Lab 3

Elke Windschitll

2023-01-29

Lab 3: Predicting the age of abalone

Abalones are marine snails. Their flesh is widely considered to be a desirable food, and is consumed raw or cooked by a variety of cultures. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age.

The data set provided includes variables related to the sex, physical dimensions of the shell, and various weight measurements, along with the number of rings in the shell. Number of rings is the stand-in here for age.

Data Exploration

Pull the abalone data from Github and take a look at it.

```
abdat<- dat <- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/abalone-data.csv") %>% select(-...1) # Select to remove the index column (should not be included )
```

```
## New names:
## Rows: 4177 Columns: 10
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (1): Sex dbl (9): ...1, Length, Diameter, Height, Whole_weight, Shucked_weight,
## Visce...
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

```
glimpse(abdat)
```

```
## Rows: 4,177
## Columns: 9
## $ Sex            <chr> "M", "M", "F", "M", "I", "I", "F", "F", "M", "F", "F", …
## $ Length         <dbl> 0.455, 0.350, 0.530, 0.440, 0.330, 0.425, 0.530, 0.545,…
## $ Diameter       <dbl> 0.365, 0.265, 0.420, 0.365, 0.255, 0.300, 0.415, 0.425,…
## $ Height         <dbl> 0.095, 0.090, 0.135, 0.125, 0.080, 0.095, 0.150, 0.125,…
## $ Whole_weight   <dbl> 0.5140, 0.2255, 0.6770, 0.5160, 0.2050, 0.3515, 0.7775,…
## $ Shucked_weight <dbl> 0.2245, 0.0995, 0.2565, 0.2155, 0.0895, 0.1410, 0.2370,…
## $ Viscera_weight <dbl> 0.1010, 0.0485, 0.1415, 0.1140, 0.0395, 0.0775, 0.1415,…
## $ Shell_weight   <dbl> 0.150, 0.070, 0.210, 0.155, 0.055, 0.120, 0.330, 0.260,…
## $ Rings          <dbl> 15, 7, 9, 10, 7, 8, 20, 16, 9, 19, 14, 10, 11, 10, 10, …
```

Data Splitting

- ***Question 1***. Split the data into training and test sets. Use a 70/30 training/test split.

```
# Stratified sampling with the rsample package
set.seed(123) #set a seed for reproducibility
split <- initial_split(data = abdat,
                       prop = .7, # 70/30 split
                       strata = "Rings")
split
```

