STA442 Assignment 2

Alexander Tran

3/10/2023

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.0     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.1.8
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-6
```

Question 1

a)

```
set.seed(1006314089)

# Training Set
X_train <- matrix(nrow = 100, ncol = 50)
for (i in 1:50) {
  X_train[,i] <- rnorm(100, 0, 1)
}

epsilon_train <- rnorm(100, 0, 1)

B <- matrix(nrow = 50, ncol = 1)
for (i in 1:20) {
  B[i] <- runif(1, 0.5, 1.5)
}
for (i in 21:50) {
```

```r
  B[i] <- runif(1, 0.2, 0.4)
}

Y_train <- matrix(nrow = 100, ncol = 1)
for (i in 1:100) {
  Y_train[i] <- sum(B*X_train[i,]) + epsilon_train[i]
}

# Test Set
X_test <- matrix(nrow = 1000, ncol = 50)
for (i in 1:50) {
  X_test[,i] <- rnorm(1000, 0, 1)
}

epsilon_test <- rnorm(1000, 0, 1)

Y_test <- matrix(nrow = 1000, ncol = 1)
for (i in 1:1000) {
  Y_test[i] <- sum(B*X_test[i,]) + epsilon_test[i]
}
```

b)

```r
Y_train_df <- as.data.frame(Y_train) %>% rename(Y = V1)
X_train_df <- as.data.frame(X_train)
Y_test_df <- as.data.frame(Y_test)
X_test_df <- as.data.frame(X_test)

train <- cbind(Y_train_df, X_train_df)
lin_reg <- lm(Y ~ ., data = train)

error_lin <- sum((Y_test - predict(lin_reg, newdata = as.data.frame(X_test_df)))**2)/1000
```

The prediction error, calculated as $\frac{1}{1000}\sum_{i=1}^{1000}(y_i - \hat{y_i})^2$ is equal to 2.428.

c)

```r
set.seed(1006314089)

ridge_cv <- cv.glmnet(x = X_train, y = Y_train, alpha = 0)
ridge_reg <- glmnet(x = X_train, y = Y_train, alpha = 0, lambda = ridge_cv$lambda.1se)

error_ridge <- sum((Y_test - predict(ridge_reg, newx = X_test))**2)/1000
```

The prediction error, calculated as $\frac{1}{1000}\sum_{i=1}^{1000}(y_i - \hat{y_i})^2$ is equal to 2.401.

d)

```r
set.seed(1006314089)

lasso_cv <- cv.glmnet(x = X_train, y = Y_train, alpha = 1)
lasso_reg <- glmnet(x = X_train, y = Y_train, alpha = 1, lambda = lasso_cv$lambda.1se)

error_lasso <- sum((Y_test - predict(lasso_reg, newx = X_test))**2)/1000
```

The prediction error, calculated as $\frac{1}{1000}\sum_{i=1}^{1000}(y_i - \hat{y_i})^2$ is equal to 2.337.

e) The LASSO method used in (d) provides the lowest prediction error on the test set. This is because LASSO optimizes driving small weight estimates to 0 in addition to driving down large weights. This results in a model with less flexibility and greater generalizability, compared to simple linear or ridge regression, given that we already expect overfitting and large variance due to a small training set size relative to the number of predictors.

f

```
set.seed(1006314089)

# Training Set
X_train2 <- matrix(nrow = 10000, ncol = 50)
for (i in 1:50) {
  X_train2[,i] <- rnorm(10000, 0, 1)
}

epsilon_train2 <- rnorm(10000, 0, 1)

B <- matrix(nrow = 50, ncol = 1)
for (i in 1:20) {
  B[i] <- runif(1, 0.5, 1.5)
}
for (i in 21:50) {
  B[i] <- runif(1, 0.2, 0.4)
}

Y_train2 <- matrix(nrow = 10000, ncol = 1)
for (i in 1:10000) {
  Y_train2[i] <- sum(B*X_train2[i,]) + epsilon_train2[i]
}

# Test Set
X_test <- matrix(nrow = 1000, ncol = 50)
for (i in 1:50) {
  X_test[,i] <- rnorm(1000, 0, 1)
}

epsilon_test <- rnorm(1000, 0, 1)

Y_test <- matrix(nrow = 1000, ncol = 1)
for (i in 1:1000) {
  Y_test[i] <- sum(B*X_test[i,]) + epsilon_test[i]
}

# Linear Regression
Y_train2_df <- as.data.frame(Y_train2) %>% rename(Y = V1)
X_train2_df <- as.data.frame(X_train2)
Y_test_df <- as.data.frame(Y_test)
X_test_df <- as.data.frame(X_test)

train2 <- cbind(Y_train2_df, X_train2_df)
lin_reg2 <- lm(Y ~ ., data = train2)
error_lin2 <- sum((Y_test - predict(lin_reg2, newdata = as.data.frame(X_test_df)))**2)/1000

