Midterm Take Home

Gus Lipkin ~ glipkin6737@floridapoly.edu

```
# load packages
library(tidyverse)
library(data.table)
library(leaps)
library(glmnet)
library(caret)
```

Intro

Note: If code is repeated, it is only commented the first time

```
# load and preview data
dt <- data.table(ISLR2::Boston)
head(dt)
```

```
##       crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1: 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2: 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3: 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4: 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5: 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6: 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

The `ISLR2::Boston` dataset contains "A data set containing housing values in 506 suburbs of Boston." If you want to learn more, I suggest visiting https://rdocumentation.org/packages/ISLR2/versions/1.3-1/topics/Boston.

Variable	Description	Type
`crim`	per capita crime rate by town.	double
`zn`	proportion of residential land zoned for lots over 25,000 sq.ft.	double
`indus`	proportion of non-retail business acres per town.	double
`chas`	Charles River dummy variable (=1 if tract bounds river; 0 otherwise).	integer (boolean)
`nox`	nitrogen oxides concentration (parts per 10 million).	double
`rm`	average number of rooms per dwelling.	double
`age`	proportion of owner-occupied units built prior to 1940	double
`dis`	weighted mean of distances to five Boston employment centres.	double
`rad`	index of accessibility to radial highways	integer
`tax`	full-value property-tax rate per $10,000.	double
`ptratio`	pupil-teacher ratio by town.	double
`lstat`	lower status of the population (percent).	double

Variable	Description	Type
medv	median value of owner-occupied homes in $1000s.	double

Summary Stats

```r
# get summary stats
summary(dt)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          lstat      
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95  
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97  
##       medv      
##  Min.   : 5.00  
##  1st Qu.:17.02  
##  Median :21.20  
##  Mean   :22.53  
##  3rd Qu.:25.00  
##  Max.   :50.00
```

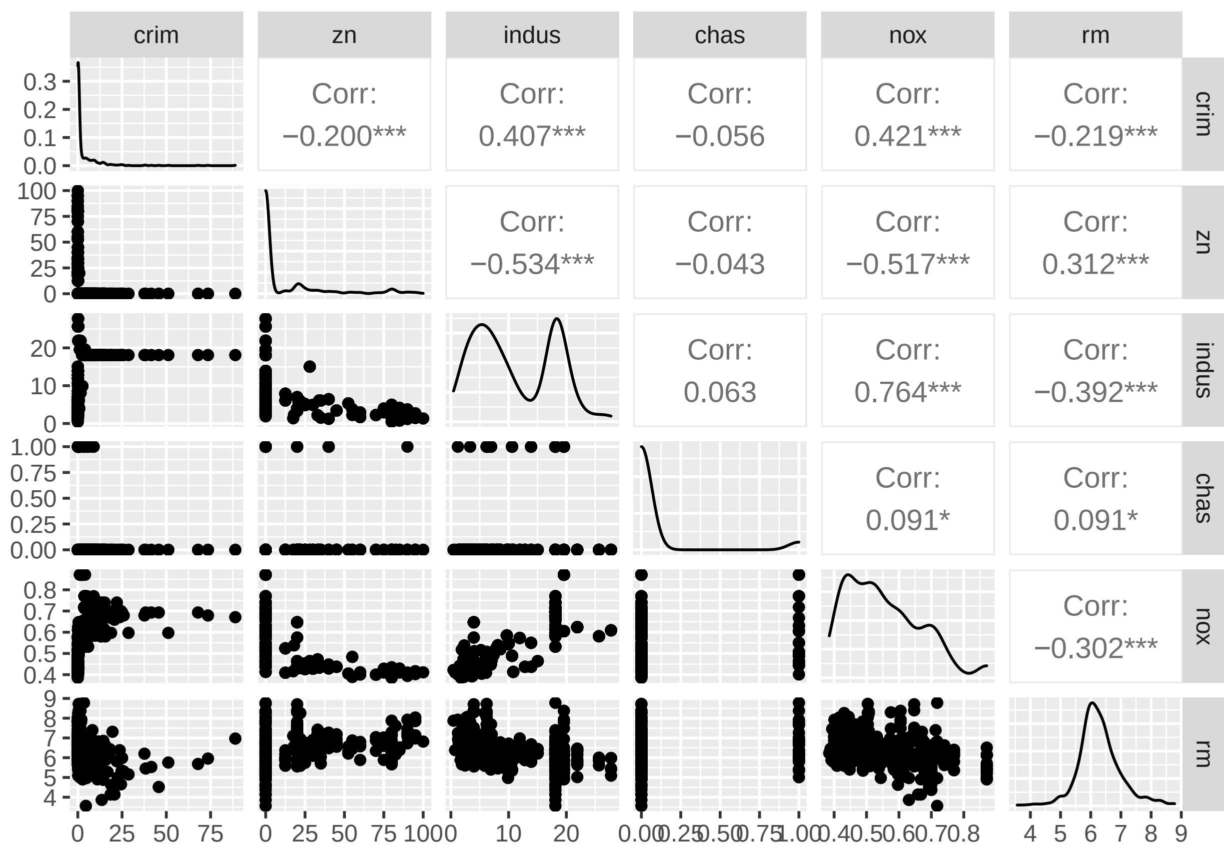
```r
# check for any correlation coefficients over .75
df <- dewey::ifelsedata(data.frame(round(cor(dt), 3)),
                        .75, "x >= y & x != 1", matchCols = FALSE)
# set the row names
rownames(df) <- colnames(df)
# preview the correlation matrix
df
```

```
##          crim zn indus chas   nox rm age dis rad tax ptratio lstat medv
## crim       NA NA    NA   NA    NA NA  NA  NA  NA  NA      NA    NA   NA
## zn         NA NA    NA   NA    NA NA  NA  NA  NA  NA      NA    NA   NA
## indus      NA NA    NA   NA 0.764 NA  NA  NA  NA  NA      NA    NA   NA
```

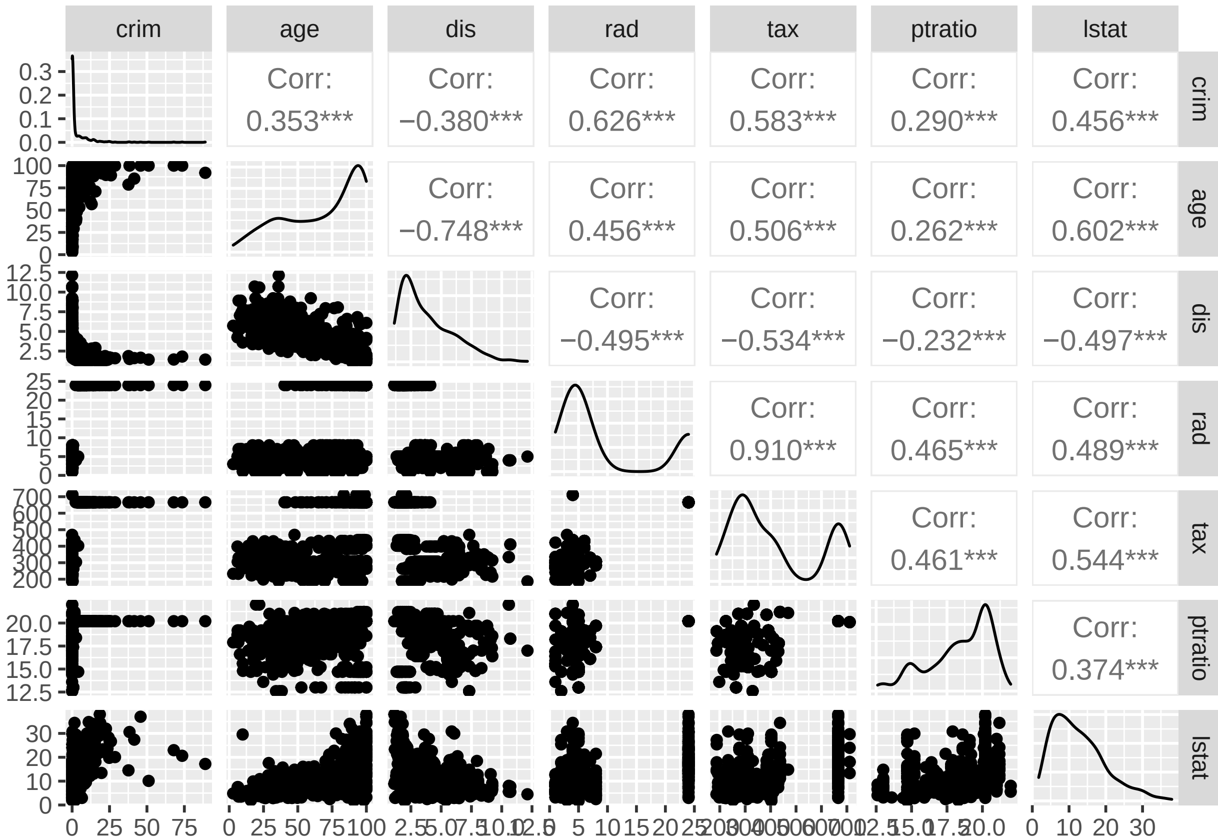
```
## chas       NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## nox        NA NA 0.764    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## rm         NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## age        NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## dis        NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## rad        NA NA     NA    NA    NA NA  NA  NA    NA  0.91       NA     NA    NA
## tax        NA NA     NA    NA    NA NA  NA  NA  0.91    NA       NA     NA    NA
## ptratio    NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## lstat      NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
## medv       NA NA     NA    NA    NA NA  NA  NA    NA    NA       NA     NA    NA
```

