Regularization

Name : Shivani Vellanki
Date : April 1, 2023.

Introduction:

In this project, we'll utilize the College dataset from the R ISLR package to carry out Ridge and Lasso regressions. To find the best model, we'll use the glmnet package to implement regularization techniques like Ridge and Lasso regression. We'll assess the accuracy of the regression models by calculating the root mean squared error. Understanding the significance of the involved values, such as lambda, deviation, mean-squared error, coefficients, and RMSE, is important for a comprehensive evaluation of the process results.

Analysis:

Importing Dataset:

```
> # Import dataset
> college_data <- ISLR::College
> view(college_data)
```

Splitting dataset to Train and Test Set:

```
> # Import dataset
> college_data <- ISLR::College
> view(college_data)
> # Split the dataset into train and test
> set.seed(123)
> trainIndex <- createDataPartition(college_data$Private, p = 0.7, list = FALSE)
> train_data <- college_data[trainIndex, ]
> test_data <- college_data[-trainIndex, ]
>
> # Create matrix
> train_matrix <- model.matrix(Grad.Rate ~ ., train_data)[,-1]
> test_matrix <- model.matrix(Grad.Rate ~ ., test_data)[,-1]
>
> train_y <- train_data$Grad.Rate
> test_y <- test_data$Grad.Rate
```

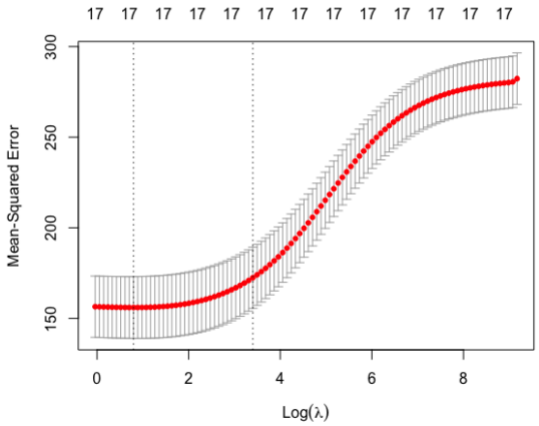
- The College Dataset is divided into two sets - train and test. The train set contains 70% of the data, and the remaining 30% is allocated to the test set. We create a matrix for both the train and test sets, which exclude the response variable, while the train_y and test_y sets contain the response variable..

Ridge Regression:

```
> # Perform Ridge regression
> set.seed(123)
> cv_ridge <- cv.glmnet(train_matrix, train_y, alpha = 0, nfolds = 10)
> cv_ridge$lambda.min
[1] 2.218028
> cv_ridge$lambda.1se
[1] 30.01099
```

- From the above result, Ridge regression with a lambda value of 2.218028 would result in the most accurate predictions. However, using lambda.1se with a value of 30.01099 may produce a simpler model with only a slightly higher level of error.

Plotting the Result:



- The graph shows how the error changes as we use different values of lambda for our model. The x-axis has numbers that go up a lot because they are shown in a special way called "log scale." The y-axis shows the amount of error we get. We can see that the number of things we use to make our model stays the same, but the amount of error changes. The dotted lines show us which values of lambda are best to use. It's recommended to use the one with the smallest error and the same number of things in the model as the others.

Fitting Ridge Model with Lambda.min :

```
> # Create models using lambda.min and lambda.1se
> model_min_ridge <- glmnet(x = train_matrix, y = train_y, alpha = 0, lambda = cv_ridge$lambda.min)
> model_min_ridge

Call:  glmnet(x = train_matrix, y = train_y, alpha = 0, lambda = cv_ridge$lambda.min)

  Df  %Dev Lambda
1 17 47.43  2.218
> coef(model_min_ridge)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  3.250032e+01
PrivateYes   4.562074e+00
Apps         5.524097e-04
Accept       1.546364e-04
Enroll       1.082462e-03
Top10perc    9.598412e-02
Top25perc    1.162124e-01
F.Undergrad -8.977996e-05
P.Undergrad -1.416583e-03
Outstate     7.020685e-04
Room.Board   2.704523e-03
Books       -2.702141e-04
Personal    -1.880439e-03
PhD          5.424212e-02
Terminal    -6.527607e-02
S.F.Ratio    9.470215e-02
perc.alumni  2.342565e-01
Expend      -2.970306e-04
```