```
## <Training/Testing/Total>
## <2922/1255/4177>
```

```
abalone_train <- training(split)
abalone_test  <- testing(split)
```

We'll follow our text book's lead and use the caret package in our approach to this task. We will use the glmnet package in order to perform ridge regression and the lasso. The main function in this package is glmnet(), which can be used to fit ridge regression models, lasso models, and more. In particular, we must pass in an x matrix of predictors as well as a y outcome vector , and we do not use the y~x syntax.

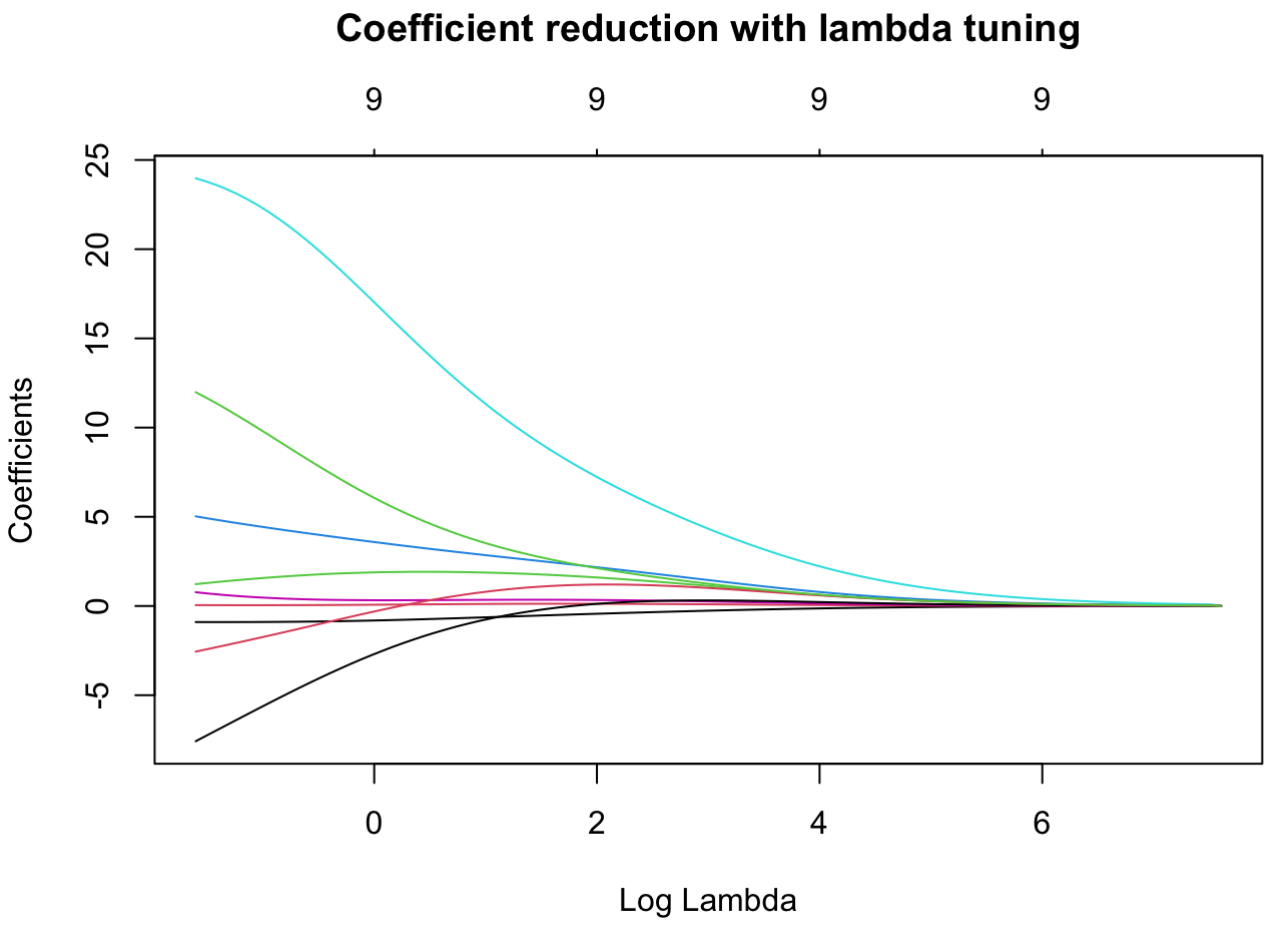
Fit a ridge regression model

- ***Question 2***. Use the model.matrix() function to create a predictor matrix, x, and assign the Rings variable to an outcome vector, y.

```
#Create training feature matrices using model.matrix() (auto encoding of categorical var
iables)

X <- model.matrix(Rings ~ ., abalone_train)[,-1] # Last bit removes intercept column

Y <- (abalone_train$Rings)
```

- ***Question 3***. Fit a ridge model (controlled by the alpha parameter) using the glmnet() function. Make a plot showing how the estimated coefficients change with lambda. (Hint: You can call plot() directly on the glmnet() objects).

```
#fit a ridge model, passing X,Y,alpha to glmnet()
abalone_ridge <- glmnet(x = X,
                y = Y,
                alpha = 0)

