DSE 6111 - Module 5 - Ch 6 HW

Nathan Monges

2024-08-04

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.2.0 --

## v broom        1.0.5      v recipes      1.0.10
## v dials        1.2.1      v rsample      1.2.1
## v dplyr        1.1.4      v tibble       3.2.1
## v ggplot2      3.5.0      v tidyr        1.3.1
## v infer        1.0.7      v tune         1.2.1
## v modeldata    1.4.0      v workflows    1.1.4
## v parsnip      1.2.1      v workflowsets 1.1.0
## v purrr        1.0.2      v yardstick    1.3.1

## Warning: package 'modeldata' was built under R version 4.3.3

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```r
library(ISLR2)
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.3.3
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
##
##     Boston

## The following object is masked from 'package:dplyr':
```

```
##
##     select
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

Exercise 9

a) Split the data set into a training set and a test set.

```r
set.seed(1)
data_split <- initial_split(College, strata = "Apps", prop = 0.75)

training_set <- training(data_split)
test_set <- testing(data_split)
```

b) Fit a linear model using least squares on the training set, and report the test error obtained.

```r
set.seed(1)

lm_recipe <- recipe(Apps ~ . , data= training_set)

lm_fit <- linear_reg() %>%
  set_engine("lm") %>%
  fit(Apps ~ ., data = training_set)

lm_pred <- lm_fit %>%
  predict(new_data = test_set) %>%
  bind_cols(test_set)

lm_pred %>%
  metrics(truth = Apps, estimate = .pred) %>%
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1105.
```

c) Fit a ridge regression model on the training set, with lambda chosen by cross-validation. Report the test error obtained.

```r
x_train <- model.matrix(Apps ~ . , training_set)[, -1]
y_train <- training_set$Apps

x_test <- model.matrix(Apps ~. , test_set)[, -1]
y_test <- test_set$Apps

set.seed(1)
```

```r
grid <- 10^seq(10, -2, length = 100)
cv_out <- cv.glmnet(x_train, y_train, alpha = 0, lamda = grid)
best_lamda <- cv_out$lambda.min

ridge_pred_best <- round(predict(cv_out, s = best_lamda, newx = x_test))
ridge_mse_best <- mean((ridge_pred_best - y_test)^2)
ridge_mse_best
```

```
## [1] 1238906
```

d) Fit a lasso model on the training set, with lambda chosen by cross- validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```r
set.seed(1)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lamda = grid)
best_lambda_lasso <- cv_lasso$lambda.min
lasso_model <- glmnet(x_train, y_train, alpha = 1, lamda = best_lambda_lasso)
lasso_pred <- predict(lasso_model, s = best_lamda, newx = x_test)
lasso_mse <- mean((lasso_pred - y_test)^2)
lasso_mse
```

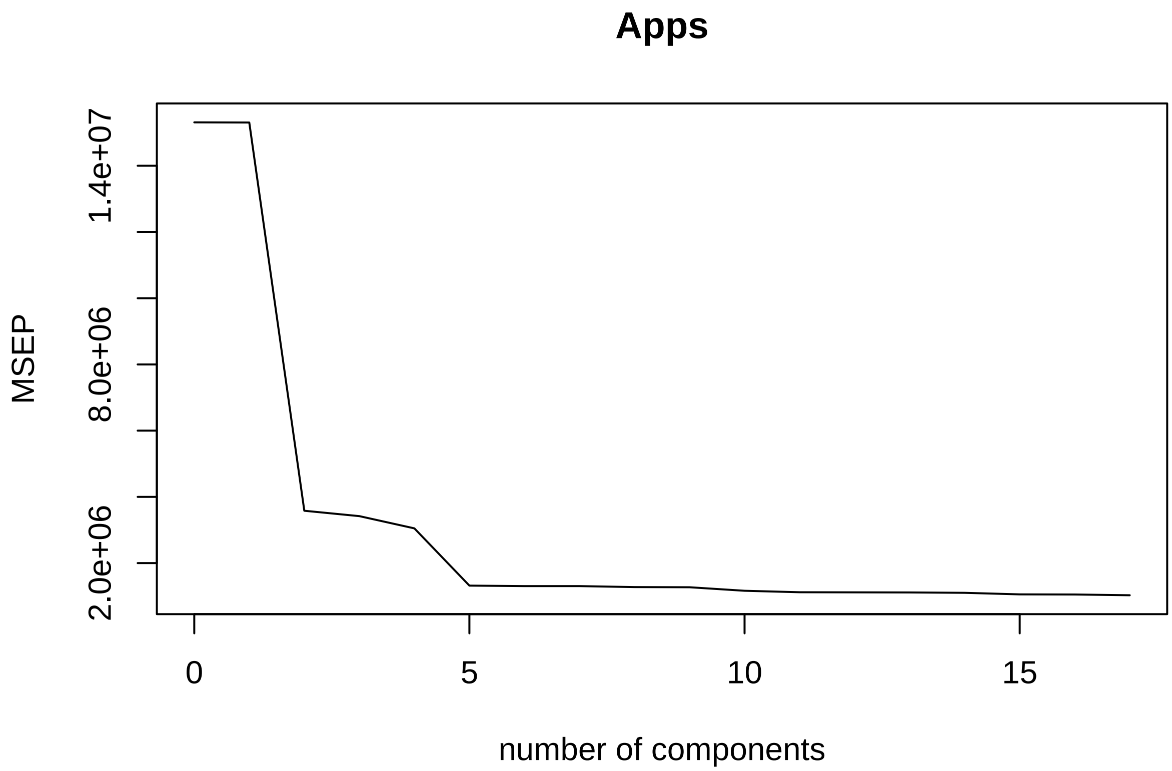
```
## [1] 1707890
```