- From the above result , when lambda.min equals 2.218, the resulting model has 18 variables. The coefficients indicate that PrivateYes, Top10perc, Top25perc, S.F.Ratio, and perc.alumni are positively important, while F.Undergrad, P.Undergrad, Books, Personal, Expend, and Terminal have negative importance. This means that these variables play a significant role in predicting the outcome variable, and regularization is helpful in avoiding overfitting.

Fitting Ridge Model with Lambda 1se:

```
> model_min_ridge2 <- glmnet(x = train_matrix, y = train_y, alpha = 0, lambda = cv_ridge$lambda.1se)
> model_min_ridge2

Call:  glmnet(x = train_matrix, y = train_y, alpha = 0, lambda = cv_ridge$lambda.1se)

  Df  %Dev Lambda
1 17 39.83  30.01
> coef(model_min_ridge2)
18 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  4.302678e+01
PrivateYes   2.494120e+00
Apps         1.755903e-04
Accept       1.701406e-04
Enroll       2.012604e-04
Top10perc    7.016730e-02
Top25perc    6.720795e-02
F.Undergrad -2.052740e-05
P.Undergrad -7.651117e-04
Outstate     3.939450e-04
Room.Board   1.325726e-03
Books        5.042397e-04
Personal    -1.307279e-03
PhD          3.375143e-02
Terminal     1.917211e-02
S.F.Ratio   -1.116679e-01
perc.alumni  1.205502e-01
Expend       9.284541e-05
```

- From the above result, when lambda.1se equals 30.01, a model is built with 18 variables, and its coefficients reveal that certain variables, such as PrivateYes, Top10perc, Top25perc, S.F.Ratio, and perc.alumni, have positive coefficients, while others like F.Undergrad, P.Undergrad, Personal, and S.F.Ratio have negative coefficients, indicating their relevance in predicting the outcome variable. Nevertheless, the cross-validation error of this model is higher than that of the model with lambda.min, indicating that it may have fewer significant variables and may be more susceptible to overfitting.

Performance Check - Training Set:

```
> # Check the performance against training set for lambda.min
> predict_ridge_min <- predict(model_min_ridge, newx = train_matrix)
> RMSE(train_data$Grad.Rate, predict_ridge_min)
[1] 12.15428
> # Check the performance against training set for lambda.1se
> predict_ridge_min2 <- predict(model_min_ridge2, newx = train_matrix)
> RMSE(train_data$Grad.Rate, predict_ridge_min2)
[1] 13.00394
```

- From the above result, the RMSE for lambda min is 12.15428 and the RMSE for lambda 1se is 13.00394.

Performance Check - Test Set:

```
> predict_ridge_min_test <- predict(model_min_ridge, newx = test_matrix)
> RMSE(test_data$Grad.Rate, predict_ridge_min_test)
[1] 13.87404
> predict_ridge_min_test2 <- predict(model_min_ridge2, newx = test_matrix)
> RMSE(test_data$Grad.Rate, predict_ridge_min_test2)
[1] 14.36223
```

- From the above result, the RMSE for lambda min is 13.87404 and the RMSE for lambda 1se is 14.36223.
- The first model (lambda.min) has a lower RMSE (12.15) than the second model (lambda.1se) on the training set, indicating better performance. However, on the test set, lambda.min still performs better (RMSE of 13.87) than lambda.1se (RMSE of 14.36).
- There might be overfitting in the model, as the root mean square error (RMSE) on the training set is smaller than the RMSE on the test set for both lambda.min and lambda.1se models. This indicates that the model might perform worse on new, unseen data, suggesting that it has overfit to the training data.

