HW8

Arjun Goyal, Richard Hardis, Alan Lo, James Trawick

10/9/2019

1. Stepwise Regression

a. Creating the Linear Model

We are using the caret package to train a linear regression model using a variable selection method of stepwise regression with the leapSeq function.

We select our training and test sets randomly, assigning 75% to the training set and 25% to the test set.

```
set.seed(825)
stepLeapFit <- train(Crime ~ ., data = training,
                  method = "leapSeq", tuneGrid = data.frame(nvmax = 1:10),
                  trControl = trainControl(## 10-fold CV
                            method = "repeatedcv",
                            number = 10,
                            ## repeated ten times
                            repeats = 10),
                  verbose=F
                 )
summary(stepLeapFit$finalModel)
```

```
## Subset selection object
## 15 Variables  (and intercept)
##        Forced in Forced out
## M          FALSE      FALSE
## So         FALSE      FALSE
## Ed         FALSE      FALSE
## Po1        FALSE      FALSE
## Po2        FALSE      FALSE
## LF         FALSE      FALSE
## M.F        FALSE      FALSE
## Pop        FALSE      FALSE
## NW         FALSE      FALSE
## U1         FALSE      FALSE
## U2         FALSE      FALSE
## Wealth     FALSE      FALSE
## Ineq       FALSE      FALSE
## Prob       FALSE      FALSE
## Time       FALSE      FALSE
## 1 subsets of each size up to 3
## Selection Algorithm: 'sequential replacement'
##          M   So  Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  Wealth Ineq Prob Time
## 1  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    " "  " "  " "
## 2  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    "*"  " "  " "
## 3  ( 1 ) " " " " " " "*" " " " " " " "*" " " " " " " " "    "*"  " "  " "
```

According to the leapSeq method of doing stepwise regression, the best performing model on the cross-validated training data included three predictor variables: Po1, M.F, and Ineq. We will then run this tuned

model on the test data to determine our prediction accuracy.

```
coef(stepLeapFit$finalModel, stepLeapFit$bestTune[['nvmax']])
```

```
## (Intercept)         Po1         M.F        Ineq
## -4690.54619   121.99529    37.95857    42.59560
```

b. Testing Prediction Accuracy

We run our prediction on the test dataset using the reduced linear model with the three predictor variables identified by the leapSeq method of variable selection.

```
stepLeapTestPrediction <- predict(stepLeapFit, testing[-ncol(testing)], interval = "prediction")
stepLeapTestPrediction
```

```
##         4        10        12        15        33        35        38
## 1611.6120  822.7207  646.6358  872.0659  760.0747  764.2911  776.9884
##        39        40        44        46
##  740.2264 1004.7306  999.8651 1085.0912
```

The Mean Squared Prediction Error for the reduced model is 64083.16.

```
mean((stepLeapTestPrediction-testing$Crime)^2)
```

```
## [1] 64083.16
```

If we build a model using linear regression on all the predicting variables, we can compare our testing accuracy and see if the reduced model's MSPE was less than the full model's MSPE.

```
mean((stepLeapFullPrediction-testing$Crime)^2)
```

```
## [1] 72198.13
```

Because the MSPE of the full model (72198.12) is higher than the MSPE of the reduced model (64083.16), the reduced model had a better prediction accuracy.

We can go one step further by comparing the two models using their AIC values.

```
stepLeapModel <- lm(Crime~Po1+M.F+Ineq, data=training)

exp((AIC(stepLeapModel) - AIC(stepLeapFull))/2)
```

```
## [1] 0.2194559
```

The full model is 21.94% as likely as the reduced model to predict the number of crimes, which gives us evidence to choose the reduced model for any further prediction.

