USASK Data Science Bootcamp T6: Statistical Machine Learning Ridge, Lasso, Elastic Net R Tutorial

TA: Kyle Gardiner Tutors: Lina Li and Jing Wang

June 13, 2023. (3:50pm-4:30pm)

This document will outline how to employ Ridge regression, Lasso Regression, and Elastic Net.

First we will load in the necessary packages for this tutorial.

```
rm(list=ls(all=TRUE)) # removes objects from the environment to start fresh

library(ISLR2) # contains the dataset "Auto" used in this tutorial
library(glmnet) # used for ridge, lasso, and elastic net
library(caret) # used for elastic net
```

We will be using the same dataset for all 3 methods so lets load in the data and split it before going into each of the methods.

Loading the data

```
data(Auto) # adds the "Auto" dataset to the environment

?Auto # Prints dataframe information in the 'help' console

str(Auto) # Displays the structure of the "Auto" dataset
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : int  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
##  - attr(*, "na.action")= 'omit' Named int [1:5] 33 127 331 337 355
##   ..- attr(*, "names")= chr [1:5] "33" "127" "331" "337" ...
```

```
dim(Auto) # Displays the dimensions of the "Auto" dataset (rows x columns)
```

```
## [1] 392   9
```

```
sum(is.na(Auto)) # checking for missing values that might need to be dealt with before data analysis
```

```
## [1] 0
```

Formatting data
Before we start building the model, we need to do some data formatting that will help us in the later sections.

```
# need to convert categorical variables to factors
Auto$cylinders<-as.factor(Auto$cylinders)
Auto$year<-as.factor(Auto$year)
Auto$origin<-as.factor(Auto$origin)

# using 'model.matrix()' to create a design matrix
xx <- model.matrix(mpg ~., Auto[,-9]) # using all the predictors except for 'name'
yy <- Auto$mpg # defining the outcome variable; 'mpg'

# specifying grid to be used later in the functions
grid<-10^seq(10, -2, length=100)
```

Splitting the Data

```
set.seed(1) # set the seed for reproducibility
id<-sample(nrow(Auto),round(nrow(Auto)*0.80)) # Used to identify indexes that fall in train or test
xx.train<-xx[id,]# 'xx.train' contains 80% of the original data for predictors
xx.test<-xx[-id,]# 'xx.test' contains the remaining 20% of the original data for predictors
yy.train<-yy[id]# 'yy.train' contains 80% of the original data for outcome
yy.test<-yy[-id]# 'yy.test' contains the remaining 20% of the original data for outcome
```

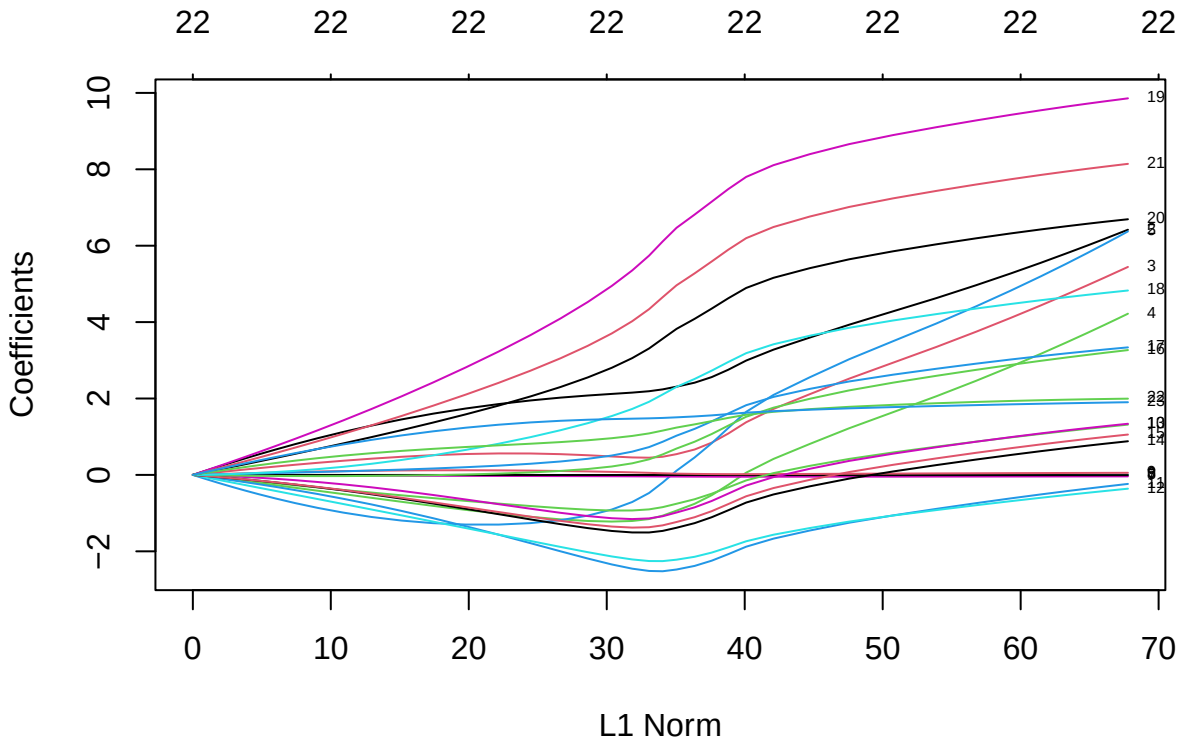
Once we have finished loading, formatting, and splitting the data, we can start with ridge regression.