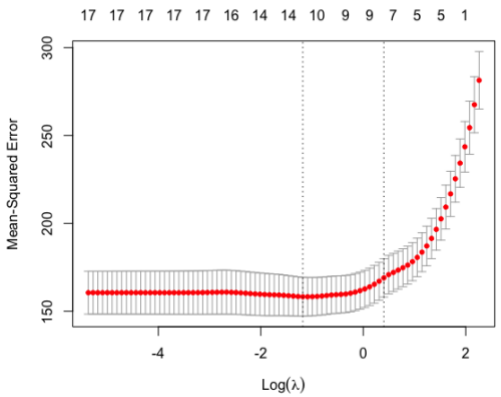
Lasso Regression:

```
> # Use cv.glmnet function to estimate lambda
> cv_lasso <- cv.glmnet(x = train_matrix, y = train_y, nfolds = 10)
> cv_lasso$lambda.min
[1] 0.3071717
> cv_lasso$lambda.1se
[1] 1.493654
> cv_lasso

Call:  cv.glmnet(x = train_matrix, y = train_y, nfolds = 10)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.3072    38   158.2 11.08      12
1se 1.4937    21   169.0 10.93       9
```



- We used the cv.glmnet function to find the best lambda value for LASSO regression. The lambda value that resulted in the lowest mean-squared error (MSE) was 0.3071717, and it had 12 non-zero coefficients. Additionally, a lambda value of 1.493654 with 9 non-zero coefficients was found to be one standard error above the

minimum lambda value. Through 10-fold cross-validation, we concluded that LASSO regularization helps prevent overfitting, as compared to using an unpenalized model.

Lasso Regression against Training Set:

```
> # Create Lasso regression model against the training set
> model_min <- glmnet(x = train_matrix, y = train_y, lambda = cv_lasso$lambda.min)
>
> coef(model_min)
18 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  3.283164e+01
PrivateYes   3.907038e+00
Apps         7.310462e-04
Accept       .
Enroll       2.771014e-05
Top10perc    6.315957e-02
Top25perc    1.340846e-01
F.Undergrad  .
P.Undergrad -1.296579e-03
Outstate     7.285431e-04
Room.Board   2.649733e-03
Books        .
Personal    -1.660685e-03
PhD          .
Terminal    -5.565326e-03
S.F.Ratio    .
perc.alumni  2.416358e-01
Expend      -2.874062e-04
```

- The above output implements a Lasso regression model using the glmnet function and selects the lambda value through cross-validation. The coef function is then utilized to obtain the coefficients of the model. The output displays the coefficients of the variables that have non-zero values in the model, which suggests that these variables are significant in predicting the response variable.

```
> model_min2 <- glmnet(x = train_matrix, y = train_y, lambda = cv_lasso$lambda.1se)
>
> coef(model_min2)
18 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  3.586008e+01
PrivateYes   2.679927e-01
Apps         7.911651e-05
Accept       .
Enroll       .
Top10perc    2.957653e-02
Top25perc    1.281738e-01
F.Undergrad  .
P.Undergrad -3.549032e-04
Outstate     8.428707e-04
Room.Board   2.026670e-03
Books        .
Personal    -6.260847e-04
PhD          .
Terminal     .
S.F.Ratio    .
perc.alumni  1.992752e-01
Expend       .
```

- The column s0 displays the intercept value, while the other columns represent the coefficients of the predictor variables. If a coefficient is not equal to zero, it suggests that the model has selected that variable as important. In this scenario, only nine coefficients are non-zero, indicating that the Lasso regression model has performed variable selection and has excluded some predictors.