2. Lasso

a. Creating the Linear Model

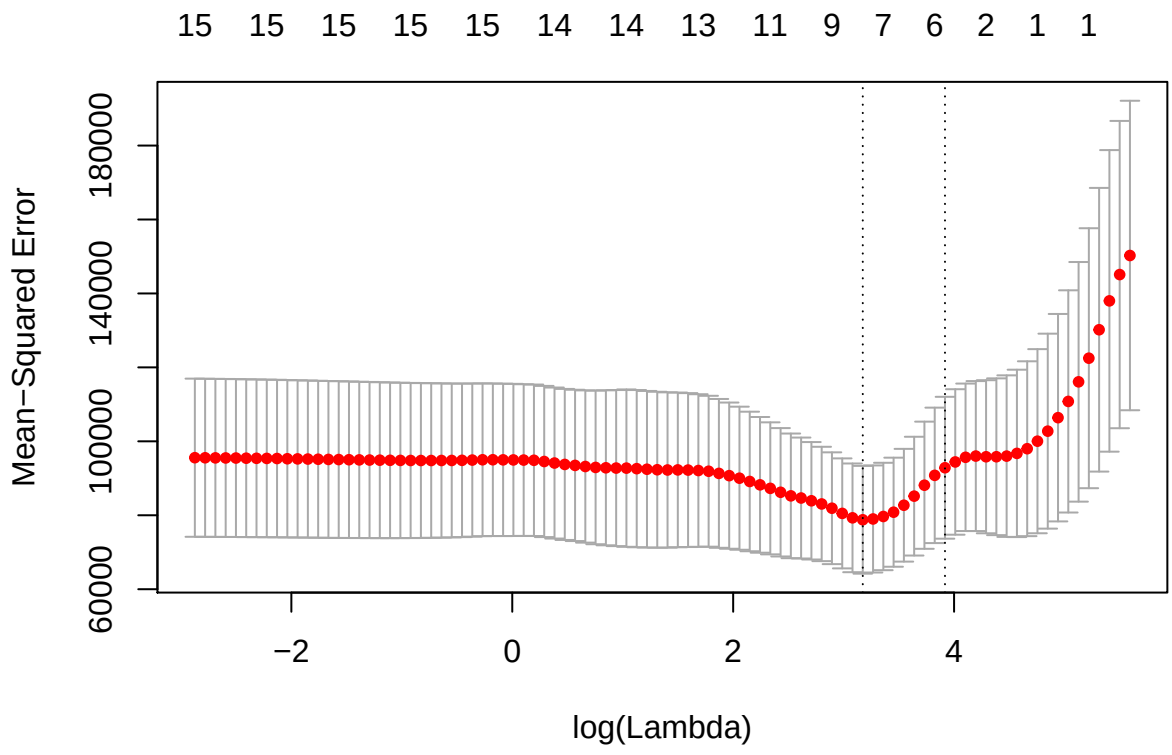
We create a matrix of the training and test data excluding the response variable to be used in the glmnet package for parts 2 and 3, scaling the datasets as well.

Joel's Term: $\lambda == \alpha$: glmnet's Term

Joel's Term: $\tau == \lambda$: glmnet's Term

USING THE MODEL'S TERMS: Running the cv.glmnet function on the training matrix and training response vector builds a linear regression model for an alpha = 1, because this is a LASSO approach, and the λ value that gives the best tradeoff between bias, variance, and the prediction.

```
set.seed(100)
crossvalfit <- cv.glmnet(scaledTraining, trainingresponsevector)
plot(crossvalfit)
```




```
crossvalfit$lambda.min
```

```
## [1] 23.88756
```

```
coef(crossvalfit, s = "lambda.min")
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                      1
## (Intercept) 900.33333
## M            16.47582
## So            .
## Ed            .
## Po1         283.64065
## Po2           .
## LF           17.80771
## M.F          77.23661
## Pop           .
## NW           26.55875
## U1            .
## U2            .
## Wealth        .
## Ineq         85.63996
## Prob        -57.36463
## Time          .
```

b. Testing Prediction Accuracy

The model tells us that a lambda of 23.88 provides us with the optimal model. The kept regression coefficients are M, Po1, LF, M.F, NEW, Ineq, and Prob. Now, we can test this model on our test dataset.

```
LASSOTestPrediction <- predict(crossvalfit, scaledTesting, interval = "prediction", s = "lambda.min")
```

```
mean((LASSOTestPrediction-testingresponsevector)^2)
```

```
## [1] 59458.41
```

The Mean Squared Prediction Error for the reduced model using LASSO variable selection is 59458.41

Compared to the MSPE for the full model (72198.13), the reduced model has a smaller value for its mean squared prediction error, and a smaller MSPE compared to the linear model produced by Stepwise Regression approach to variable selection. By this measure, the LASSO reduced model has a better prediction accuracy than the other two models.