Ridge Regression

(a) Ridge Regression Model Building

```
# To fit the ridge regression model, we use the 'glmnet()' function from the glmnet package
ridge.mod <- glmnet(xx.train, yy.train, alpha = 0, lambda = grid)
# IMPORTANT: specifying 'alpha' is necessary to apply the proper regularization penalty to the model
# 'alpha=0' specifies ridge regression

plot(ridge.mod, label = T)
```

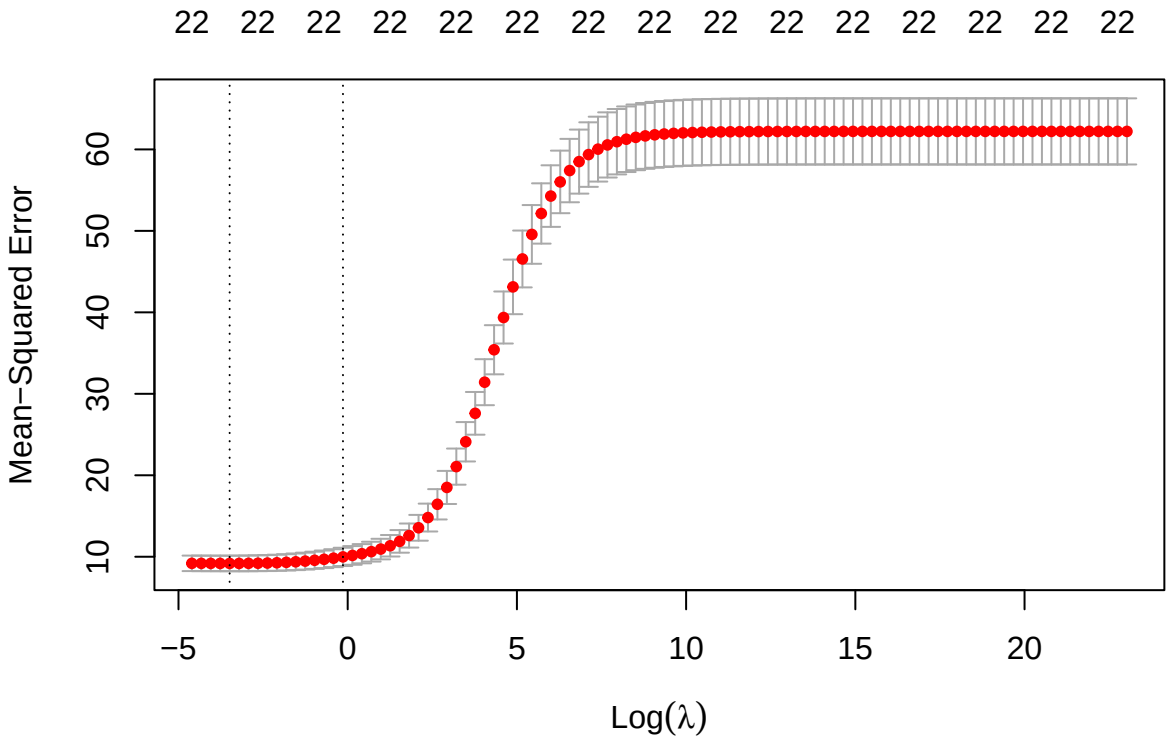



```
# plot to visualize how the coefficient estimates change as the shrinkage penalty changes
```