```
# produce pairs plots with correlation coefficient
GGally::ggpairs(dt[, c(1:6)], progress = FALSE)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```




```
GGally::ggpairs(dt[, c(1, 7:12)], progress = FALSE)
```



The first output is the basic summary statistics, the second is a correlation matrix, but only keeping values above .75 There's nothing crazy with these numbers. It is weird that only `tax` and `rad` are correlated above .75, but then again highways decrease property taxes or something. idk. I'm not an urban anything.

Splitting the data

```r
# set the seed
set.seed(123)

# randomly generate TRUE/FALSE to split the data at an 80/20 split
rowPicker <- sample(c(TRUE, FALSE), nrow(dt), replace = TRUE, prob = c(.8, .2))

# split the data
trainDt <- dt[rowPicker]
testDt <- dt[!rowPicker]
```

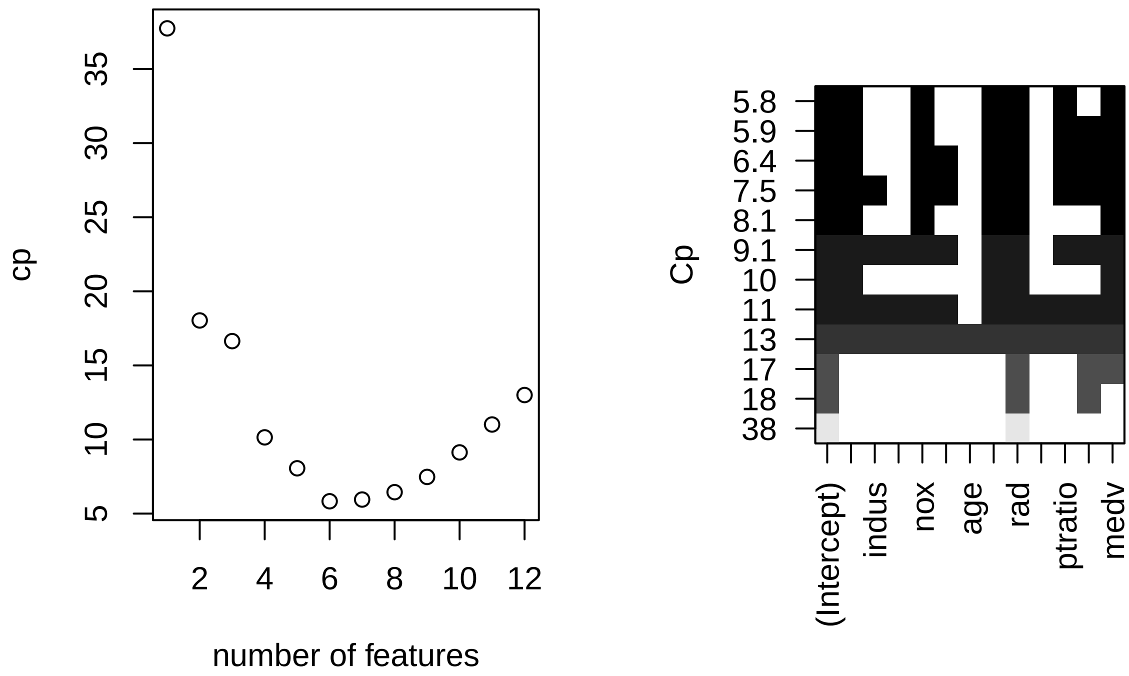
Subset selection

Normal

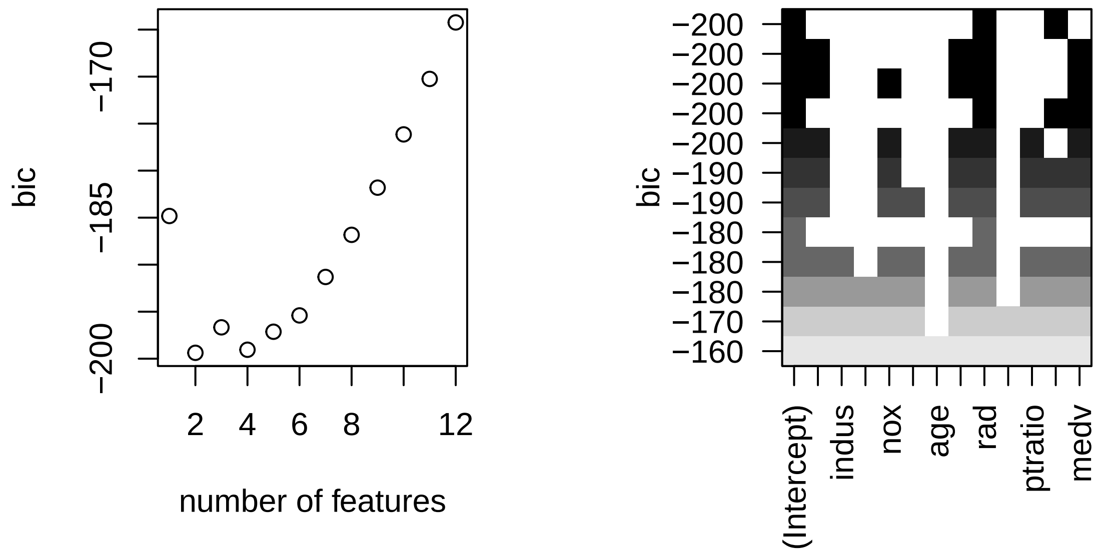
```r
# run `regsubsets`
best_fit <- regsubsets(crim ~ ., trainDt, nvmax = 12)
best_summary <- summary(best_fit)

# create a data.table of the BIC, CP, and R^2
data.table("BIC" = best_summary$bic,
           "Cp" = best_summary$cp,
           "r2" = best_summary$adjr2)[order(r2 * -1, BIC, Cp)]
```

```
##           BIC        Cp        r2
##  1: -186.8296  6.447092 0.4318334
##  2: -181.8102  7.474736 0.4318031
##  3: -191.3088  5.948498 0.4311148
##  4: -176.1453  9.129592 0.4308783
##  5: -195.4015  5.834203 0.4298571
##  6: -170.2449 11.012837 0.4296224
##  7: -164.2371 13.000000 0.4282112
##  8: -197.1359  8.057768 0.4253132
##  9: -199.0502 10.145538 0.4209823
## 10: -196.6675 16.638599 0.4104990
## 11: -199.3763 18.033853 0.4071937
## 12: -184.8256 37.744021 0.3783624
```

```r
# show the CP chart in two formats side-by-side
par(mfrow = c(1,2))
plot(best_summary$cp, xlab = "number of features", ylab = "cp")
plot(best_fit, scale = "Cp")
```




```
# show the BIC chart in two formats side-by-side
par(mfrow = c(1, 2))
plot(best_summary$bic, xlab = "number of features", ylab = "bic")
plot(best_fit, scale = "bic")
```




```r
# save the best formula
normal <- as.formula("crim ~ + zn + nox + dis + rad + ptratio + lstat + medv")
```

