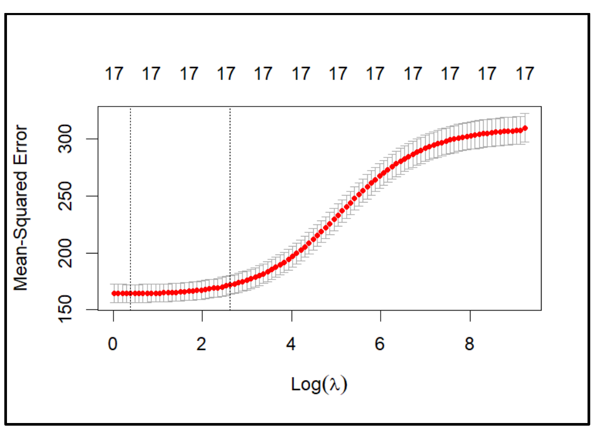
The mission of this assignment is to show how a dependent variable can be predicted with the usage of regularization models. We will be using the same dataset as the previous assignment to demonstrate which of the used models like Ridge and Lasso will perform better. The previous dataset used is called ‘college’ which contains 18 variables and 777 rows, and the predicted variable we will use is called ‘Grad.rate’. After looking at the performances of both the lasso and ridge models, a stepwise model will be created to see if it has better results than the previous modeling techniques.

Analysis: Ridge Regression

1. The first step is to break up the college data set into training and test sets as was previously done in the last assignment. This is a customary practice in machine learning where the dataset is divided into a pair of subsets with the training data going through a process where the model identifies and learns patterns and trends in the data. The training data like the previous assignment will also be 70 percent of the dataset. The remaining portion will be the test set that will be used to look at the model’s performance on data not encountered before (Barkved, 2022). The two different subsets are put into variables and then matrix variables where the ‘Grad.rate’ variable is removed and put into a separate variable for regularization modeling.

2. and 3. The ridge regression model will be built first by using the cv.glmnet function to establish the lambda.min and lambda.lse values and then will plot them. The lambda parameter is important in the regularization process, performing tasks like keeping overfitting under control and shrinking coefficients.

```
> set.seed(123)
> cv.ridge <- cv.glmnet(train_x, train_y, alpha = 0, nfolds = 10)
> log(cv.ridge$lambda.min)
[1] 0.3880099
> log(cv.ridge$lambda.1se)
[1] 2.62082
```



The lambda.min value is 0.388 which indicates that it could be less sparse with regards to the coefficients in the model. The lambda.1se value of 2.621 means that it could be sparser and the coefficients being smaller. In the plot the vertical dotted lines show where the lambda.min and lambda.1se values fall, and the number of variables from the data set all remain which could potentially mean that they are all still relevant to the model.

4. Using the cv.glmnet function, a ridge regression model is fitted, and the coefficients are compared between lambda.min and lambda.1se followed by the coefficients with no regularization.

```
> coef(model.ridge)
18 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept)   3.972866e+01
PrivateYes    4.373132e+00
Apps          8.612021e-04
Accept        1.632762e-04
Enroll       -3.615392e-04
Top10perc     6.480415e-02
Top25perc     1.426209e-01
F.Undergrad  -4.552089e-05
P.Undergrad  -1.185306e-03
Outstate      6.398481e-04
Room.Board    2.043375e-03
Books        -5.169888e-04
Personal     -2.939102e-03
PhD           1.004773e-01
Terminal     -9.560248e-02
S.F.Ratio    -1.801664e-01
perc.alumni   2.961645e-01
Expend       -3.614924e-04
```

```
> coef(model_1se_ridge)
18 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept)   4.189904e+01
PrivateYes    3.293228e+00
Apps          2.940924e-04
Accept        2.340064e-04
Enroll        3.715852e-05
Top10perc     8.291118e-02
Top25perc     9.466930e-02
F.Undergrad  -4.208574e-05
P.Undergrad  -8.630806e-04
Outstate      4.870042e-04
Room.Board    1.492526e-03
Books        -6.175465e-04
Personal     -2.300453e-03
PhD           4.525940e-02
Terminal      4.658045e-04
S.F.Ratio    -1.695197e-01
perc.alumni   1.907969e-01
Expend       -2.280042e-06
```

```
> coef(ols_ridge)
  (Intercept)    PrivateYes          Apps        Accept        Enroll     Top10perc
 3.989851e+01  4.853194e+00  1.737966e-03 -8.479738e-04 -9.953913e-04 -3.837352e-03
    Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board         Books
 1.834768e-01  6.592359e-05 -1.290497e-03  7.473591e-04  2.044515e-03 -2.903097e-04
     Personal           PhD      Terminal     S.F.Ratio   perc.alumni        Expend
-2.969186e-03  1.574857e-01 -1.595563e-01 -2.197230e-01  3.290068e-01 -5.437189e-04
```