#plot() the glmnet model object
plot(abalone_ridge, xvar = "lambda")
title("Coefficient reduction with lambda tuning", line = 3)
```

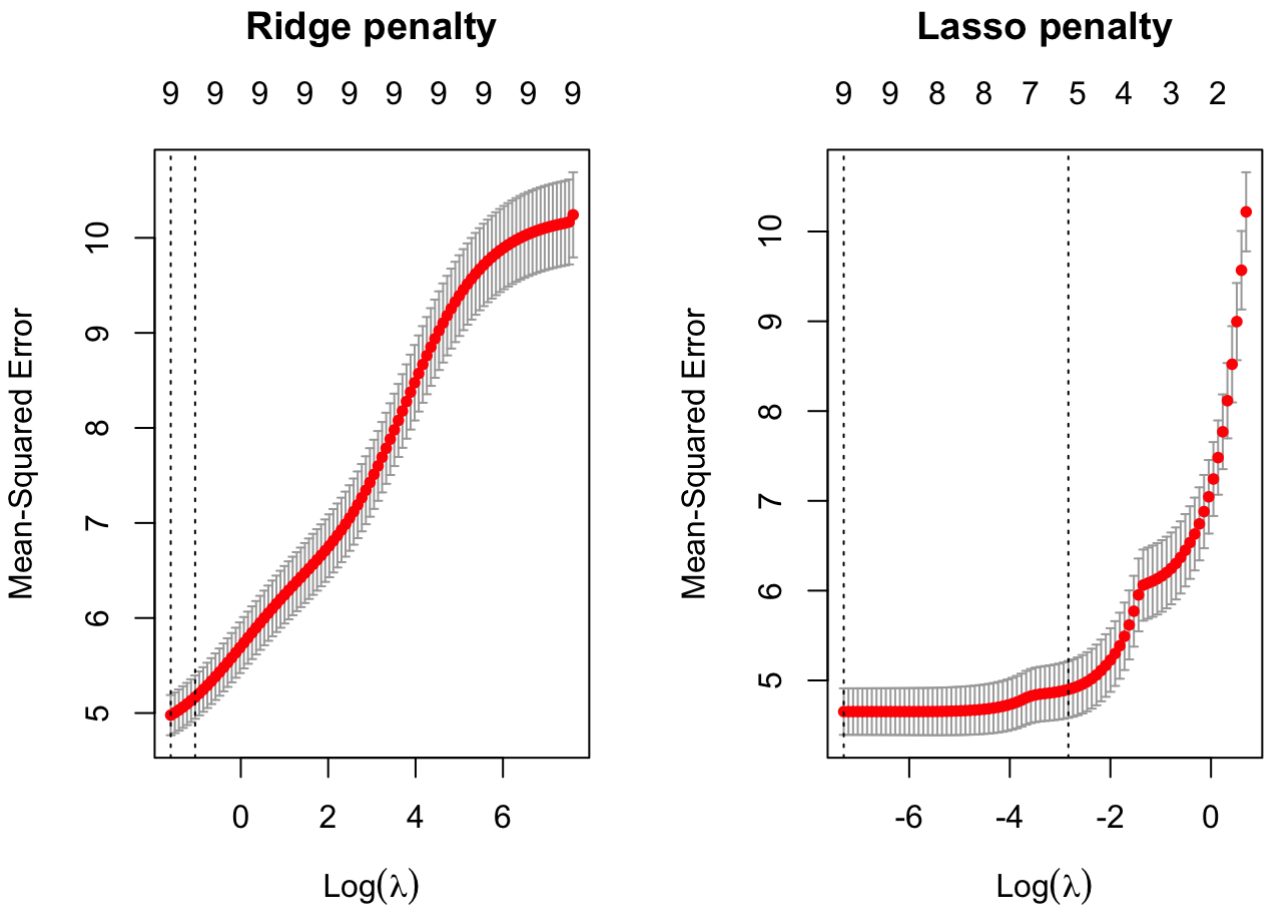


Using *k*-fold cross validation resampling and tuning our models

In lecture we learned about two methods of estimating our model's generalization error by resampling, cross validation and bootstrapping. We'll use the *k*-fold cross validation method in this lab. Recall that lambda is a tuning parameter that helps keep our model from over-fitting to the training data. Tuning is the process of finding the optima value of lamba.

- ***Question 4***. This time fit a ridge regression model and a lasso model, both with using cross validation. The glmnet package kindly provides a cv.glmnet() function to do this (similar to the glmnet() function that we just used). Use the alpha argument to control which type of model you are running. Plot the results.