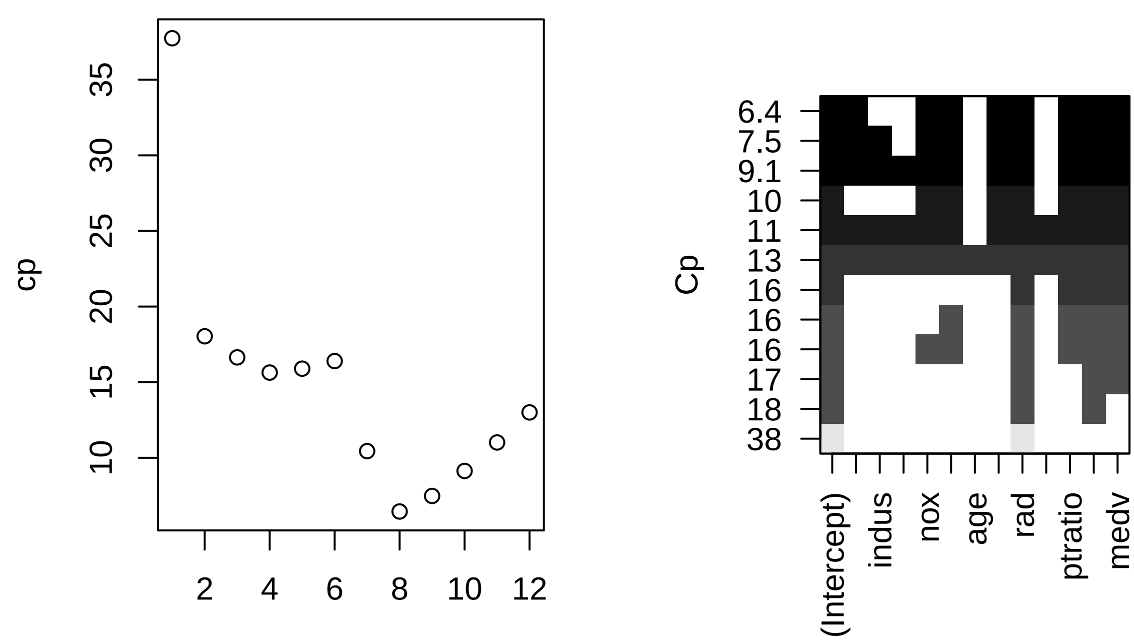
Forward

```r
best_fit <- regsubsets(crim ~ ., trainDt, nvmax = 12, method = "forward")
best_summary <- summary(best_fit)

data.table("BIC" = best_summary$bic,
           "Cp" = best_summary$cp,
           "r2" = best_summary$adjr2)[order(r2 * -1, BIC, Cp)]
```

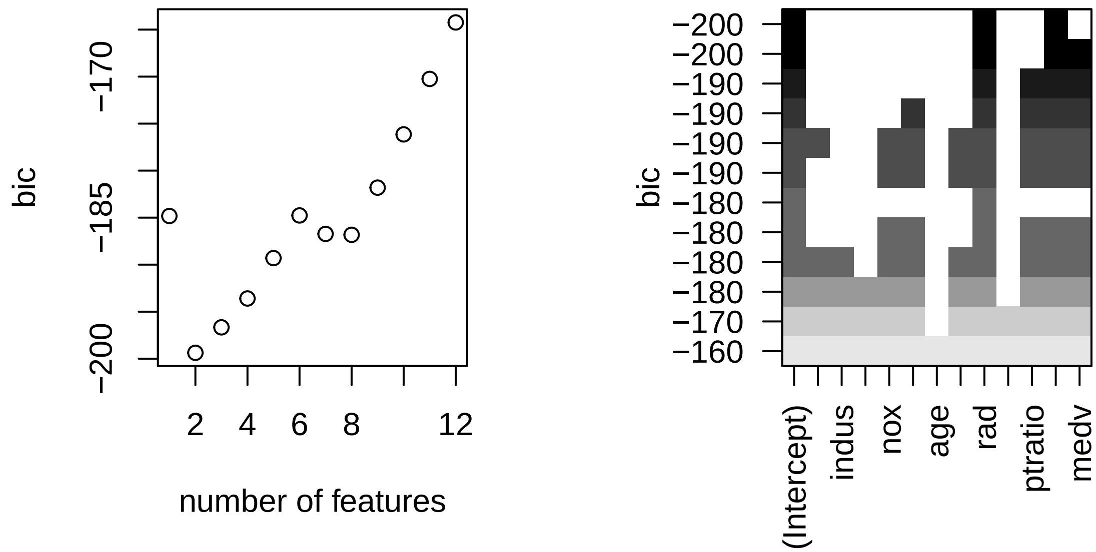
```
##            BIC        Cp        r2
##  1: -186.8296  6.447092 0.4318334
##  2: -181.8102  7.474736 0.4318031
##  3: -176.1453  9.129592 0.4308783
##  4: -170.2449 11.012837 0.4296224
##  5: -164.2371 13.000000 0.4282112
##  6: -186.7314 10.439187 0.4247590
##  7: -184.7655 16.395178 0.4149469
##  8: -189.3015 15.891514 0.4142806
##  9: -193.5994 15.634578 0.4132709
## 10: -196.6675 16.638599 0.4104990
```

```
## 11: -199.3763 18.033853 0.4071937
## 12: -184.8256 37.744021 0.3783624
```

```r
par(mfrow = c(1,2))
plot(best_summary$cp, xlab = "number of features", ylab = "cp")
plot(best_fit, scale = "Cp")
```




```r
par(mfrow = c(1, 2))
plot(best_summary$bic, xlab = "number of features", ylab = "bic")
plot(best_fit, scale = "bic")
```



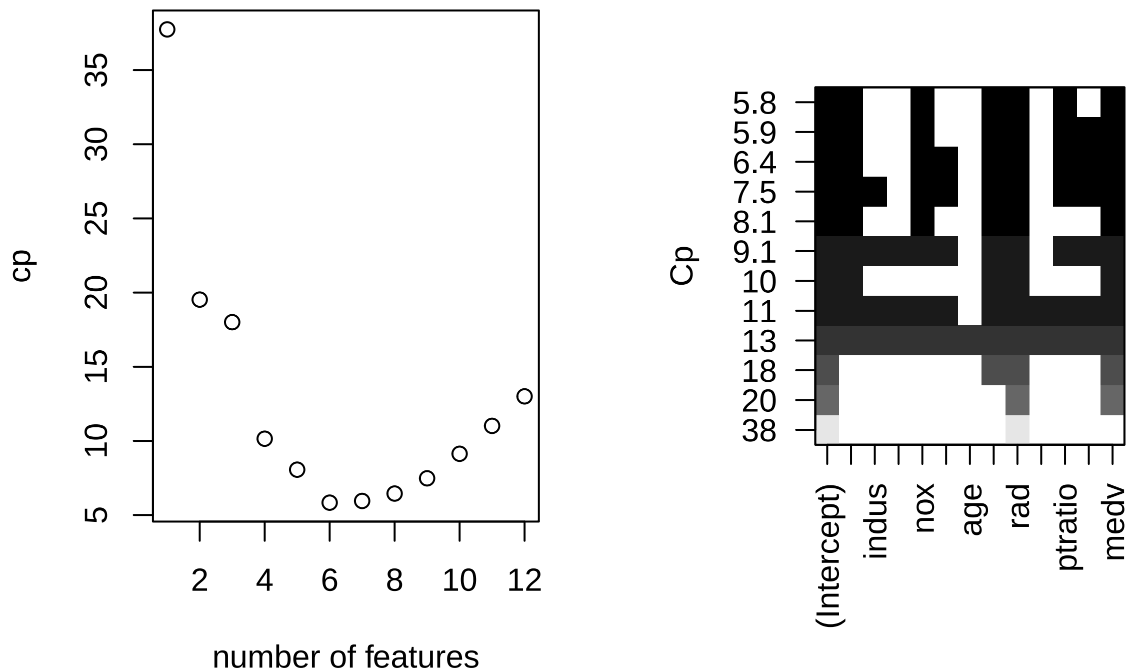

```
forward <- as.formula("crim ~ + rad + lstat")
```