# Ridge Regression
```

```r
ridge_cv2 <- cv.glmnet(x = X_train2, y = Y_train2, alpha = 0)
ridge_reg2 <- glmnet(x = X_train2, y = Y_train2, alpha = 0, lambda = ridge_cv2$lambda.1se)
error_ridge2 <- sum((Y_test - predict(ridge_reg2, newx = X_test))**2)/1000

# LASSO Regression
lasso_cv2 <- cv.glmnet(x = X_train2, y = Y_train2, alpha = 1)
lasso_reg2 <- glmnet(x = X_train2, y = Y_train2, alpha = 1, lambda = lasso_cv2$lambda.1se)
error_lasso2 <- sum((Y_test - predict(lasso_reg2, newx = X_test))**2)/1000
```

The prediction error of the linear, ridge, and LASSO regressions are now 0.900, 0.955, and 0.925, respectively. It can be seen that increasing the training set size from 100 to 10000 significantly decreased prediction error for all methods, which implies that there was significant overfitting when fitting on the smaller training set. Of the three, the prediction error of the linear regression is now the lowest, having the greatest decrease after increasing the training set size. This is likely due to the fact that the training data was all simulated from $N(0,1)$, and after increasing the training set size, the sample distribution of X_i became closer to $N(0,1)$ and the least square estimates were unbiases with low variance. In addition, it became less likely that there were any significantly small or large weight estimates, such that the penalization from LASSO and ridge regression were no longer necessary and only introduced more bias to increase prediction error.

Question 2

```
## If the package is not installed then use ##
## install.packages('NHANES') And install.packages('tidyverse')
library(tidyverse)
library(NHANES)
small.nhanes <- na.omit(NHANES[NHANES$SurveyYr=="2011_12"
& NHANES$Age > 17,c(1,3,4,8:11,13,25,61)])
small.nhanes <- small.nhanes %>%
group_by(ID) %>% filter(row_number()==1)
```

a)

```
set.seed(1006314089)

sample <- small.nhanes[sample(nrow(small.nhanes), 500),]

logit_mod <- glm(SmokeNow ~ . - ID, family = binomial, data = sample)
summary(logit_mod)
```

```
##
## Call:
## glm(formula = SmokeNow ~ . - ID, family = binomial, data = sample)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0517  -0.9172  -0.5576   0.9972   2.1789
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                1.967351   1.297390   1.516   0.1294
## Gendermale                 0.313115   0.226002   1.385   0.1659
## Age                       -0.035379   0.008135  -4.349 1.37e-05 ***
## Race3Black                 0.485549   0.572987   0.847   0.3968
## Race3Hispanic             -0.129257   0.640748  -0.202   0.8401
## Race3Mexican              -0.176552   0.632681  -0.279   0.7802
## Race3White                -0.015883   0.488082  -0.033   0.9740
## Race3Other                 1.044106   0.748109   1.396   0.1628
## Education9 - 11th Grade    0.820900   0.467735   1.755   0.0793 .
## EducationHigh School       0.531622   0.438926   1.211   0.2258
## EducationSome College      0.414193   0.440059   0.941   0.3466
## EducationCollege Grad     -0.306669   0.487213  -0.629   0.5291
## MaritalStatusLivePartner   0.731067   0.435643   1.678   0.0933 .
## MaritalStatusMarried      -0.166515   0.337609  -0.493   0.6219
## MaritalStatusNeverMarried  0.053260   0.402631   0.132   0.8948
## MaritalStatusSeparated     1.146655   0.721937   1.588   0.1122
## MaritalStatusWidowed       0.108939   0.501493   0.217   0.8280
## HHIncome 5000-9999         0.162164   0.817472   0.198   0.8428
## HHIncome10000-14999       -0.160948   0.704329  -0.229   0.8192
## HHIncome15000-19999       -0.405803   0.727556  -0.558   0.5770
## HHIncome20000-24999       -0.158961   0.728112  -0.218   0.8272
## HHIncome25000-34999       -0.728362   0.716537  -1.017   0.3094
## HHIncome35000-44999       -0.346336   0.747481  -0.463   0.6431
## HHIncome45000-54999       -0.662395   0.791743  -0.837   0.4028
## HHIncome55000-64999        0.405484   0.863994   0.469   0.6388
```

