4.13

Greg Matesi

July 5, 2020

Load the data

```r
options(digits=4)

# MASS library for the Boston data and lda()
library(MASS)

attach(Boston)

# class library for knn()
library(class)

# Initial look at the data
dim(Boston)
```

```
## [1] 506  14
```

```r
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age   dis rad tax ptratio black lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.090   1 296    15.3 396.9  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.967   2 242    17.8 396.9  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.967   2 242    17.8 392.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.062   3 222    18.7 394.6  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.062   3 222    18.7 396.9  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.062   3 222    18.7 394.1  5.21 28.7
```

```r
summary(crim)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.01    0.08    0.26    3.61    3.68   88.98
```

Make dummy variable

Make a dummy variable called highcrime. This variable will be “TRUE” if crim is greater than it’s median. And it will be “FALSE” if crim is less than it’s median.

```r
med.crim <- median(crim)
Boston$highcrime <- Boston$crim > med.crim

# remove the crim variable. It has been replaced with highcrime
Boston <- Boston[,-c(1)]
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 4):
##
##     age, black, chas, dis, indus, lstat, medv, nox, ptratio, rad, rm,
##     tax, zn
```

```r
names(Boston)
```

```
##  [1] "zn"        "indus"     "chas"      "nox"       "rm"        "age"
##  [7] "dis"       "rad"       "tax"       "ptratio"   "black"     "lstat"
## [13] "medv"      "highcrime"
```

```r
dim(Boston)
```

```
## [1] 506  14
```

```r
# there should be an equal amount of observations above and below the mean
summary(Boston$highcrime)
```

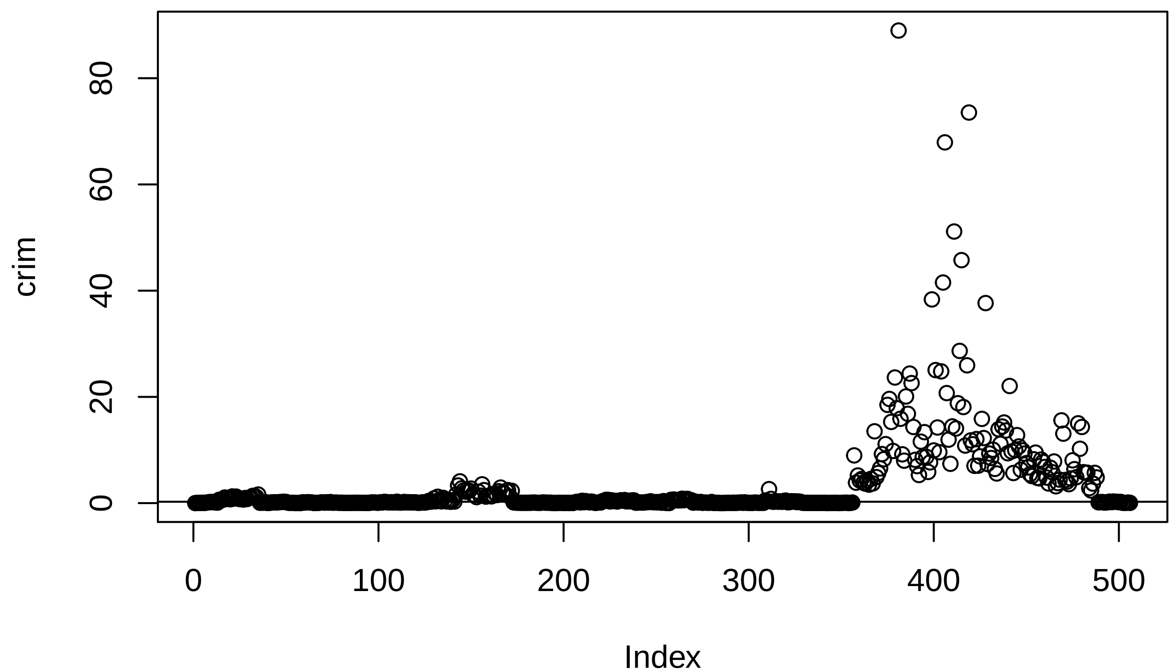
```
##    Mode   FALSE    TRUE
## logical     253     253
```

```r
# Looking at the highcrime variable.
cor(Boston)
```