Backward

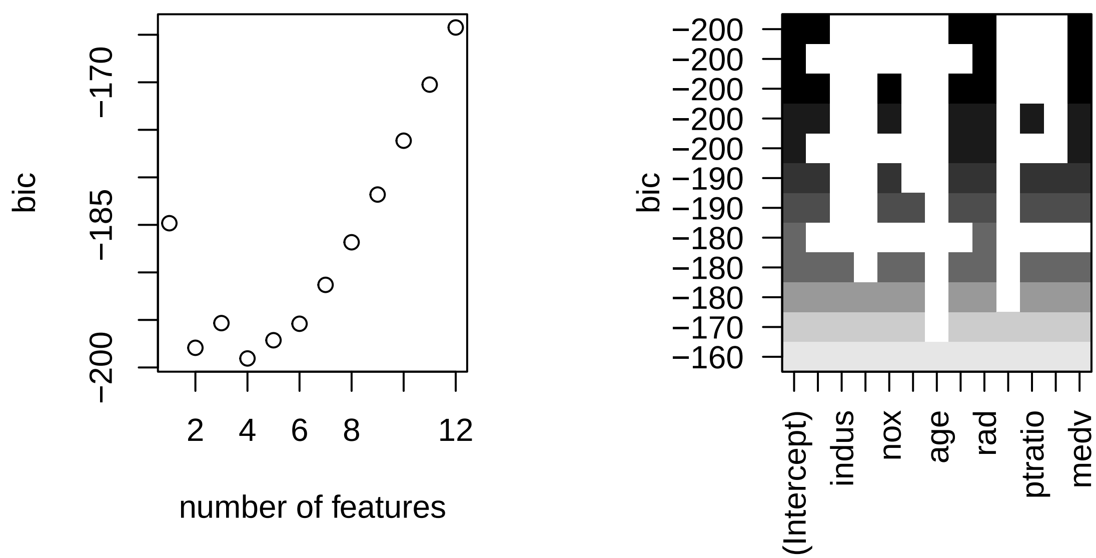
```
best_fit <- regsubsets(crim ~ ., trainDt, nvmax = 12, method = "backward")
best_summary <- summary(best_fit)

data.table("BIC" = best_summary$bic,
           "Cp" = best_summary$cp,
           "r2" = best_summary$adjr2)[order(r2 * -1, BIC, Cp)]
```

```
##            BIC        Cp        r2
##  1: -186.8296  6.447092 0.4318334
##  2: -181.8102  7.474736 0.4318031
##  3: -191.3088  5.948498 0.4311148
##  4: -176.1453  9.129592 0.4308783
##  5: -195.4015  5.834203 0.4298571
##  6: -170.2449 11.012837 0.4296224
##  7: -164.2371 13.000000 0.4282112
##  8: -197.1359  8.057768 0.4253132
##  9: -199.0502 10.145538 0.4209823
## 10: -195.3334 18.002912 0.4085869
## 11: -197.9301 19.524980 0.4051090
## 12: -184.8256 37.744021 0.3783624
```

```
par(mfrow = c(1,2))
plot(best_summary$cp, xlab = "number of features", ylab = "cp")
plot(best_fit, scale = "Cp")
```




```
par(mfrow = c(1, 2))
plot(best_summary$bic, xlab = "number of features", ylab = "bic")
plot(best_fit, scale = "bic")
```




```
backward <- as.formula("crim ~ + zn + dis + rad + medv")
```

Final selection

```
# run regsearch to find the best model
regs <- dewey::regsearch(trainDt, "crim", c(colnames(trainDt[, !c("crim")]), "lstat*rad"), 1, 12, "gaus
```

```
## [1] "Assembling regressions..."
## [1] "Creating 8190 formulas. Please be patient, this may take a while."
## [1] "Creating regressions..."
## [1] "Running 5119 regressions. Please be patient, this may take a while."
## [1] "Running regressions..."
```

```
regs
```

```
##                     formula      aic      bic rSquare warn   xIntercept crim
##    1:            crim ~ + rad 2807.800 2819.864 0.37987   NA 8.046658e-06   NA
##    2:            crim ~ + tax 2836.126 2848.190 0.33574   NA 4.991853e-19   NA
##    3:          crim ~ + lstat 2906.574 2918.637 0.21187   NA 1.571262e-05   NA
##    4:            crim ~ + nox 2927.416 2939.480 0.17097   NA 6.154631e-12   NA
##    5:          crim ~ + indus 2932.558 2944.621 0.16056   NA 6.000980e-03   NA
##   ---
## 5115:    crim ~ + chas + medv 2938.297 2954.381 0.15291   NA 2.419867e-26   NA
```

```
## 5116:       crim ~ + age + zn 2950.419 2966.504 0.12762   NA 4.466770e-03   NA
## 5117: crim ~ + chas + ptratio 2969.617 2985.701 0.08600   NA 9.139189e-07   NA
## 5118:      crim ~ + chas + rm 2982.391 2998.475 0.05722   NA 1.155439e-08   NA
## 5119:           crim ~ + chas 3003.200 3015.263 0.00356   NA 5.978764e-16   NA
##              zn        indus      chas          nox          rm          age
##    1:        NA           NA        NA           NA          NA           NA
##    2:        NA           NA        NA           NA          NA           NA
##    3:        NA           NA        NA           NA          NA           NA
##    4:        NA           NA        NA 1.906815e-18          NA           NA
##    5:        NA 2.539146e-17        NA           NA          NA           NA
##   ---
## 5115:        NA           NA 0.9444930           NA          NA           NA
## 5116: 0.8523224           NA        NA           NA          NA 5.135844e-10
## 5117:        NA           NA 0.6180833           NA          NA           NA
## 5118:        NA           NA 0.4476415           NA 1.97917e-06           NA
## 5119:        NA           NA 0.2269931           NA          NA           NA
##       dis          rad          tax      ptratio       lstat         medv
##    1:  NA 1.828185e-44           NA           NA          NA           NA
##    2:  NA           NA 2.567657e-38           NA          NA           NA
##    3:  NA           NA           NA           NA 5.38158e-23           NA
##    4:  NA           NA           NA           NA          NA           NA
##    5:  NA           NA           NA           NA          NA           NA
##   ---
## 5115:  NA           NA           NA           NA          NA 3.80451e-16
## 5116:  NA           NA           NA           NA          NA           NA
## 5117:  NA           NA           NA 2.855178e-09          NA           NA
## 5118:  NA           NA           NA           NA          NA           NA
## 5119:  NA           NA           NA           NA          NA           NA
##       lstat.rad
##    1:        NA
##    2:        NA
##    3:        NA
##    4:        NA
##    5:        NA
##   ---
## 5115:        NA
## 5116:        NA
## 5117:        NA
## 5118:        NA
## 5119:        NA
```

```r
# select the best model and save it
dewey <- as.formula("crim ~ + lstat + rad")
```

```r
# create a vector of all formulas
forms <- c(normal, forward, backward, dewey)

# lapply(forms, function(x) {
#   print(x)
#   summary(lm(formula = x, testDt))
#   })

# print the formula and summary stats for each one
for(x in forms) {
```

```r
print(x)
print(summary(lm(formula = x, testDt)))
}
```

```
## crim ~ +zn + nox + dis + rad + ptratio + lstat + medv
##
## Call:
## lm(formula = x, data = testDt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2349 -1.2980 -0.0535  0.8753 13.9012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.63421    5.93751   0.612    0.542
## zn           0.02007    0.02072   0.969    0.335
## nox         -5.32924    5.01379  -1.063    0.291
## dis         -0.29191    0.24244  -1.204    0.232
## rad          0.43554    0.04809   9.057 3.77e-14 ***
## ptratio     -0.09728    0.18226  -0.534    0.595
## lstat        0.12934    0.08337   1.551    0.124
## medv        -0.05048    0.06593  -0.766    0.446
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.841 on 86 degrees of freedom
## Multiple R-squared:  0.7015, Adjusted R-squared:  0.6772
## F-statistic: 28.87 on 7 and 86 DF,  p-value: < 2.2e-16
##
## crim ~ +rad + lstat
##
## Call:
## lm(formula = x, data = testDt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3812 -0.9701  0.1127  0.7752 14.2313
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.29654    0.64550  -5.107 1.79e-06 ***
## rad          0.41729    0.03731  11.186  < 2e-16 ***
## lstat        0.15302    0.04708   3.250  0.00162 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.811 on 91 degrees of freedom
## Multiple R-squared:  0.6909, Adjusted R-squared:  0.6841
## F-statistic: 101.7 on 2 and 91 DF,  p-value: < 2.2e-16
##
## crim ~ +zn + dis + rad + medv
##
```

```
## Call:
## lm(formula = x, data = testDt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1607 -0.9204 -0.0167  0.4745 14.2917
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.02802    1.28714   1.576   0.1187
## zn           0.02386    0.01930   1.236   0.2196
## dis         -0.31035    0.18399  -1.687   0.0952 .
## rad          0.40829    0.04075  10.020 2.92e-16 ***
## medv        -0.10350    0.03973  -2.605   0.0108 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.846 on 89 degrees of freedom
## Multiple R-squared:  0.6899, Adjusted R-squared:  0.676
## F-statistic:  49.5 on 4 and 89 DF,  p-value: < 2.2e-16
##
## crim ~ +lstat + rad
##
## Call:
## lm(formula = x, data = testDt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3812 -0.9701  0.1127  0.7752 14.2313
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.29654    0.64550  -5.107 1.79e-06 ***
## lstat        0.15302    0.04708   3.250  0.00162 **
## rad          0.41729    0.03731  11.186  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.811 on 91 degrees of freedom
## Multiple R-squared:  0.6909, Adjusted R-squared:  0.6841
## F-statistic: 101.7 on 2 and 91 DF,  p-value: < 2.2e-16
```