3. Elastic Net

a. Building the Linear Model

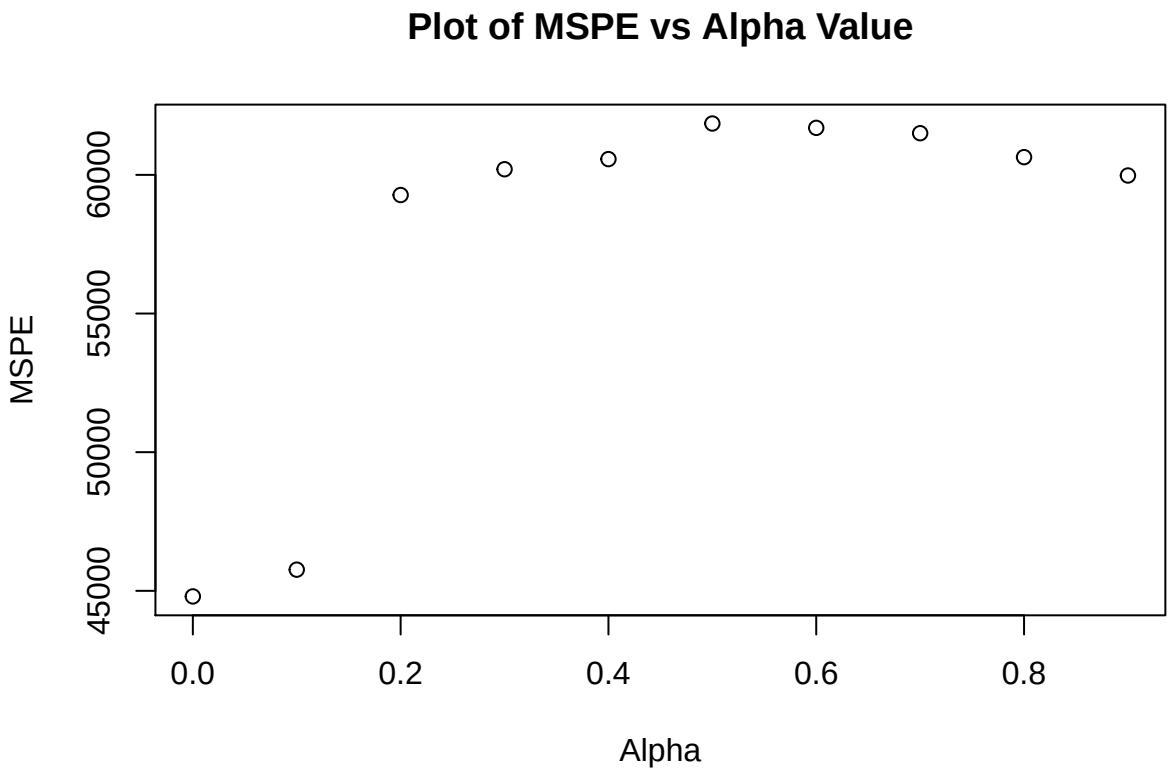
For an Elastic Net model, we will test values of alpha from 0 - 0.9, where an alpha = 0 would be equivalent to using the Ridge Regression approach to model selection, and testing values of alpha between 0 and 0.9 would help us identify a value of alpha that produces the least MSPE for the test response.

```
MSPE <- rep(0, 10)
i = 1
for(alpha in seq(0, 0.9, 0.1)){

set.seed(100)
crossvalfit1 <- cv.glmnet(scaledTraining, trainingresponsevector, alpha = alpha)

ElasticNetTestPrediction <- predict(crossvalfit1, scaledTesting, interval = "prediction", s = "lambda.m
```

```
MSPE[i] = mean((ElasticNetTestPrediction-testingresponsevector)^2)

i = i + 1
}
```

Using the following plot, we can determine that an alpha value of 0, corresponding to the Ridge Regression approach to variable selection, will produce the least amount of MSPE (44799) for our test set.

```
plot(x=seq(0,0.9,0.1), MSPE, main = "Plot of MSPE vs Alpha Value", xlab = "Alpha")
```



An alpha value of 0 produces a model that uses all 15 predictor values, and an alpha value of 0.1 only eliminates the Time predictor. If our goal is to eliminate a few more variables, we can continue with an alpha value of 0.2, which still produces an MSPE of 59274, less than the other approaches detailed above while also eliminating 4 predictor variables.

```
set.seed(100)
crossvalfitEN <- cv.glmnet(scaledTraining, trainingresponsevector, alpha = 0.2)
coef(crossvalfitEN, s = "lambda.min")
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 900.3333333
## M            20.8453619
## So            .
## Ed            0.7674928
```

```
## Po1         148.5310192
## Po2          94.8623043
## LF           33.1817341
## M.F          68.0926557
## Pop           .
## NW           44.5385527
## U1           -2.2523779
## U2           17.9702710
## Wealth        .
## Ineq         48.8674459
## Prob        -61.2046230
## Time          .
```