```
# Apply CV ridge regression to abaolone data. Same arguments as before to glmnet()
abalone_cv_ridge <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 0
)

# Apply CV lasso regression to abalone data
abalone_cv_lasso <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 1
)

# plot results
par(mfrow = c(1, 2))
plot(abalone_cv_ridge, main = "Ridge penalty\n\n")
plot(abalone_cv_lasso, main = "Lasso penalty\n\n")
```



- ***Question 5***. Interpret the graphs. What is being show on the axes here? How does the performance of the models change with the value of lambda?

  Here, on the x axis we have the increasing values of log lamba for tuning the model. On the y axis we have the Mean Squared Error values. On the top axis we have the number of features retained in the model. The first dotted vertical line in each graph represents the minimum Mean Squared Error. The second dotted line indicates one standard deviation from the minimum MSE. For the ridge penalty,

increasing the lambda increases the MSE fairly quickly (and thus decreases the performance of the model) and might not be the best method for tuning here as we want to have a low MSE. In the lasso penalty, the MSE remains the same just under 5 for a while when increasing lambda and then increases dramatically after one SD. The lasso penatly retains 5 variables at one SD from the minimum MSE and the rigde penatly retains all 9 variables.

- ***Question 6***. Inspect the ridge model object you created with cv.glmnet(). The $cvm column shows the MSEs for each cv fold. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?

```
# View ridge model summary
summary(abalone_cv_ridge)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         4    -none- call
## name         1    -none- character
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
# Find the minimun mse
min_mse_r <- min(abalone_cv_ridge$cvm)
# Find the lambda of the minimum mse
min_lambda_r <- abalone_cv_ridge$lambda.min
# Answer Q
print(paste("The minimun MSE in the ridge model is", min_mse_r ,"at a value of", mi
n_lambda_r, "for lambda."))
```

```
## [1] "The minimun MSE in the ridge model is 4.978248482699 at a value of 0.201214
210150836 for lambda."
```

- ***Question 7***. Do the same for the lasso model. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?

```
# View lasso model summary
summary(abalone_cv_lasso)
```

```
##            Length Class  Mode
## lambda     87     -none- numeric
## cvm        87     -none- numeric
## cvsd       87     -none- numeric
## cvup       87     -none- numeric
## cvlo       87     -none- numeric
## nzero      87     -none- numeric
## call        4     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
## index       2     -none- numeric
```

```
# Find the minimun mse
min_mse_l <- min(abalone_cv_lasso$cvm)
# Find the lambda of the minimum mse
min_lambda_l <- abalone_cv_lasso$lambda.min
# Answer Q
print(paste("The minimun MSE in the ridge model is", min_mse_l ,"at a value of", mi
n_lambda_l, "for lambda."))
```

```
## [1] "The minimun MSE in the ridge model is 4.65276174691365 at a value of 0.0006
74390080148105 for lambda."
```

Data scientists often use the “one-standard-error” rule when tuning lambda to select the best model. This rule tells us to pick the most parsimonious model (fewest number of predictors) while still remaining within one standard error of the overall minimum cross validation error. The cv.glmnet() model object has a column that automatically finds the value of lambda associated with the model that produces an MSE that is one standard error from the MSE minimum ($lambda.1se).

- ***Question 8.*** Find the number of predictors associated with this model (hint: the $nzero is the # of predictors column).

```
# Find number of predictors for ridge model at 1 SD
ridge_predictors <- abalone_cv_ridge$nzero[abalone_cv_ridge$lambda == abalone_cv_ridge$l
ambda.1se]

print(paste("The number of predictors associated with the ridge model within one standar
d deviation of the minimum MSE is", ridge_predictors))
```

```
## [1] "The number of predictors associated with the ridge model within one standard dev
iation of the minimum MSE is 9"
```

```
# Find number of predictors for lasso model at 1 SD
lasso_predictors <- abalone_cv_lasso$nzero[abalone_cv_lasso$lambda == abalone_cv_lasso$l
ambda.1se]

print(paste("The number of predictors associated with the lasso model within one standar
d deviation of the minimum MSE is", lasso_predictors))
```

```
## [1] "The number of predictors associated with the lasso model within one standard dev
iation of the minimum MSE is 5"
```

- **Question 9.** Which regularized regression worked better for this task, ridge or lasso? Explain your answer.

  The Lasso regularized regression worked better here than the ridge penalty for a couple of reasons. First, the miniminum MSE is slightly smaller in the lasso penatly indicating better model performance. Second, the Lasso regularized regression performs feature selection and brings the number of features down from 9 to 5. The ridge penalty retains all 9 features, some of which likely do not need to be in the model. Therefore, for this task the lasso worked best.