10-fold Validation

```r
# set the trainControl method to 10-fold
train_control <- trainControl(method = "cv", number = 10)

# train the model
model <- train(normal, data = trainDt, method = "lm",
                     trControl = train_control)
# use the model to predict for the test data
prediction_ridge <- predict(model, newdata = testDt)
```

```r
# save the stats for the current model
Normal <- c("Type" = "10 Fold",
            "R_Square" = R2(prediction_ridge, testDt$crim),
            "RMSE" = RMSE(prediction_ridge, testDt$crim),
            "MAE" = MAE(prediction_ridge, testDt$crim))

train_control <- trainControl(method ="cv", number = 10)
model <- train(forward, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model, newdata = testDt)
Forward <- c("Type" = "10 Fold",
             "R_Square" = R2(prediction_ridge, testDt$crim),
             "RMSE" = RMSE(prediction_ridge, testDt$crim),
             "MAE" = MAE(prediction_ridge, testDt$crim))

model <- train(backward, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model, newdata = testDt)
Backward <- c("Type" = "10 Fold",
              "R_Square" = R2(prediction_ridge, testDt$crim),
              "RMSE" = RMSE(prediction_ridge, testDt$crim),
              "MAE" = MAE(prediction_ridge, testDt$crim))


model <- train(dewey, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model, newdata = testDt)
Dewey <- c("Type" = "10 Fold",
           "R_Square" = R2(prediction_ridge, testDt$crim),
           "RMSE" = RMSE(prediction_ridge, testDt$crim),
           "MAE" = MAE(prediction_ridge, testDt$crim))

# bind all the model results into a data.frame
regStats <- rbind(Normal, Forward, Backward, Dewey)
```

LOOCV

```r
train_control <- trainControl(method = "LOOCV")


model_ridge <- train(normal, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model_ridge, newdata = testDt)
Normal <- c("Type" = "LOOCV",
            "R_Square" = R2(prediction_ridge, testDt$crim),
            "RMSE" = RMSE(prediction_ridge, testDt$crim),
            "MAE" = MAE(prediction_ridge, testDt$crim))

train_control <- trainControl(method ="cv", number = 10)
model_ridge <- train(forward, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model_ridge, newdata = testDt)
```

```r
Forward <- c("Type" = "LOOCV",
             "R_Square" = R2(prediction_ridge, testDt$crim),
             "RMSE" = RMSE(prediction_ridge, testDt$crim),
             "MAE" = MAE(prediction_ridge, testDt$crim))

model_ridge <- train(backward, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model_ridge, newdata = testDt)
Backward <- c("Type" = "LOOCV",
              "R_Square" = R2(prediction_ridge, testDt$crim),
              "RMSE" = RMSE(prediction_ridge, testDt$crim),
              "MAE" = MAE(prediction_ridge, testDt$crim))


model_ridge <- train(dewey, data = trainDt, method = "lm",
                     trControl = train_control)
prediction_ridge <- predict(model_ridge, newdata = testDt)
Dewey <- c("Type" = "LOOCV",
           "R_Square" = R2(prediction_ridge, testDt$crim),
           "RMSE" = RMSE(prediction_ridge, testDt$crim),
           "MAE" = MAE(prediction_ridge, testDt$crim))


regStats <- rbind(regStats, Normal, Forward, Backward, Dewey)
```

```r
# preview all the regressions so far
regStats
```

```
##          Type      R_Square            RMSE               MAE
## Normal   "10 Fold" "0.672415531514889" "3.76250483312505" "2.84888093130358"
## Forward  "10 Fold" "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Backward "10 Fold" "0.676189775369158" "3.62179308014776" "2.61017890603348"
## Dewey    "10 Fold" "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Normal   "LOOCV"   "0.672415531514889" "3.76250483312505" "2.84888093130358"
## Forward  "LOOCV"   "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Backward "LOOCV"   "0.676189775369158" "3.62179308014776" "2.61017890603348"
## Dewey    "LOOCV"   "0.688825316227434" "3.41610927790962" "2.34648962836827"
```

Again, `regsubsets` produces slightly better models, but mine is almost as good and is more *parsimonious*. As `lstat` increases by one, `crim` increases by .237 and when `rad` increases by one, `crim` increases by .522.

Ridge and Lasso Regression

10-fold Validation

```r
# First define the traincontrol to specify k-fold
train_control <- trainControl(method ="cv", number = 10)

# define the lambda
lambda <- 10^seq(-2, 5, length = 1000)
```

Ridge Regression

```
model_ridge <- train(crim ~ ., data = trainDt, method = "glmnet",
                     trControl = train_control,
                     tuneGrid = expand.grid(alpha = 0, lambda = lambda))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
summary(model_ridge)
```

```
##             Length Class      Mode
## a0           100   -none-     numeric
## beta        1200   dgCMatrix  S4
## df           100   -none-     numeric
## dim            2   -none-     numeric
## lambda       100   -none-     numeric
## dev.ratio    100   -none-     numeric
## nulldev        1   -none-     numeric
## npasses        1   -none-     numeric
## jerr           1   -none-     numeric
## offset         1   -none-     logical
## call           5   -none-     call
## nobs           1   -none-     numeric
## lambdaOpt      1   -none-     numeric
## xNames        12   -none-     character
## problemType    1   -none-     character
## tuneValue      2   data.frame list
## obsLevels      1   -none-     logical
## param          0   -none-     list
```

```
# get the coefficients
coef(model_ridge$finalModel, model_ridge$bestTune$lambda)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  7.8457780777
## zn           0.0394541881
## indus       -0.0867310441
## chas        -0.9298187135
## nox         -6.7774611223
## rm           0.5824520486
## age         -0.0004376844
## dis         -0.8984261957
## rad          0.4491764668
## tax          0.0050581097
## ptratio     -0.2219848644
## lstat        0.1658805138
## medv        -0.1873594453
```

```r
prediction_ridge <- predict(model_ridge, newdata = testDt)

Ridge <- c("Type" = "10 Fold",
           "R_Square" = R2(prediction_ridge, testDt$crim),
           "RMSE" = RMSE(prediction_ridge, testDt$crim),
           "MAE" = MAE(prediction_ridge, testDt$crim))
```