(*b*) **Cross Validation to identify best λ that reduces error**

```
ridge.cv <- cv.glmnet(xx.train, yy.train, alpha = 0, lambda = grid, nfolds=10)
# 10 fold cross validation

plot(ridge.cv) # plot to visualize how MSE changes as lambda changes
```



(*c*) **Assessing model accuracy**

```
bestlam.ridge <- ridge.cv$lambda.1se # can either use 'lambda.min' or 'lambda.1se'

ridge.pred=predict(ridge.mod, s=bestlam.ridge, newx=xx.test)
# predicting the values using the non-cv ridge model and the best lambda from cross validation

mean((ridge.pred-yy.test)^2) # calculating mean squared error for this model
```

```
## [1] 7.698884
```

We will compare the accuracy of these models at the very end.

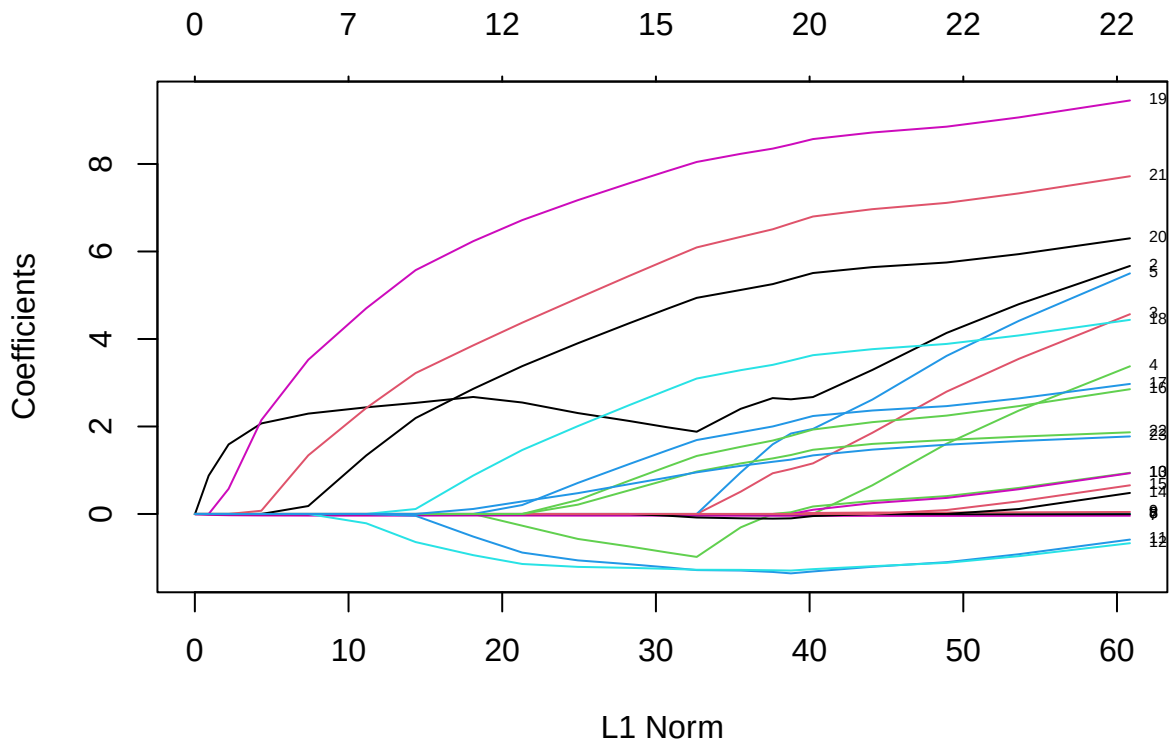
Lasso Regression

(*a*) **Lasso Regression Model Building**

```
