Stat 897 Fall 2017 Data Analysis Assignment 9

Penn State

Due October 29, 2017

In this assignment we will use the Boston data found in the MASS library.

1. Fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. At this point in the class, you should feel fairly comfortable making such an open-ended exploration.

Describe your findings, show appropriate results, and determine why some techniques perform better or worse and whic had the best performance.

```
library(MASS)
library(class)
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-12
```

```
library(leaps)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
data("Boston")
attach(Boston)
```

Let's start with parameter selection for the Boston data set. We will use forward selection, lasso and ridge here:

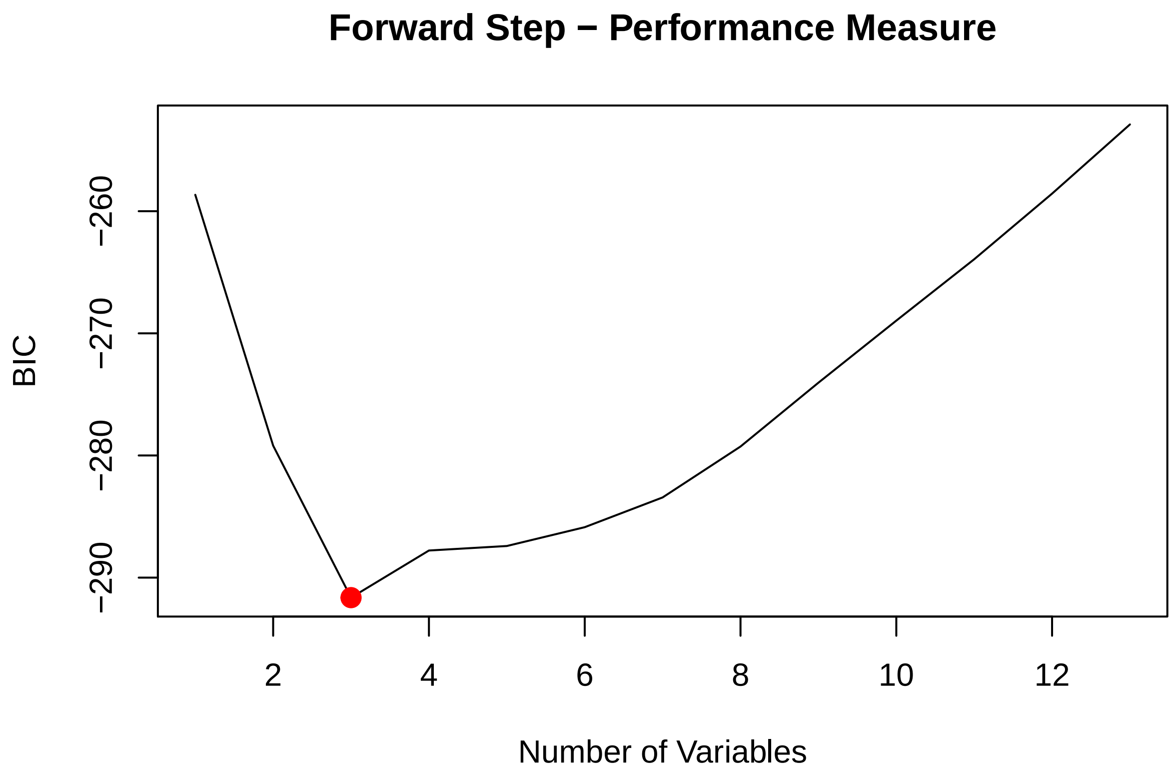
```
x = model.matrix(crim ~ . - 1, data = Boston)
y = Boston$crim

n = nrow(Boston)
p = ncol(Boston) - 1
set.seed (801)
trainingRows=sample (nrow(Boston), n*0.7, replace = FALSE)
train = Boston[trainingRows,]
test = Boston[-trainingRows,]
train.mat <- model.matrix(crim~ ., data = train)
test.mat <- model.matrix(crim~ ., data = test)

# Forward Selection | BIC
regfit.fwd=regsubsets (crim~.,data=train, nvmax =14, method='forward')
reg.summary = summary (regfit.fwd)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = train, nvmax = 14, 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 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"   "*"
```