```r
lasso_coef <- predict(lasso_model, type = "coefficients", s = best_lambda_lasso)
non_zero_coef <- sum(lasso_coef != 0) - 1
non_zero_coef
```

```
## [1] 13
```

e) Fit a PCR model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

```r
set.seed(2)
pcr_model <- pcr(Apps ~ . , data = training_set, validaton = "CV")
validationplot(pcr_model, val.type = "MSEP") # min MSEP at 15 components
```

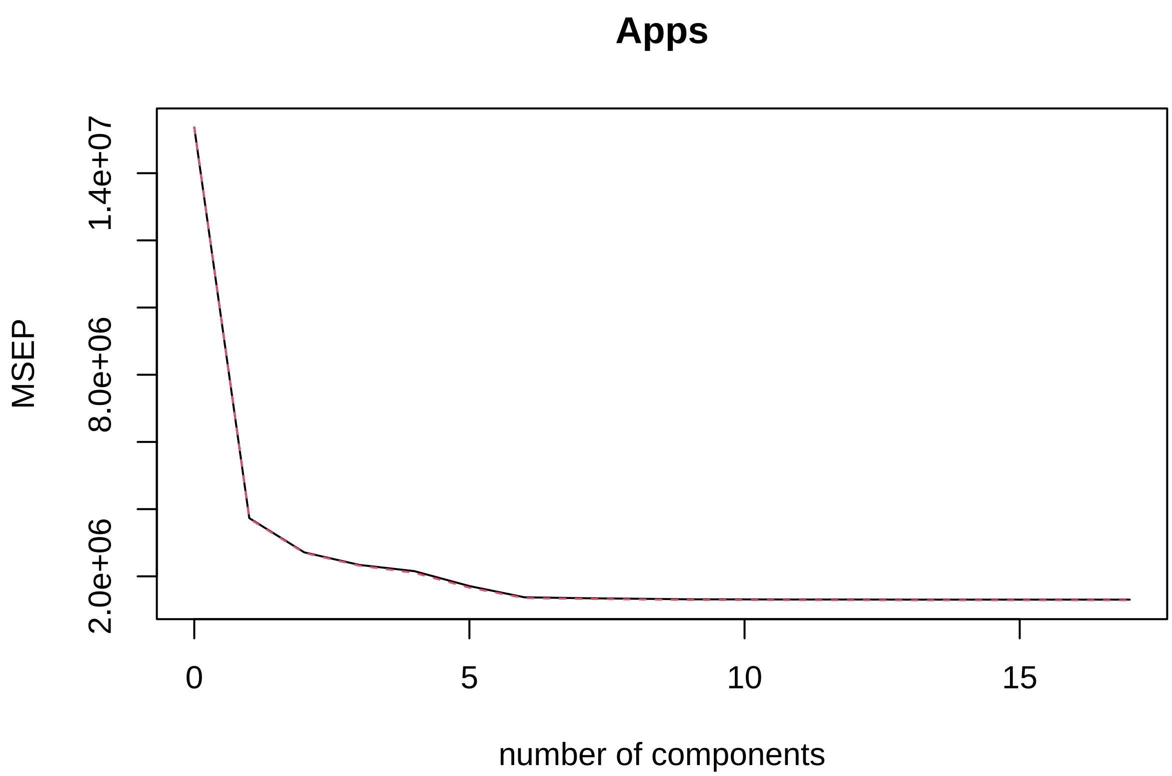