LASSO Regression

```r
model_lasso <- train(crim ~ ., data = trainDt, method = "glmnet",
                     trControl = train_control,
                     tuneGrid = expand.grid(alpha = 1, lambda = lambda))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
summary(model_lasso)
```

```
##             Length Class      Mode
## a0           77    -none-     numeric
## beta        924    dgCMatrix  S4
## df           77    -none-     numeric
## dim           2    -none-     numeric
## lambda       77    -none-     numeric
## dev.ratio    77    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames       12    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         0    -none-     list
```

```r
coef(model_lasso$finalModel, model_lasso$bestTune$lambda)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 10.22150762
## zn           0.04097843
## indus       -0.04933820
## chas        -0.70294764
## nox         -6.44460113
## rm           0.33561020
## age           .
```

```
## dis         -0.85830616
## rad          0.54947059
## tax          .
## ptratio     -0.22309305
## lstat        0.13018587
## medv        -0.19062597
```

```
prediction_lasso <- predict(model_lasso, newdata = testDt)

LASSO <- c("Type" = "10 Fold",
           "R_Square" = R2(prediction_lasso, testDt$crim),
           "RMSE" = RMSE(prediction_lasso, testDt$crim),
           "MAE" = MAE(prediction_lasso, testDt$crim))

regStats <- rbind(regStats, Ridge, LASSO)
```

A quick linear model

```
model_linear <- train(crim ~ ., data = trainDt,
                      method = "lm",
                      metric = "Rsquared")
coef(model_linear$finalModel)
```

```
##   (Intercept)            zn         indus          chas           nox
##  17.922807744   0.053237621  -0.072227157  -0.844467579 -12.940740846
##            rm           age           dis           rad           tax
##   0.791934168  -0.002455158  -1.290113308   0.619246825  -0.002140583
##       ptratio         lstat          medv
##  -0.400684841   0.142182254  -0.266084003
```

```
prediction_linear <- predict(model_linear, newdata = testDt)
```

Model Comparisons

```
# create data.frames to compare the results from the ridge, LASSO, and linear models
data.frame(
  ridge = as.data.frame.matrix(coef(model_ridge$finalModel, model_ridge$finalModel$lambdaOpt)),
  lasso = as.data.frame.matrix(coef(model_lasso$finalModel, model_lasso$finalModel$lambdaOpt)),
  linear = (model_linear$finalModel$coefficients)
)
```

```
##                       s1        s1.1        linear
## (Intercept)  7.8457780777 10.22150762  17.922807744
## zn           0.0394541881  0.04097843   0.053237621
## indus       -0.0867310441 -0.04933820  -0.072227157
## chas        -0.9298187135 -0.70294764  -0.844467579
## nox         -6.7774611223 -6.44460113 -12.940740846
## rm           0.5824520486  0.33561020   0.791934168
```

```
## age         -0.0004376844  0.00000000  -0.002455158
## dis         -0.8984261957 -0.85830616  -1.290113308
## rad          0.4491764668  0.54947059   0.619246825
## tax          0.0050581097  0.00000000  -0.002140583
## ptratio     -0.2219848644 -0.22309305  -0.400684841
## lstat        0.1658805138  0.13018587   0.142182254
## medv        -0.1873594453 -0.19062597  -0.266084003
```

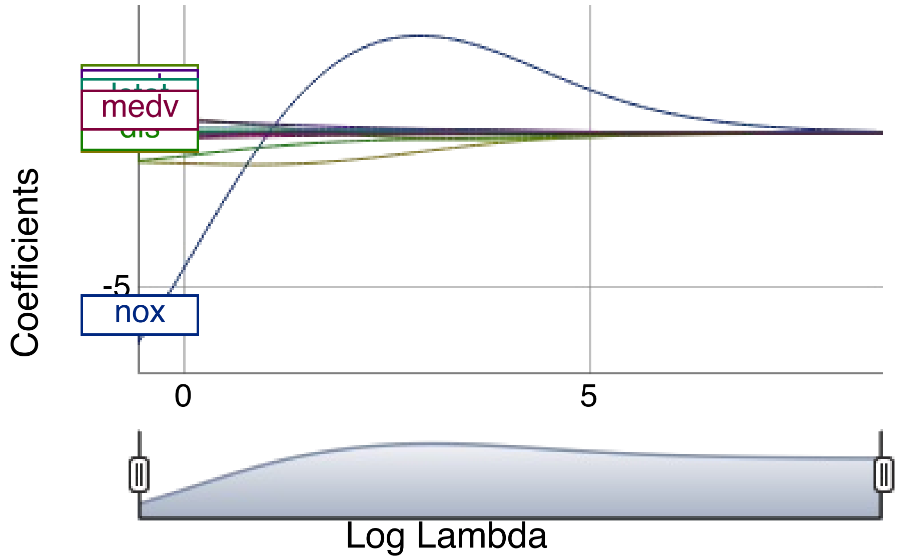
```
data.frame(
  ridge = as.data.frame.matrix(coef(model_ridge$finalModel, model_ridge$finalModel$lambdaOpt)),
  lasso = as.data.frame.matrix(coef(model_lasso$finalModel, model_lasso$finalModel$lambdaOpt)),
  linear = (model_linear$finalModel$coefficients)
) %>%
rename(ridge = s1, lasso = s1.1)
```

```
##                     ridge       lasso        linear
## (Intercept)  7.8457780777 10.22150762  17.922807744
## zn           0.0394541881  0.04097843   0.053237621
## indus       -0.0867310441 -0.04933820  -0.072227157
## chas        -0.9298187135 -0.70294764  -0.844467579
## nox         -6.7774611223 -6.44460113 -12.940740846
## rm           0.5824520486  0.33561020   0.791934168
## age         -0.0004376844  0.00000000  -0.002455158
## dis         -0.8984261957 -0.85830616  -1.290113308
## rad          0.4491764668  0.54947059   0.619246825
## tax          0.0050581097  0.00000000  -0.002140583
## ptratio     -0.2219848644 -0.22309305  -0.400684841
## lstat        0.1658805138  0.13018587   0.142182254
## medv        -0.1873594453 -0.19062597  -0.266084003
```