It appears from the three models that there is not much difference between all the variables. The most interesting aspect of the models is that they look similar even when there is no regularization applied.

5. & 6. The ridge regression fit model is then looked at against the training and test sets to see how well it performs. This is shown by calculating the root mean square error for each pair as well as the root mean square error of the full model.

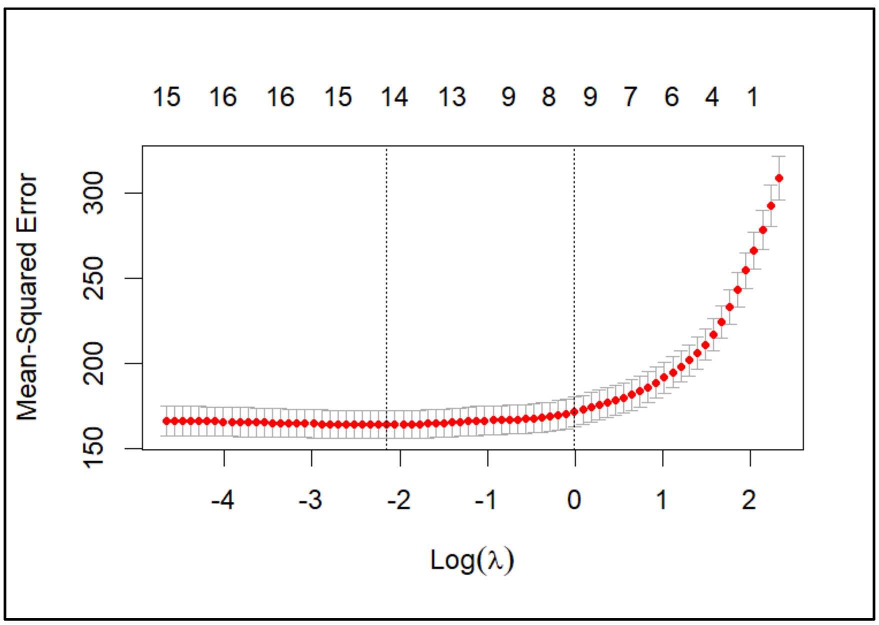
```
> #Compare rmse values
> ridge_rmse_train
[1] 12.96324
> ridge_rmse_test
[1] 12.9456
> rmse(test$Grad.Rate, preds_ols_ridge)
[1] 13.30244
```

From these fit model values, the training and test RMSE values are quite close. This suggests that the model is performing similarly on both the training and test data, which is a good sign. Compared to the full model the values are also close, so it does not look like overfitting from the training and test sets is a concern.

Analysis: Lasso Regression

7. & 8. The lasso regression model will be built first by using the cv.glmnet function to establish the lambda.min and lambda.lse values and then will be put into a plot.

```
> set.seed(123)
> cv.lasso <- cv.glmnet(train_x, train_y, alpha = 1, nfolds = 10)
> log(cv.lasso$lambda.min)
[1] -2.14716
> log(cv.lasso$lambda.1se)
[1] -0.007383546
```



The lambda.min and lambda.1se values are both small, and when plotted the lambda.min value has removed three variables from the table and the lambda.1se value has removed eight variables.