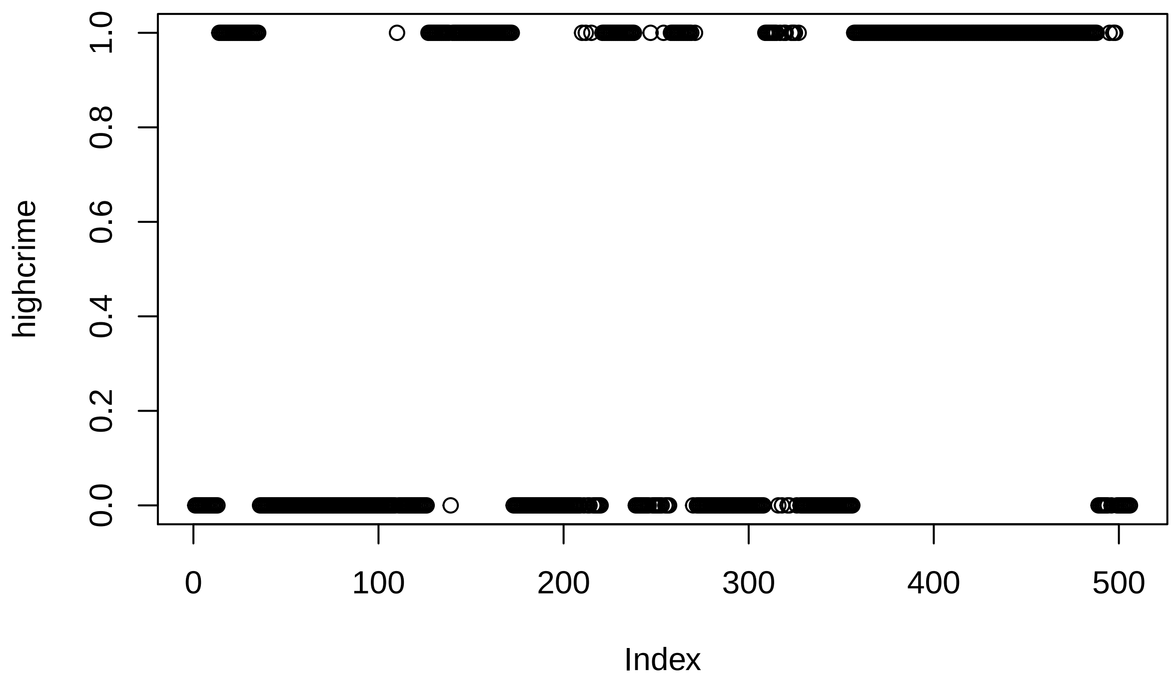
```
##                zn     indus      chas     nox       rm      age      dis
## zn         1.0000 -0.53383 -0.042697 -0.5166  0.31199 -0.56954  0.66441
## indus     -0.5338  1.00000  0.062938  0.7637 -0.39168  0.64478 -0.70803
## chas      -0.0427  0.06294  1.000000  0.0912  0.09125  0.08652 -0.09918
## nox       -0.5166  0.76365  0.091203  1.0000 -0.30219  0.73147 -0.76923
## rm         0.3120 -0.39168  0.091251 -0.3022  1.00000 -0.24026  0.20525
## age       -0.5695  0.64478  0.086518  0.7315 -0.24026  1.00000 -0.74788
## dis        0.6644 -0.70803 -0.099176 -0.7692  0.20525 -0.74788  1.00000
## rad       -0.3119  0.59513 -0.007368  0.6114 -0.20985  0.45602 -0.49459
## tax       -0.3146  0.72076 -0.035587  0.6680 -0.29205  0.50646 -0.53443
## ptratio   -0.3917  0.38325 -0.121515  0.1889 -0.35550  0.26152 -0.23247
## black      0.1755 -0.35698  0.048788 -0.3801  0.12807 -0.27353  0.29151
## lstat     -0.4130  0.60380 -0.053929  0.5909 -0.61381  0.60234 -0.49700
## medv       0.3604 -0.48373  0.175260 -0.4273  0.69536 -0.37695  0.24993
## highcrime -0.4362  0.60326  0.070097  0.7232 -0.15637  0.61394 -0.61634
##                 rad      tax ptratio    black    lstat    medv highcrime
## zn        -0.311948 -0.31456 -0.3917  0.17552 -0.41299  0.3604   -0.4362
## indus      0.595129  0.72076  0.3832 -0.35698  0.60380 -0.4837    0.6033
## chas      -0.007368 -0.03559 -0.1215  0.04879 -0.05393  0.1753    0.0701
```

```
## nox        0.611441  0.66802  0.1889 -0.38005  0.59088 -0.4273    0.7232
## rm        -0.209847 -0.29205 -0.3555  0.12807 -0.61381  0.6954   -0.1564
## age        0.456022  0.50646  0.2615 -0.27353  0.60234 -0.3770    0.6139
## dis       -0.494588 -0.53443 -0.2325  0.29151 -0.49700  0.2499   -0.6163
## rad        1.000000  0.91023  0.4647 -0.44441  0.48868 -0.3816    0.6198
## tax        0.910228  1.00000  0.4609 -0.44181  0.54399 -0.4685    0.6087
## ptratio    0.464741  0.46085  1.0000 -0.17738  0.37404 -0.5078    0.2536
## black     -0.444413 -0.44181 -0.1774  1.00000 -0.36609  0.3335   -0.3512
## lstat      0.488676  0.54399  0.3740 -0.36609  1.00000 -0.7377    0.4533
## medv      -0.381626 -0.46854 -0.5078  0.33346 -0.73766  1.0000   -0.2630
## highcrime  0.619786  0.60874  0.2536 -0.35121  0.45326 -0.2630    1.0000
```

```r
plot(crim)
abline(h=med.crim)
```




```r
plot(highcrime)
```