```
pcr_pred <- predict(pcr_model, newdata = test_set, ncomp = 15)
pcr_mse <- mean((pcr_pred - test_set$Apps)^2)
pcr_mse
```

```
## [1] 1233054
```

f) Fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

```
set.seed(3)

pls_model <- plsr(Apps ~ ., data = training_set, scale = TRUE, validation = "CV")
validationplot(pls_model, val.type = "MSEP") # min MSEP at 7 components
```




```
pls_pred <- predict(pls_model, newdata = test_set, ncomp = 7)
pls_mse <- mean((pls_pred - test_set$Apps)^2)
pls_mse
```

```
## [1] 1288563
```

g) Comment on the results obtained. How accurately can we pre dict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

There are notable differences in the results obtained from the differnt modeling approaches predicting the number of college applications. The linear model using least sqaures gave the lowest MSE of 110.571, which was the lowest of all models, suggesting that it provides the most accurate predictions. The lasso approach gave the highest test MSE with 170890 signifying the lowest predictive capability of all the models. The PCR and PLS models gave MSE values of 1233054 and 1288563 which compare but are slightly less than the MSE from ridge regression, 1238906. The linear model stands out from the other models in its predictive accuracy.

Exercise 11

a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
Boston <- Boston
split_data <- initial_split(Boston, prop = 0.75)
train_set <- training(split_data)
test_set <- testing(split_data)
```

Best Subset Selection

```
set.seed(1)

best_subset <- regsubsets(crim ~ . , data = train_set, nvmax = 13)
best_subset_summary <- summary(best_subset)
```

```
test.mat <- model.matrix(crim ~ . , data = test_set)
val.errors <- rep(NA, 13)
for (i in 1:13) {
  coefi <- coef(best_subset, id = i)
  pred <- test.mat[, names(coefi)] %*% coefi
  val.errors[i] <- mean((test_set$crim - pred)^2)
}
which.min(val.errors) #the best model is the one that contains 12 variables, has lowest validation erro
```

```
## [1] 12
```

```
coef(best_subset, 12)
```

```
##   (Intercept)            zn         indus          chas           nox
##  22.782626958   0.048044009  -0.056222750  -0.834958657 -14.740555507
##            rm           dis           rad           tax       ptratio
##   0.423483508  -1.208060077   0.671799565  -0.006027167  -0.384226935
##         black         lstat          medv
##  -0.005456931   0.145136999  -0.248838009
```

```
val.errors #test mse of 67.25
```

```
##  [1] 15.69931 16.24096 15.66656 15.32826 16.24802 16.29269 16.35605 15.51679
##  [9] 15.49401 15.36881 15.36390 15.28466 15.35448
```

Ridge Regression

```
set.seed(1)
train_x <- model.matrix(crim ~ . , train_set)[, -1]
train_y <- train_set$crim
test_x <- model.matrix(crim ~ . , test_set)[, -1]
test_y <- test_set$crim

grid <- 10^seq(10, -2, length = 100)
ridge_cv <- cv.glmnet(train_x, train_y, alpha = 0, lamda = grid)
best_lambda_ridge <- ridge_cv$lambda.min

ridge_pred <-predict(ridge_cv, s = best_lambda_ridge, newx= test_x)
ridge_mse <- mean((ridge_pred - test_y)^2)
ridge_mse # test MSE of 68.81
```

```
## [1] 13.88244
```