9. Implementing the cv.glmnet function, a lasso regression model is fitted, and the coefficients are compared between lambda.min and lambda.1se followed by the coefficients with no regularization.

```
> coef(model_lasso)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept) 38.1215088695
PrivateYes   4.6092706321
Apps         0.0010515998
Accept       .
Enroll      -0.0004925081
Top10perc    0.0159020922
Top25perc    0.1749512633
F.Undergrad  .
P.Undergrad -0.0012492900
Outstate     0.0006818643
Room.Board   0.0020495405
Books        .
Personal    -0.0029337997
PhD          0.1059159235
Terminal    -0.1090245955
S.F.Ratio   -0.1559738443
perc.alumni  0.3194285107
Expend      -0.0003837549
```

```
> coef(model_1se_lasso)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept) 36.4479897891
PrivateYes   1.8353316358
Apps         0.0002700320
Accept       .
Enroll       .
Top10perc    0.0098867060
Top25perc    0.1678576382
F.Undergrad  .
P.Undergrad -0.0007608107
Outstate     0.0007425247
Room.Board   0.0016429893
Books        .
Personal    -0.0021294365
PhD          .
Terminal     .
S.F.Ratio    .
perc.alumni  0.2686584846
Expend       .
```

```
> coef(ols_lasso)
  (Intercept)    PrivateYes          Apps        Accept        Enroll     Top10perc
 3.989851e+01  4.853194e+00  1.737966e-03 -8.479738e-04 -9.953913e-04 -3.837352e-03
    Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board         Books
 1.834768e-01  6.592359e-05 -1.290497e-03  7.473591e-04  2.044515e-03 -2.903097e-04
     Personal           PhD      Terminal     S.F.Ratio   perc.alumni        Expend
-2.969186e-03  1.574857e-01 -1.595563e-01 -2.197230e-01  3.290068e-01 -5.437189e-04
```

Between the lambda models there have been some variables that have been eliminated, with the variables ‘Accept’, ‘F. Underground’, and ‘Books’ being removed from both models. Overall, the coefficients from the lambda.min and lambda.1se models are both a lot smaller than the fitted ridge regression models with a majority of the coefficients being very close to zero.

10. & 11. The lasso regression fit model is then evaluated against the training and test sets to see how well they perform. This is done by calculating the root mean square error for each pair as well as the root mean square error of the full model.

```
> #Compare rmse values
> train_rmse_lasso
[1] 12.87856
> test_rmse_lasso
[1] 12.91357
> rmse(test$Grad.Rate, preds_ols_lasso)
[1] 13.30244
>
```

Similar to the root mean square error values of the ridge regression fit models, the training and test RMSE values are also quite close. This suggests that the model is performing similarly on both the training and test data. Additionally, both training and test set RMSE values are lower than the RMSE of the full model. These observations indicate that the model is not exhibiting significant overfitting.

12. Comparing the ridge and lasso models to the test set, their root mean square error values are alike. They would both perform well for this data set, but if one regression model needed to be chosen then the lasso RMSE (12.91357) would get the edge over the ridge RMSE (12.9456).

13. The stepwise selection is performed to compare against the ridge and lasso regression models followed by a fitted model to determine whether this method is an improvement.

```
> summary(model_step)

Call:
lm(formula = Grad.Rate ~ Private + Apps + Top25perc + P.Undergrad +
    Outstate + Room.Board + Personal + PhD + Terminal + perc.alumni +
    Expend, data = College)

Residuals:
    Min      1Q  Median      3Q     Max
-51.684  -7.488  -0.282   7.363  53.482

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.4888648  3.3489573  10.000  < 2e-16 ***
PrivateYes   3.5847682  1.6283712   2.201  0.02800 *
Apps         0.0008950  0.0001609   5.563 3.67e-08 ***
Top25perc    0.1697318  0.0321993   5.271 1.76e-07 ***
P.Undergrad -0.0016749  0.0003631  -4.613 4.65e-06 ***
Outstate     0.0010061  0.0002257   4.458 9.51e-06 ***
Room.Board   0.0018799  0.0005795   3.244  0.00123 **
Personal    -0.0018516  0.0007485  -2.474  0.01358 *
PhD          0.0997365  0.0554704   1.798  0.07257 .
Terminal    -0.0950484  0.0612000  -1.553  0.12082
perc.alumni  0.2887259  0.0484841   5.955 3.96e-09 ***
Expend      -0.0003942  0.0001290  -3.055  0.00233 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.73 on 765 degrees of freedom
Multiple R-squared:  0.4585,	Adjusted R-squared:  0.4507
F-statistic: 58.88 on 11 and 765 DF,  p-value: < 2.2e-16
```

```
> coef(model_step)
  (Intercept)    PrivateYes          Apps      Top25perc    P.Undergrad      Outstate
33.4888648432  3.5847682354  0.0008949751  0.1697317832 -0.0016748544  0.0010060810
   Room.Board      Personal           PhD       Terminal    perc.alumni        Expend
 0.0018799059 -0.0018516261  0.0997365432 -0.0950484084  0.2887259430 -0.0003942095
> stepwise_rmse
[1] 12.74155
```