Subset

Subset the data into a training set and a test set. The variable subsetfrac can be adjusted to allow for a larger or smaller fraction to be used as a test set.

```
subsetfrac <- 1/3
train.index <- sample(dim(Boston)[1], (1-subsetfrac)*dim(Boston)[1] )

Boston.test <- Boston[-c(train.index),]
Boston.train <- Boston[c(train.index),]

highcrime.test <- highcrime[-c(train.index)]
```

Logistic

Fit a logistic regression model using nox, ptratio, chas, and indus as variables.

```
glm.fit <- glm(highcrime~nox+ptratio+chas+indus,
               data = Boston.train,
               family = binomial)

coef(glm.fit)
```

```
## (Intercept)         nox     ptratio        chas       indus
##   -19.76394    34.45769     0.09389     0.97389    -0.06036
```

```r
summary(glm.fit)
```

```
##
## Call:
## glm(formula = highcrime ~ nox + ptratio + chas + indus, family = binomial,
##     data = Boston.train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.233  -0.376  -0.120   0.415   2.647
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.7639     3.0335   -6.52  7.3e-11 ***
## nox          34.4577     5.1486    6.69  2.2e-11 ***
## ptratio       0.0939     0.0933    1.01     0.31
## chas          0.9739     0.6568    1.48     0.14
## indus        -0.0604     0.0450   -1.34     0.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 467.18  on 336  degrees of freedom
## Residual deviance: 207.18  on 332  degrees of freedom
## AIC: 217.2
##
## Number of Fisher Scoring iterations: 7
```

Predict the probability that a neighborhood will have a high crime rate.

```r
glm.probs <- predict(glm.fit,
                     newdata = Boston.test,
                     type = "response")
glm.probs[1:10]
```

```
##       4       5       9      10      12      14      15      16      17      19
## 0.08643 0.08643 0.31953 0.31953 0.31953 0.56335 0.56335 0.56335 0.56335 0.56335
```

Predict whether a neighborhood in the test set will have a high crime rate probability of greater or less than 50%

```r
probability <- .5
glm.pred <- rep(FALSE, length(highcrime.test))
glm.pred[glm.probs > probability]=TRUE
```

Assess the model.

```
glm.table <- table(glm.pred,highcrime.test)
glm.table
```

```
##          highcrime.test
## glm.pred FALSE TRUE
##    FALSE    70   12
##    TRUE     14   73
```

```
mean(glm.pred==highcrime.test)
```

```
## [1] 0.8462
```

```
(glm.table[1,1]+glm.table[2,2]) / length(highcrime.test)
```

```
## [1] 0.8462
```