```
## HHIncome65000-74999       -0.714962   0.885037  -0.808   0.4192
## HHIncome75000-99999       -0.873012   0.898731  -0.971   0.3314
## HHIncomemore 99999        -0.245078   0.891151  -0.275   0.7833
## Poverty                   -0.071541   0.141471  -0.506   0.6131
## BPSysAve                  -0.004401   0.007205  -0.611   0.5414
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 685.44  on 499  degrees of freedom
## Residual deviance: 576.16  on 470  degrees of freedom
## AIC: 636.16
##
## Number of Fisher Scoring iterations: 4
```

Based on the output above, most predictors are not statistically significant. Only variables *Age*, *Education = 9 - 11th Grade*, and *MaritalStatus = LivePartner* significantly explain the outcome variable, with the latter two only being significant at the 10% level. With this in mind, we should consider removing some predictors from the model.

Regarding interpretation of coefficients, we see that the odds of smoking is multiplied by $e^{-0.035379} = 0.965$ per year of *Age* (a decrease per year), while the odds are higher for *Education = 9 - 11th Grade* and *MaritalStatus = LivePartner* by $e^{0.820900} = 2.273$ and $e^{0.731067} = 2.077$ times compared to their respective baseline groups, *Education = 8th Grade* and *MaritalStatus = Divorced*.

b)

```
set.seed(1006314089)

aic_step_model <- step(logit_mod, trace = 0, k = 2)
aic_selected <- attr(terms(aic_step_model), "term.labels")
aic_selected
```

```
## [1] "Age"       "Education" "Poverty"
```

```
bic_step_model <- step(logit_mod, trace = 0, k = log(500))
bic_selected <- attr(terms(bic_step_model), "term.labels")
bic_selected
```

```
## [1] "Age"     "Poverty"
```

```
elastnet_reg <- cv.glmnet(x = model.matrix(logit_mod), y = as.numeric(sample$SmokeNow) - 1,
                          family = "binomial", alpha = 0.5)
elastnet_selected <- coef(elastnet_reg, s = elastnet_reg$lambda.1se)
elastnet_selected
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)                 1.110586856
## (Intercept)                 .
## Gendermale                  .
## Age                        -0.024068995
## Race3Black                  .
## Race3Hispanic               .
## Race3Mexican                .
## Race3White                 -0.006834978
## Race3Other                  .
```

```
## Education9 - 11th Grade     0.123434550
## EducationHigh School        .
## EducationSome College       .
## EducationCollege Grad      -0.264108139
## MaritalStatusLivePartner    0.202853263
## MaritalStatusMarried       -0.041811348
## MaritalStatusNeverMarried   .
## MaritalStatusSeparated      .
## MaritalStatusWidowed        .
## HHIncome 5000-9999          .
## HHIncome10000-14999         .
## HHIncome15000-19999         .
## HHIncome20000-24999         .
## HHIncome25000-34999         .
## HHIncome35000-44999         .
## HHIncome45000-54999         .
## HHIncome55000-64999         .
## HHIncome65000-74999         .
## HHIncome75000-99999         .
## HHIncomemore 99999          .
## Poverty                    -0.046503019
## BPSysAve                    .
```

```
elastnet_reg2 <- cv.glmnet(x = model.matrix(logit_mod), y = as.numeric(sample$SmokeNow) - 1,
                           family = "binomial", alpha = 1)
elastnet_selected2 <- coef(elastnet_reg, s = elastnet_reg2$lambda.1se)
elastnet_selected2
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)                 1.20473497
## (Intercept)                 .
## Gendermale                  .
## Age                        -0.02528439
## Race3Black                  .
## Race3Hispanic               .
## Race3Mexican                .
## Race3White                 -0.02858768
## Race3Other                  0.03679893
## Education9 - 11th Grade     0.15514222
## EducationHigh School        .
## EducationSome College       .
## EducationCollege Grad      -0.30196600
## MaritalStatusLivePartner    0.23723479
## MaritalStatusMarried       -0.06249439
## MaritalStatusNeverMarried   .
## MaritalStatusSeparated      0.01298444
## MaritalStatusWidowed        .
## HHIncome 5000-9999          .
## HHIncome10000-14999         .
## HHIncome15000-19999         .
## HHIncome20000-24999         .
## HHIncome25000-34999         .
## HHIncome35000-44999         .
## HHIncome45000-54999         .
```

```
## HHIncome55000-64999         .
## HHIncome65000-74999         .
## HHIncome75000-99999         .
## HHIncomemore 99999          .
## Poverty                   -0.05143559
## BPSysAve                    .
```

The different selection methods do not all select the same model. While there are some variables selected in all models such as *Age* and *Poverty*, there are others that are only in some models. This is because they all use different criteria to select the best model. AIC and BIC are metrics that penalize a fitted model by a function of the number of parameters and number of observations, while elastic-net penalizes the model in the form a cost function that must be optimized for in regression parameter estimates. While all methods can reduce overfitting, they do so in different ways that may select different models. Also note that elastic net selects the same model for $\alpha = 0.5$ and $\alpha = 1$.