The calculated root mean square error of 12.74155 is a small improvement over the lasso regression model. The coefficients of the variables that remain after the model is performed are also closer to zero, and the intercept of 33.48 from the table is smaller than the lasso and ridge regression models. I would prefer to use this regression model over the previous ones because of these factors along with the benefits of stepwise selection that include reducing the risk of overfitting and its ability to improve the model's overall performance.

Conclusion

The assignment covered several regularization methods on a dataset to see which regression model would perform the best on a selected dependent variable. Each function used had reliable results after the training and test sets from the data set were implemented and all showed evidence of the data having a lower chance of overfitting. My choice of the regression models would be the stepwise selection for having the lowest root mean square error and for having the capability to identify the right predictors for the response variable.

Appendix

```r
#Module 4 Assignment - Regularization

install.packages("ISLR")
library(ISLR)
install.packages("pROC")
library(pROC)
install.packages("caret")
library(caret)
library(ggplot2)
library(gridExtra)
install.packages("glmnet")
library(glmnet)
install.packages("Metrics")
library(Metrics)
attach(College)

#Question 1: Split the data into a train and test set - refer to the Feature_Selection_R.pdf
#document for information on how to split a data set.

#Load data
data(College)
dataset <- College

#Split data into train and test sets
set.seed(123)
trainIndex <- sort(sample(x = nrow(dataset), size = nrow(dataset) * 0.7))
train <- dataset[trainIndex,]
test <- dataset[-trainIndex,]

train_x <- model.matrix(Grad.Rate ~., train)[,-1]
```

```r
test_x <- model.matrix(Grad.Rate ~., test)[,-1]

train_y <- train$Grad.Rate
test_y <- test$Grad.Rate

head(train_x, n = 10)
head(test_x)
head(train_y, n = 10)
head(test_y)

#Question 2:  Use the cv.glmnet function to estimate the lambda.min and lambda.1se values. Compare and
#discuss the values.
set.seed(123)
cv.ridge <- cv.glmnet(train_x, train_y, alpha = 0, nfolds = 10)

log(cv.ridge$lambda.min)
log(cv.ridge$lambda.1se)

#Question 3: Plot the results from the cv.glmnet function provide an interpretation. What does this plot
#tell us?
plot(cv.ridge)

#Question 4: Fit a Ridge regression model against the training set and report on the coefficients. Is there
#anything interesting?
model.ridge <- glmnet(train_x, train_y, alpha = 0, lambda = cv.ridge$lambda.min)
model.ridge

#Display regression coefficients
coef(model.ridge)
```

```
#Fit the final model on the training data using lambda.lse
model_lse_ridge <- glmnet(train_x, train_y, alpha = 0, lambda = cv.ridge$lambda.1se)

#Display regression coefficients
coef(model_lse_ridge)

#Display coefficients of ols model with no regularization
ols_ridge <- lm(Grad.Rate ~ ., data = train)
coef(ols_ridge)

#Question 5: Determine the performance of the fit model against the training set by calculating the root
#mean square error (RMSE). sqrt(mean((actual - predicted)^2))

#View RMSE of full model
preds_ols_ridge <- predict(ols_ridge, new = test)
rmse(test$Grad.Rate, preds_ols_ridge)

#Make predictions on the train data using lambda.min
preds_train_ridge <- predict(model_lse_ridge, newx = train_x)
train_rmse_ridge <- rmse(train_y, preds_train_ridge)

#Compute the RMSE
ridge_rmse_train <- sqrt(mean((train_y - preds_train_ridge)^2))
ridge_rmse_train

#Question 6: Determine the performance of the fit model against the test set by calculating the root mean
#square error (RMSE). Is your model overfit?

preds_test_ridge <- predict(model_lse_ridge, newx = test_x)
test_rmse_ridge <- rmse(test_y, preds_test_ridge)
```

```

#Compute the RMSE
ridge_rmse_test <- sqrt(mean((test_y - preds_test_ridge)^2))
ridge_rmse_test

#Compare rmse values
ridge_rmse_train
ridge_rmse_test
rmse(test$Grad.Rate, preds_ols_ridge)

#Question 7: Use the cv.glmnet function to estimate the lambda.min and lambda.1se values. Compare and
#discuss the values.
set.seed(123)
cv.lasso <- cv.glmnet(train_x, train_y, alpha = 1, nfolds = 10)

log(cv.lasso$lambda.min)
log(cv.lasso$lambda.1se)

#Question 8:  Plot the results from the cv.glmnet function provide an interpretation. What does this plot
#tell us?
plot(cv.lasso)

#Question 9:  Fit a LASSO regression model against the training set and report on the coefficients. Do any
#coefficients reduce to zero? If so, which ones?
model_lasso <- glmnet(train_x, train_y, alpha = 1, lambda = cv.lasso$lambda.min)
model_lasso

#Display regression coefficients
coef(model_lasso)