Lasso Regression

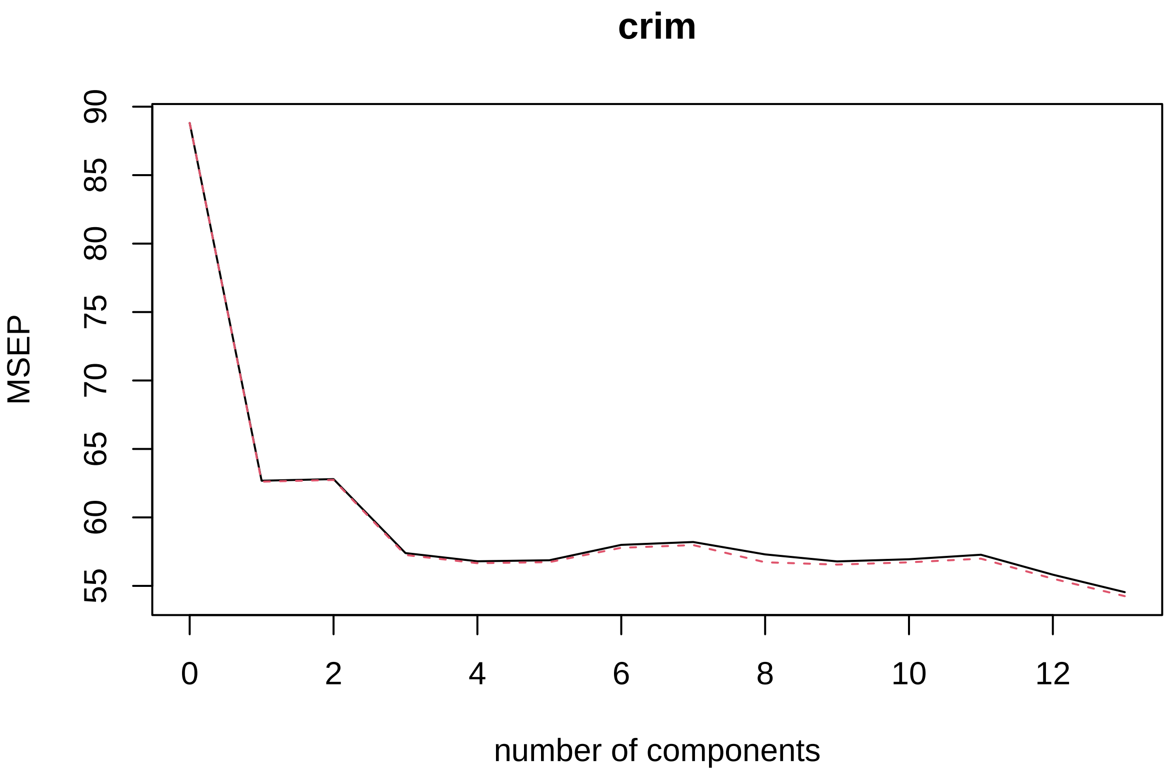
```
set.seed(1)
lasso_cv <- cv.glmnet(train_x, train_y, alpha = 1, lamda = grid)
lasso_lamda <- lasso_cv$lambda.min

pred_lasso <- predict(lasso_cv, s = lasso_lamda, newx = test_x)
mse_lasso <- mean((pred_lasso - test_y)^2)
mse_lasso #test MSE of 68.15
```

```
## [1] 14.315
```

PCR

```
set.seed(1)
pcr_crim <- pcr(crim ~ . , data = train_set, scale = TRUE, validation = "CV")
```

```r
validationplot(pcr_crim, val.type = "MSEP") #lowest MSEP at 8 components
```




```r
pcr_crim_pred <- predict(pcr_crim, newdata = test_set, ncomp = 8)
pcr_crim_mse <- mean((pcr_crim_pred - test_set$crim)^2)
pcr_crim_mse #test MSE of 70.86
```

```
## [1] 15.16245
```