Performance Check - Train Set:

```
> predict_lasso_min <- predict(model_min, newx = train_matrix)
> RMSE(train_data$Grad.Rate, predict_lasso_min)
[1] 12.20319
>
> predict_lasso_1se <- predict(model_min2, newx = train_matrix)
> RMSE(train_data$Grad.Rate, predict_lasso_1se)
[1] 12.74556
```

- From the above result, the RMSE for lambda min is 12.20319 and the RMSE for lambda 1se is 12.74556.

Performance Check - Test Set:

```
> predict_lasso_1se <- predict(model_min2, newx = train_matrix)
> RMSE(train_data$Grad.Rate, predict_lasso_1se)
[1] 12.74556
> predict_lasso.min.test=predict(model.min1,newx = test_matrix)
> RMSE(test_data$Grad.Rate,predict_lasso.min.test)
[1] 13.36622
```

- From the above result, the RMSE for lambda min is 12.74556 and the RMSE for lambda 1se is 13.36622

- We calculated the root mean squared error (RMSE) to evaluate how well the Lasso regression models were performing on the training set. The RMSE value for the model using the minimum lambda value was 12.20, while the RMSE value for the model using the 1se lambda value was 12.75. To evaluate the performance on the test set, the Lasso regression model using the minimum lambda value was employed to predict Grad.Rate, and the resulting RMSE was found to be 13.37.

Which model performed better?

The LASSO and Ridge are methods used in statistics to build models that help to understand and predict outcomes based on different variables. The LASSO method tends to make models that are easier to understand because they have fewer variables, while the Ridge method keeps all variables. Although both methods give similar results in terms of the accuracy of predictions, the LASSO method tends to give coefficients that are close to zero, which can make the model simpler to use.

Stepwise Selection:

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

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
```

```
> stepwise_model$coefficients
  (Intercept)    PrivateYes          Apps     Top25perc   P.Undergrad      Outstate    Room.Board
33.4888648432  3.5847682354  0.0008949751  0.1697317832 -0.0016748544  0.0010060810  0.0018799059
     Personal           PhD      Terminal   perc.alumni        Expend
-0.0018516261  0.0997365432 -0.0950484084  0.2887259430 -0.0003942095
```

- The above result in equation format is

  Grad.Rate = 33.4889 + 3.5848(PrivateYes) + 0.0009(Apps) + 0.1697(Top25perc) - 0.0017(P.Undergrad) + 0.0010(Outstate) + 0.0019(Room.Board) - 0.0019(Personal) + 0.0997(PhD) - 0.0950(Terminal) + 0.2887(perc.alumni) - 0.0004(Expend)

  The stepwise and ridge models have similar accuracy measures, and both start with a model containing all variables and then remove or keep variables based on their usefulness. Lasso is similar but addresses some issues with the other models, like biased estimates or overly complex models, by setting some parameters to zero based on a specific criterion.

Preferred method:

- Based on the given results, it can be interpreted that the Lasso model produces models with fewer predictors that are easier to understand and use, as some of the coefficients in Lasso are close to zero. On the other hand, none of the coefficients in Ridge are close to zero. However, it's important to note that both models have similar RMSE values, so the choice between Lasso and Ridge may depend on the specific goals and

preferences of the analysis. Therefore, it's difficult to say whether Lasso is more preferable than Ridge in general.

Conclusion:

The study compared the prediction accuracy and the interpretability of coefficients between the Lasso and Ridge models. The Lasso model was found to be better than the Ridge model in terms of producing a simpler and easier to understand model, while both models had similar prediction accuracy. However, it is important to consider the specific context and objectives of the analysis and the balance between interpretability and prediction accuracy when choosing between the two models.

References:

Oleszak, M. (2019, November 12). *Regularization in R Tutorial: Ridge, Lasso and Elastic Net*. https://www.datacamp.com/tutorial/tutorial-ridge-lasso-elastic-net

Stepwise Regression Essentials in R - Articles - STHDA. (2018, November 3). http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/