```

```
#Fit the final model on the training data using lambda.1se
model_lse_lasso <- glmnet(train_x, train_y, alpha = 1, lambda = cv.lasso$lambda.1se)

#Display regression coefficients
coef(model_lse_lasso)

#Display coefficients of ols model with no regularization
ols_lasso <- lm(Grad.Rate ~ ., data = train)
coef(ols_lasso)

#Question 10: Determine the performance of the fit model against the training set by calculating the root
#mean square error (RMSE). sqrt(mean((actual - predicted)^2))

#View RMSE of full model
preds_ols_lasso <- predict(ols_lasso, new = test)
rmse(test$Grad.Rate, preds_ols_lasso)

#Make predictions on the train data using lambda.min
preds_train_lasso <- predict(model_lse_lasso, newx = train_x)
train_rmse_lasso <- rmse(train_y, preds_train_lasso)

#Compute the RMSE
lasso_rmse <- sqrt(mean((train_y - preds_train_lasso)^2))
lasso_rmse

#Question 11: Determine the performance of the fit model against the test set by calculating the root mean
#square error (RMSE). Is your model overfit?

preds_test_lasso <- predict(model_lse_lasso, newx = test_x)
test_rmse_lasso <- rmse(test_y, preds_test_lasso)
```

```r
test_rmse_lasso <- rmse(test_y, preds_test_lasso)

#Compare rmse values
train_rmse_lasso
test_rmse_lasso
rmse(test$Grad.Rate, preds_ols_lasso)

#Comparison
#Question 12: Which model performed better and why? Is that what you expected?

#Question 13: Refer to the Intermediate_Analytics_Feature_Selection_R.pdf document for how to perform stepwise
#selection and then fit a model. Did this model perform better or as well as Ridge regression
#or LASSO? Which method do you prefer and why?

# Stepwise selection method and fit the model
step(lm(Grad.Rate ~ ., data = College), direction = 'both')
model_step <- step(lm(Grad.Rate ~ ., data = College), direction = 'both')
summary(model_step)
coef(model_step)

#Make predictions on Grad Rate
stepwise <- predict(model_step, newdata = test)
stepwise

#Compute the RMSE
stepwise_rmse <- sqrt(mean((test$Grad.Rate - stepwise)^2))
stepwise_rmse
```

References

Barkved, K. (2022, Feb. 11th). *The Difference Between Training Data vs. Test Data in Machine Learning. Obviously.ai.* https://www.obviously.ai/post/the-difference-between-training-data-vs-test-data-in-machine-learning

Breheny, P. & Zeng, Y. (2018). The biglasso Package: A Memory- and Computation-Efficient Solver for Lasso Model Fitting with Big Data in R. *Journal of Statistical Software, Volume VV, Issue II.* https://arxiv.org/pdf/1701.05936.pdf

Chat GPT. (2023, December 10th). Default (GPT 3.5). < https://chat.openai.com/>