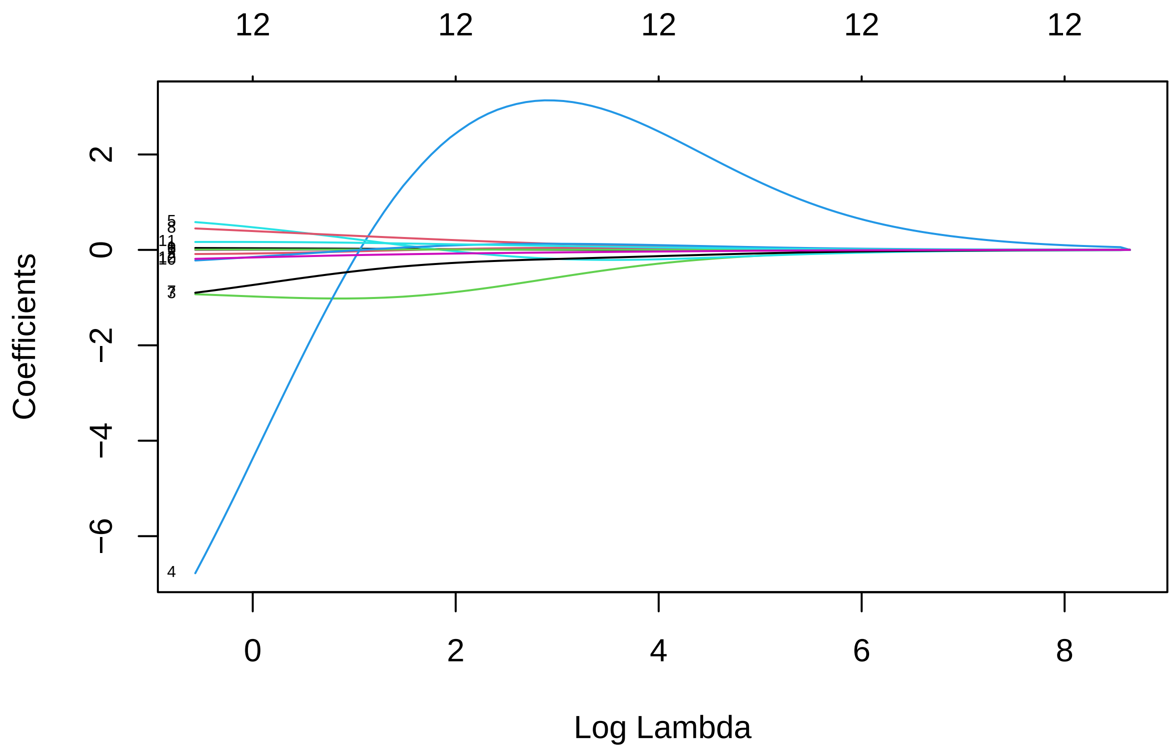
```
c("Ridge_Rsq" = R2(prediction_ridge, testDt$crim),
  "Lasso_Rsq" = R2(prediction_lasso, testDt$crim),
  "Linear_Rsq" = R2(prediction_linear, testDt$crim))
```

```
##  Ridge_Rsq  Lasso_Rsq Linear_Rsq
##  0.6826257  0.6951040  0.6856090
```

```
# chart the coefficients as lambda increases
library(coefplot)

coefpath(model_ridge$finalModel)
```




```
plot(model_ridge$finalModel, xvar = "lambda", label = T)
```



LOOCV

```
# First define the traincontrol to specify k-fold
train_control <- trainControl(method ="LOOCV")

lambda <- 10^seq(-2, 5, length = 1000)
```

Ridge Regression

```
model_ridge <- train(crim ~ ., data = trainDt, method = "glmnet",
                     trControl = train_control,
                     tuneGrid = expand.grid(alpha = 0, lambda = lambda))
summary(model_ridge)
```

```
##             Length Class      Mode
## a0           100   -none-     numeric
## beta        1200   dgCMatrix  S4
## df           100   -none-     numeric
## dim            2   -none-     numeric
## lambda       100   -none-     numeric
## dev.ratio    100   -none-     numeric
```

```
## nulldev        1    -none-     numeric
## npasses        1    -none-     numeric
## jerr           1    -none-     numeric
## offset         1    -none-     logical
## call           5    -none-     call
## nobs           1    -none-     numeric
## lambdaOpt      1    -none-     numeric
## xNames        12    -none-     character
## problemType    1    -none-     character
## tuneValue      2    data.frame list
## obsLevels      1    -none-     logical
## param          0    -none-     list
```

```
coef(model_ridge$finalModel, model_ridge$bestTune$lambda)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  7.8457780777
## zn           0.0394541881
## indus       -0.0867310441
## chas        -0.9298187135
## nox         -6.7774611223
## rm           0.5824520486
## age         -0.0004376844
## dis         -0.8984261957
## rad          0.4491764668
## tax          0.0050581097
## ptratio     -0.2219848644
## lstat        0.1658805138
## medv        -0.1873594453
```

```
prediction_ridge <- predict(model_ridge, newdata = testDt)

Ridge <- c("Type" = "LOOCV",
           "R_Square" = R2(prediction_ridge, testDt$crim),
           "RMSE" = RMSE(prediction_ridge, testDt$crim),
           "MAE" = MAE(prediction_ridge, testDt$crim))
```

LASSO Regression

```
model_lasso <- train(crim ~ ., data = trainDt, method = "glmnet",
                     trControl = train_control,
                     tuneGrid = expand.grid(alpha = 1, lambda = lambda))
summary(model_lasso)
```

```
##             Length Class      Mode
## a0           77    -none-     numeric
## beta        924    dgCMatrix  S4
## df           77    -none-     numeric
## dim           2    -none-     numeric
```

```
## lambda       77    -none-     numeric
## dev.ratio    77    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames       12    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         0    -none-     list
```

```
coef(model_lasso$finalModel, model_lasso$bestTune$lambda)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  14.25157753
## zn            0.04702113
## indus        -0.07062848
## chas         -0.76177849
## nox         -10.21282576
## rm            0.57977335
## age           .
## dis          -1.08925741
## rad           0.57151769
## tax           .
## ptratio      -0.32248621
## lstat         0.13731756
## medv         -0.23010282
```

```
prediction_lasso <- predict(model_lasso, newdata = testDt)

LASSO <- c("Type" = "LOOCV",
           "R_Square" = R2(prediction_lasso, testDt$crim),
           "RMSE" = RMSE(prediction_lasso, testDt$crim),
           "MAE" = MAE(prediction_lasso, testDt$crim))

regStats <- rbind(regStats, Ridge, LASSO)
```

A quick linear model

```
model_linear <- train(crim ~ ., data = trainDt,
                      method = "lm",
                      metric = "Rsquared")
coef(model_linear$finalModel)
```

```
##   (Intercept)            zn         indus          chas           nox
##   17.922807744   0.053237621  -0.072227157  -0.844467579 -12.940740846
```

```
##            rm           age           dis           rad           tax
##   0.791934168  -0.002455158  -1.290113308   0.619246825  -0.002140583
##       ptratio         lstat          medv
##  -0.400684841   0.142182254  -0.266084003
```

```
prediction_linear <- predict(model_linear, newdata = testDt)
```

Model Comparisons

```
data.frame(
  ridge = as.data.frame.matrix(coef(model_ridge$finalModel, model_ridge$finalModel$lambdaOpt)),
  lasso = as.data.frame.matrix(coef(model_lasso$finalModel, model_lasso$finalModel$lambdaOpt)),
  linear = (model_linear$finalModel$coefficients)
)
```

```
##                        s1        s1.1        linear
## (Intercept)  7.8457780777  14.25157753  17.922807744
## zn           0.0394541881   0.04702113   0.053237621
## indus       -0.0867310441  -0.07062848  -0.072227157
## chas        -0.9298187135  -0.76177849  -0.844467579
## nox         -6.7774611223 -10.21282576 -12.940740846
## rm           0.5824520486   0.57977335   0.791934168
## age         -0.0004376844   0.00000000  -0.002455158
## dis         -0.8984261957  -1.08925741  -1.290113308
## rad          0.4491764668   0.57151769   0.619246825
## tax          0.0050581097   0.00000000  -0.002140583
## ptratio     -0.2219848644  -0.32248621  -0.400684841
## lstat        0.1658805138   0.13731756   0.142182254
## medv        -0.1873594453  -0.23010282  -0.266084003
```