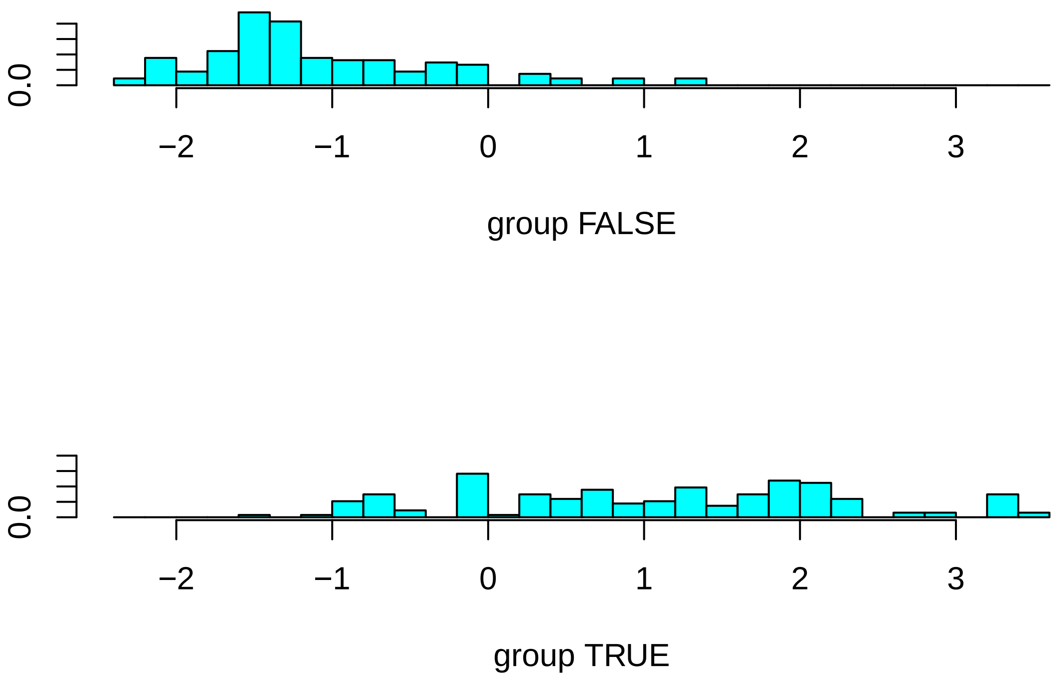
LDA

Fit linear discriminate analysis model on the Boston data.

```
lda.fit <- lda(highcrime~nox+ptratio+chas+indus,
               data=Boston,
               subset=train.index)
lda.fit
```

```
## Call:
## lda(highcrime ~ nox + ptratio + chas + indus, data = Boston,
##     subset = train.index)
##
## Prior probabilities of groups:
##  FALSE   TRUE
## 0.5015 0.4985
##
## Group means:
##          nox ptratio    chas  indus
## FALSE 0.4716   18.03 0.05325  6.947
## TRUE  0.6388   18.91 0.08333 15.282
##
## Coefficients of linear discriminants:
##              LD1
## nox     10.77092
## ptratio  0.08476
## chas     0.13859
## indus    0.03436
```

```
plot(lda.fit)
```



Use the lda model to predict whether neighborhoods in the test set will have a high crime rate.

```
lda.pred=predict(lda.fit, Boston.test)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class <- lda.pred$class
lda.table <- table(lda.class, highcrime.test)
lda.table
```

```
##          highcrime.test
## lda.class FALSE TRUE
##     FALSE    76   22
##     TRUE      8   63
```

```
mean(lda.class==highcrime.test)
```

```
## [1] 0.8225
```

```
(lda.table[1,1]+lda.table[2,2]) / length(highcrime.test)
```

```
## [1] 0.8225
```

KNN

```
variable.list <- c("nox", "ptratio", "chas", "indus")
train.x <- Boston.train[,variable.list]
test.x  <- Boston.test[,variable.list]
train.highcrime <- cbind(Boston.train$highcrime)
```

```
numk = 1
knn.pred <- knn(train.x,
                test.x,
                train.highcrime,
                k=numk)

knn.table <- table(knn.pred,highcrime.test)
knn.table
```

```
##         highcrime.test
## knn.pred FALSE TRUE
##    FALSE    77    3
##    TRUE      7   82
```

```
(knn.table[1,1]+knn.table[2,2])/length(highcrime.test)
```

```
## [1] 0.9408
```

```
mean(knn.pred == highcrime.test)
```

```
## [1] 0.9408
```

Analysis

We see that lda and logistic regression predicts equally well on the test set. Each of these two models predict correctly for ~85% of the observations. The third model, k nearest neighbors with k=1 outperforms them in this respect. It predicts correctly on the test set ~95% of the time.

Sensitivity

The true positive rate of these three methods is examined below.

```
glm.table[2,2]/(glm.table[2,2]+glm.table[1,2])
```

```
## [1] 0.8588
```

```
lda.table[2,2]/(lda.table[2,2]+lda.table[1,2])
```

```
## [1] 0.7412
```

```
knn.table[2,2]/(knn.table[2,2]+knn.table[1,2])
```

```
## [1] 0.9647
```

Knn with k=1 outperforms both of the other two models in sensitivity, the ability to correctly predict a true high crime rate. ## Specificity

The true negative rate for the three models is examined blow.

```
glm.table[1,1]/(glm.table[1,1]+glm.table[2,1])
```

```
## [1] 0.8333
```

```
lda.table[1,1]/(lda.table[1,1]+lda.table[2,1])
```

```
## [1] 0.9048
```

```
knn.table[1,1]/(knn.table[1,1]+knn.table[2,1])
```

```
## [1] 0.9167
```

We see that knn with k=1 also outperforms the other two models in sensitivity. The ability to correctly predict a low crime rate.