# To fit the lasso regression model, we use the 'glmnet()' function from the glmnet package
lasso.mod <- glmnet(xx.train, yy.train, alpha = 1, lambda = grid)
# IMPORTANT: specifying 'alpha' is necessary to apply the proper regularization penality to the model
```

```
                                        # 'alpha=1' specifies lasso regression
plot(lasso.mod, label = T)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

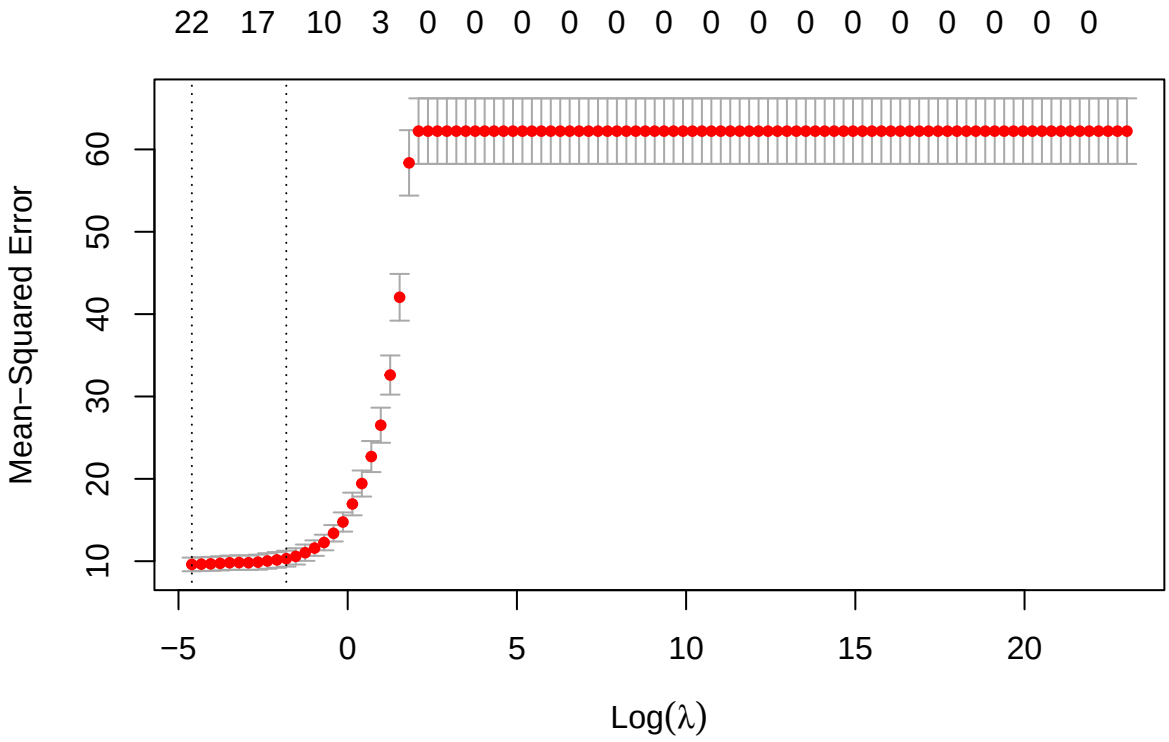



```
# plot to visualize how the coefficient estimates change as the shrinkage penalty changes
```

(*b*) **Cross Validation to identify best λ that reduces error**

```
lasso.cv <- cv.glmnet(xx.train, yy.train, alpha = 1, lambda = grid, nfolds=10)
# 10 fold cross validation

plot(lasso.cv)# plot to visualize how MSE changes as lambda changes
```



(*c*) **Assessing model accuracy**