Based on the models shown in predicting the per capita crime rate in the Boston dataset, the best subset selection approache seems to be the best option for its prediction accuracy based on having the lowest test MSE of 67.25. The PCR approach is shown to perform the worst of all the models with the highest test MSE of 70.8659. Whereas, ridge regression and the lasso give similar results in test MSE with values of 68.81 and 68.15.

b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, crossvalidation, or some other reasonable alternative, as opposed to using training error.

```r
set.seed(1)
crim_fwd <- regsubsets(crim ~ ., data = train_set, nvmax = 13, method = "forward")
summary(crim_fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = train_set, nvmax = 13, method = "forward")
## 13 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
```

```
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: forward
##           zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
## 1  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " "   " " 
## 2  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " "   "*" 
## 3  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   "*"   "*" 
## 4  ( 1 )  "*" " "   " "  " " " " " " " " "*" " " " "     " "   "*"   "*" 
## 5  ( 1 )  "*" " "   " "  " " " " " " "*" "*" " " " "     " "   "*"   "*" 
## 6  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " " "     " "   "*"   "*" 
## 7  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " "*"     " "   "*"   "*" 
## 8  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"   "*" 
## 9  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" "*" "*"     "*"   "*"   "*" 
## 10  ( 1 ) "*" "*"   " "  "*" " " " " "*" "*" "*" "*"     "*"   "*"   "*" 
## 11  ( 1 ) "*" "*"   " "  "*" "*" " " "*" "*" "*" "*"     "*"   "*"   "*" 
## 12  ( 1 ) "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"   "*" 
## 13  ( 1 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"   "*"
```

```r
k <- 10
n <- nrow(train_set)
set.seed(1)
folds <- sample(rep(1:k, length = n))
crim.cv.errors <- matrix(NA, k, 13, dimnames = list(NULL, paste(1:13))) #cv to choose among models of d

predict.regsubsets <- function(object, newdata, id, ...) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[, xvars] %*% coefi
} #there is no predict() method for regsubsets(), write our own predict method from lab

for (j in 1:k) {
  best.fit <- regsubsets(crim ~ ., data = train_set[folds != j, ], nvmax = 13, method = "forward")
  for (i in 1:13) {
    pred <- predict.regsubsets(best.fit, train_set[folds == j, ], id = i)
    crim.cv.errors[j, i] <- mean((train_set$crim[folds == j] - pred)^2)
  }
} #for loop that performs CROSS-VALIDATION from lab

mean.cv.errors <- apply(crim.cv.errors, 2, mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8 
## 56.60163 55.32511 55.33387 54.93172 54.62816 54.36416 53.81020 53.53562 
##        9       10       11       12       13 
## 53.10555 53.23397 53.41121 53.26017 53.27381
```

```r
best_model_size <- which.min(mean.cv.errors) #best moddel based on lowest cv error
best_model_size #model with 9 varibales gives lowest cv error
```

```
## 9
## 9
```

```r
regfit_best <- regsubsets(crim ~ ., data = train_set, nvmax = 13, method = "forward") #model fit on tra
final_coef <- coef(regfit_best, best_model_size)

test.mat <- model.matrix(crim ~ ., data = test_set)
test_pred <- test.mat[, names(final_coef)] %*% final_coef #predictions on test from final model

test_mse <- mean((test_set$crim - test_pred)^2)
test_mse #67.51 test MSE
```

```
## [1] 15.49401
```

c) Does your chosen model involve all of the features in the data set? Why or why not?

My model does not involve all of the features in the data set becauase I chose to use the stepwise selection approach for my model which includes only the vairbales that will provide the optimal model in prediction accuracy on the validation set. I used k-cross validation and allocated the each observation to one of k = 10 folds and stored the results of the test error from the for loops in order to choose which number of variables provides the best test error. This was 9 variables which is why my model does not invole all of the variables in the dataset.