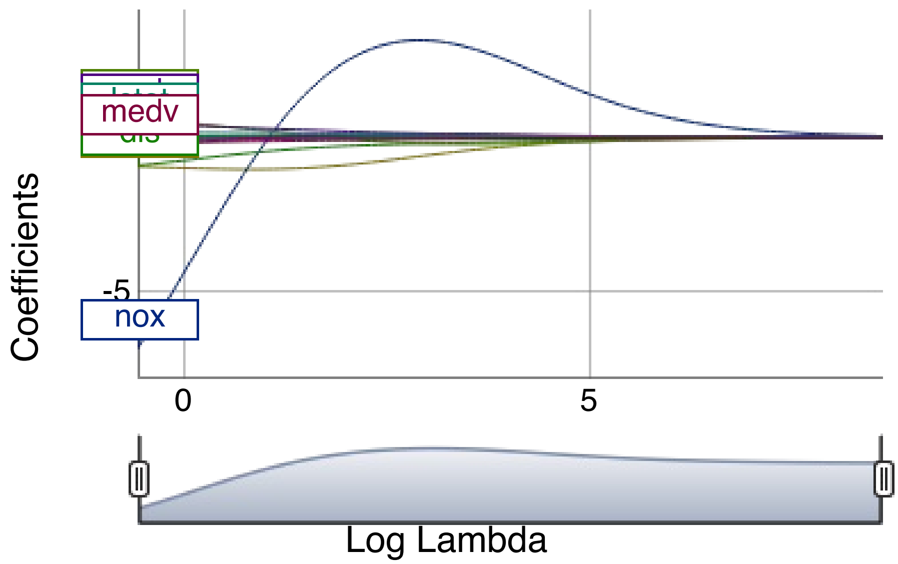
```
data.frame(
  ridge = as.data.frame.matrix(coef(model_ridge$finalModel, model_ridge$finalModel$lambdaOpt)),
  lasso = as.data.frame.matrix(coef(model_lasso$finalModel, model_lasso$finalModel$lambdaOpt)),
  linear = (model_linear$finalModel$coefficients)
) %>%
rename(ridge = s1, lasso = s1.1)
```

```
##                     ridge       lasso        linear
## (Intercept)  7.8457780777  14.25157753  17.922807744
## zn           0.0394541881   0.04702113   0.053237621
## indus       -0.0867310441  -0.07062848  -0.072227157
## chas        -0.9298187135  -0.76177849  -0.844467579
## nox         -6.7774611223 -10.21282576 -12.940740846
## rm           0.5824520486   0.57977335   0.791934168
## age         -0.0004376844   0.00000000  -0.002455158
## dis         -0.8984261957  -1.08925741  -1.290113308
## rad          0.4491764668   0.57151769   0.619246825
## tax          0.0050581097   0.00000000  -0.002140583
## ptratio     -0.2219848644  -0.32248621  -0.400684841
## lstat        0.1658805138   0.13731756   0.142182254
## medv        -0.1873594453  -0.23010282  -0.266084003
```

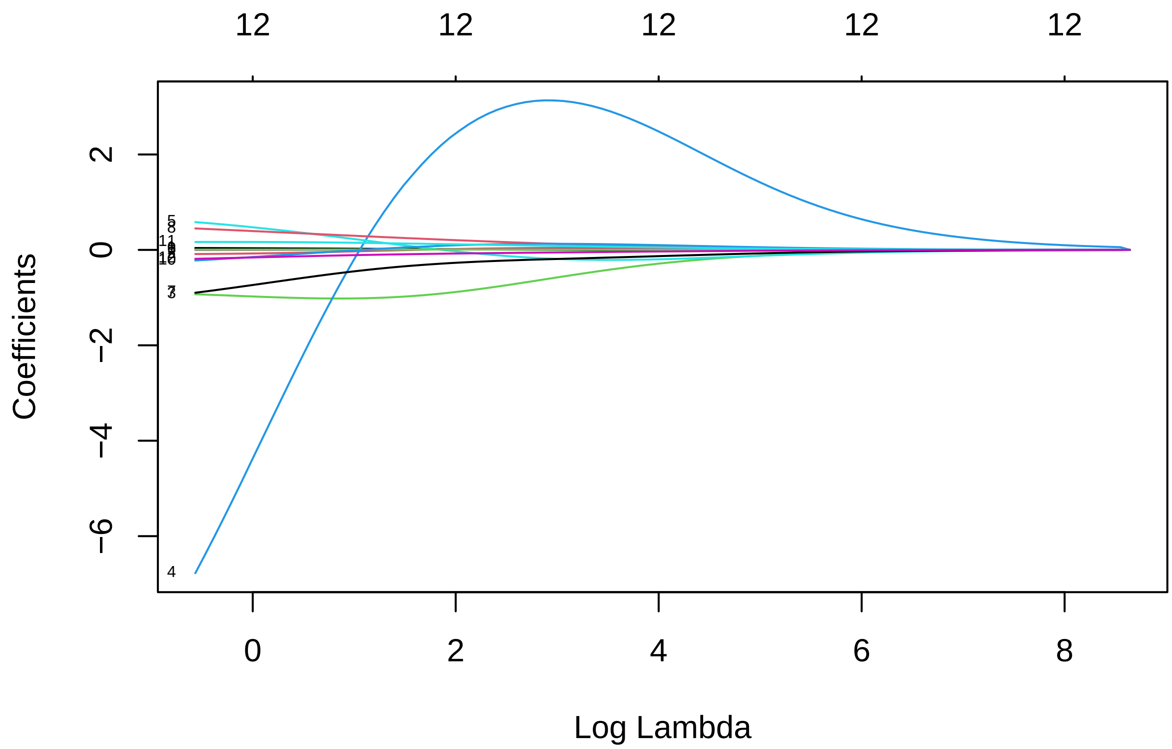
```r
c("Ridge_Rsq" = R2(prediction_ridge, testDt$crim),
  "Lasso_Rsq" = R2(prediction_lasso, testDt$crim),
  "Linear_Rsq" = R2(prediction_linear, testDt$crim))
```

```
##  Ridge_Rsq  Lasso_Rsq Linear_Rsq 
##  0.6826257  0.6899497  0.6856090
```

```r
library(coefplot)

coefpath(model_ridge$finalModel)
```




```r
plot(model_ridge$finalModel, xvar = "lambda", label = T)
```



Closing Thoughts

```r
# convert the stats to an actual data.table and preview them
regStats
```

```
##          Type      R_Square             RMSE                MAE
## Normal   "10 Fold" "0.672415531514889" "3.76250483312505" "2.84888093130358"
## Forward  "10 Fold" "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Backward "10 Fold" "0.676189775369158" "3.62179308014776" "2.61017890603348"
## Dewey    "10 Fold" "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Normal   "LOOCV"   "0.672415531514889" "3.76250483312505" "2.84888093130358"
## Forward  "LOOCV"   "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Backward "LOOCV"   "0.676189775369158" "3.62179308014776" "2.61017890603348"
## Dewey    "LOOCV"   "0.688825316227434" "3.41610927790962" "2.34648962836827"
## Ridge    "10 Fold" "0.682625748162246" "3.57006978669577" "2.67489924552463"
## LASSO    "10 Fold" "0.695104028728848" "3.5079189807032"  "2.57309885980009"
## Ridge    "LOOCV"   "0.682625748162246" "3.57006978669577" "2.67489924552463"
## LASSO    "LOOCV"   "0.689949681902693" "3.64562535063843" "2.75537667733455"
```

```r
regStats <- data.table("Model" = names(regStats[,1]), regStats)
regStats[order(-R_Square)]
```

```
##         Model    Type          R_Square             RMSE              MAE
##  1:     LASSO 10 Fold 0.695104028728848  3.5079189807032 2.57309885980009
##  2:     LASSO   LOOCV 0.689949681902693 3.64562535063843 2.75537667733455
##  3:   Forward 10 Fold 0.688825316227434 3.41610927790962 2.34648962836827
##  4:     Dewey 10 Fold 0.688825316227434 3.41610927790962 2.34648962836827
##  5:   Forward   LOOCV 0.688825316227434 3.41610927790962 2.34648962836827
##  6:     Dewey   LOOCV 0.688825316227434 3.41610927790962 2.34648962836827
##  7:     Ridge 10 Fold 0.682625748162246 3.57006978669577 2.67489924552463
##  8:     Ridge   LOOCV 0.682625748162246 3.57006978669577 2.67489924552463
##  9: Backward 10 Fold 0.676189775369158 3.62179308014776 2.61017890603348
## 10: Backward   LOOCV 0.676189775369158 3.62179308014776 2.61017890603348
## 11:   Normal 10 Fold 0.672415531514889 3.76250483312505 2.84888093130358
## 12:   Normal   LOOCV 0.672415531514889 3.76250483312505 2.84888093130358
```

LASSO regression with both 10-fold and LOOCV produced the best results. I did notice that they each produced a different λ value which I found mildly interesting. I'm assuming that this is largely just due to differences in how the model was developed. I'm pleasantly surprised to find my subset method ranked tied for third place with the forward subsets. I'm a little surprised that both models had the same R^2, $RMSE$, and MAE for both 10-fold and LOOCV. I am surprised to find Ridge towards the bottom of the list, but not at all surprised that backwards and normal subsets did even worse.