```
bestlam.lasso <- lasso.cv$lambda.1se # can either use 'lambda.min' or 'lambda.1se'

lasso.pred=predict(lasso.mod, s=bestlam.lasso, newx=xx.test)
# predicting the values using the non-cv lasso model and the best lambda from cross validation

mean((lasso.pred-yy.test)^2) # calculating mean squared error for this model
```

```
## [1] 7.476549
```

Again, we will compare the performance of all the model at the end.

Elastic Net

Elastic Net combines Ridge and Lasso regression.

(*a*) **Cross Validation**
In this section, we use cross validation to find the most appropriate model for our data. Cross validation allows us to select the best α and λ to reduce the error in the model

```
# trainControl allows us to define control parameters to use in the following 'train()' function
cv_10 = trainControl(method = "cv", number = 10) # here we specify 10 fold cross validation


set.seed(1) # set seed for reproducibility

# Using the 'train()' function, we train a cross validation elastic net
elastic.net.cv = train(
  mpg ~.,             # formula to predict mpg
  data = Auto[id,-9], # using the training data with all the predictor, except for 'name'
  method = "glmnet",  # 'method="glmnet"' specifies using elastic net
  trControl = cv_10)  # using the control parameters we specified earlier
```

(*b*) **Determining Best α and λ**
After fitting the cross validation elastic net, we can extract the values of α and λ that correspond to the smallest Root Mean Squared Error (RMSE).

```
elastic.net.cv # prints table for each combination of alpha and lambda and corresponding RMSE
```

```
## glmnet
##
## 314 samples
##   7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 284, 283, 283, 281, 283, 282, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      RMSE      Rsquared   MAE
##   0.10   0.01294453  3.042737  0.8524046  2.308646
##   0.10   0.12944527  3.067450  0.8506485  2.313612
##   0.10   1.29445274  3.234601  0.8394891  2.440086
##   0.55   0.01294453  3.046206  0.8520572  2.307647
##   0.55   0.12944527  3.087948  0.8493096  2.320911
##   0.55   1.29445274  3.704561  0.8051795  2.803092
##   1.00   0.01294453  3.050636  0.8517215  2.308897
##   1.00   0.12944527  3.113617  0.8476793  2.338149
##   1.00   1.29445274  4.163707  0.7677863  3.142528
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.01294453.
```

```
# extracting optimal alpha and lambda
myid <- order(elastic.net.cv$results$RMSE)[1] # determines the index that has the smallest RMSE
```

```
# extracts the alpha that corresponds to the smallest RMSE
mybest.alpha <- elastic.net.cv$results$alpha[myid]

# extracts the lambda that corresponds to the smallest RMSE
mybest.lambda <- elastic.net.cv$results$lambda[myid]
```

(c) Fitting Elastic Net with optimal α and λ

```
# to do this, use the 'glmnet()' function
elastic.net.model<- glmnet(xx.train, yy.train, alpha = mybest.alpha, lambda = mybest.lambda)
# using 'mybest.alpha' and 'mybest.lambda' from cross validation

coef(elastic.net.model)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  30.805500302
## (Intercept)   .
## cylinders4    6.247375595
## cylinders5    5.235532941
## cylinders6    3.995908710
## cylinders8    6.102187506
## displacement  0.007275972
## horsepower   -0.038544698
## weight       -0.004997299
## acceleration  0.053986586
## year71        1.255078477
## year72       -0.294989347
## year73       -0.412742892
## year74        1.276020491
## year75        0.816982936
## year76        0.994356936
## year77        3.200435545
## year78        3.281689080
## year79        4.767955393
## year80        9.788016445
## year81        6.630889068
## year82        8.070358153
## origin2       1.978721333
## origin3       1.885491137
```

```
# prints the coefficient estimates for the variables used in the elastic net model
```

(*d*) **Assessing model accuracy**
Once we have built the elastic net model, we can assess its accuracy.

```
# Again, we predict the test outcomes using the built elastic net model
elastic.net.pred<-predict(elastic.net.model, s=mybest.lambda, newx=xx.test)
# need to specify 's=mybest.lambda' to ensure we are using the proper lambda to predict

# Now we can find the Mean Squared Error for this model
mean((elastic.net.pred-yy.test)^2)
```

```
## [1] 7.656109
```

Final Comparisons

Lasso MSE: 7.698884
Ridge MSE: 7.476549
Elastic Net MSE: 7.656109

This concludes the tutorial of how to employ ridge regression, lasso regression, and elastic net in R.