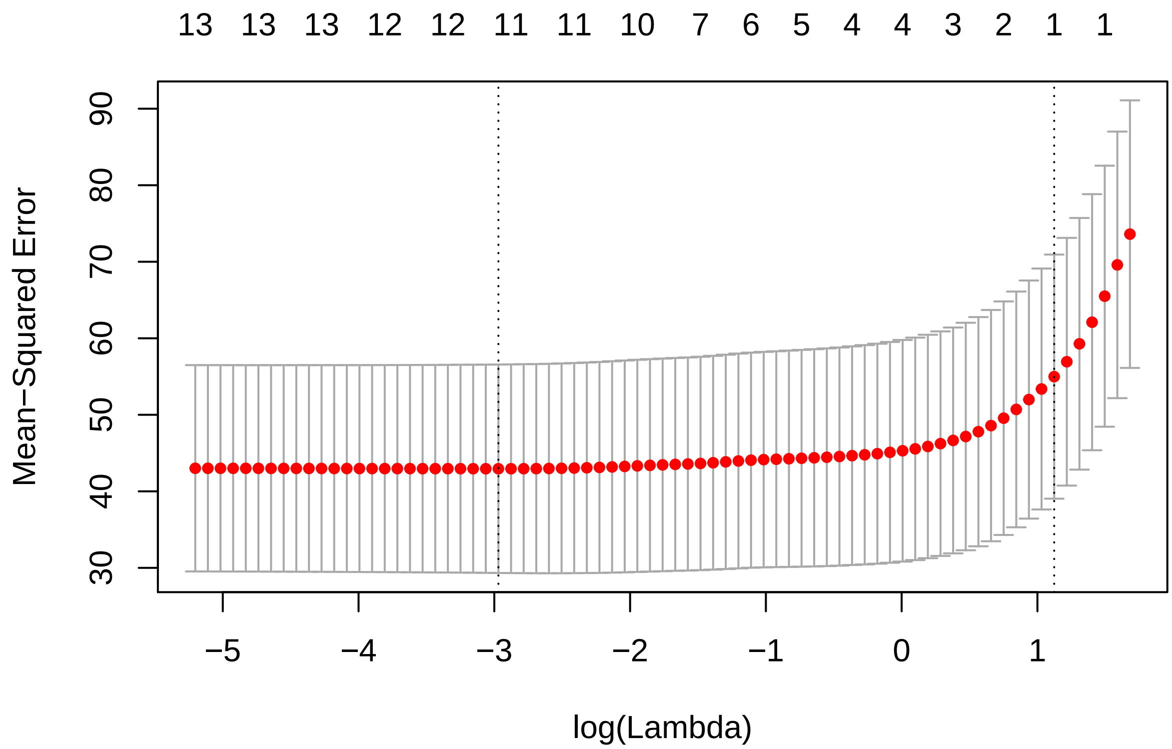
```r
plot(reg.summary$bic, xlab ="Number of Variables",ylab="BIC", type = 'l', main = 'Forward Step - Perfor
which.min (reg.summary$bic )
```

```
## [1] 3
```

```r
points (which.min (reg.summary$bic ), reg.summary$bic[which.min (reg.summary$bic )], col ="red",cex =2,
```




```r
#LASSO
grid =10^ seq (10,-2, length =100)
cv.lasso = cv.glmnet(x, y, type.measure = "mse", nfolds=10)
plot(cv.lasso)
```




```
bestlam.lasso=cv.lasso$lambda.min #find the best tuning parameter

fit.lasso <- glmnet(train.mat, train$crim, alpha = 1, lambda = grid, thresh = 1e-12)
pred.lasso=predict (fit.lasso, s=bestlam.lasso, newx=test.mat)
mean(( pred.lasso - test$crim)^2)
```

```
## [1] 104.9072
```

```
final.lasso=glmnet(x,y,alpha=1) #fit on the entire data set to extract coef
lasso.coef=predict(final.lasso,type="coefficients",s=bestlam.lasso)[1:14,]
lasso.coef
```

```
##   (Intercept)           zn        indus         chas          nox
## 12.666630982  0.036235999 -0.070797081 -0.585263575 -6.966972654
##            rm          age          dis          rad          tax
##   0.229498615  0.000000000 -0.787816119  0.514355222  0.000000000
##       ptratio        black        lstat         medv
## -0.188016174 -0.007548647  0.125278643 -0.157926207
```

```
length(lasso.coef[lasso.coef !=0])
```

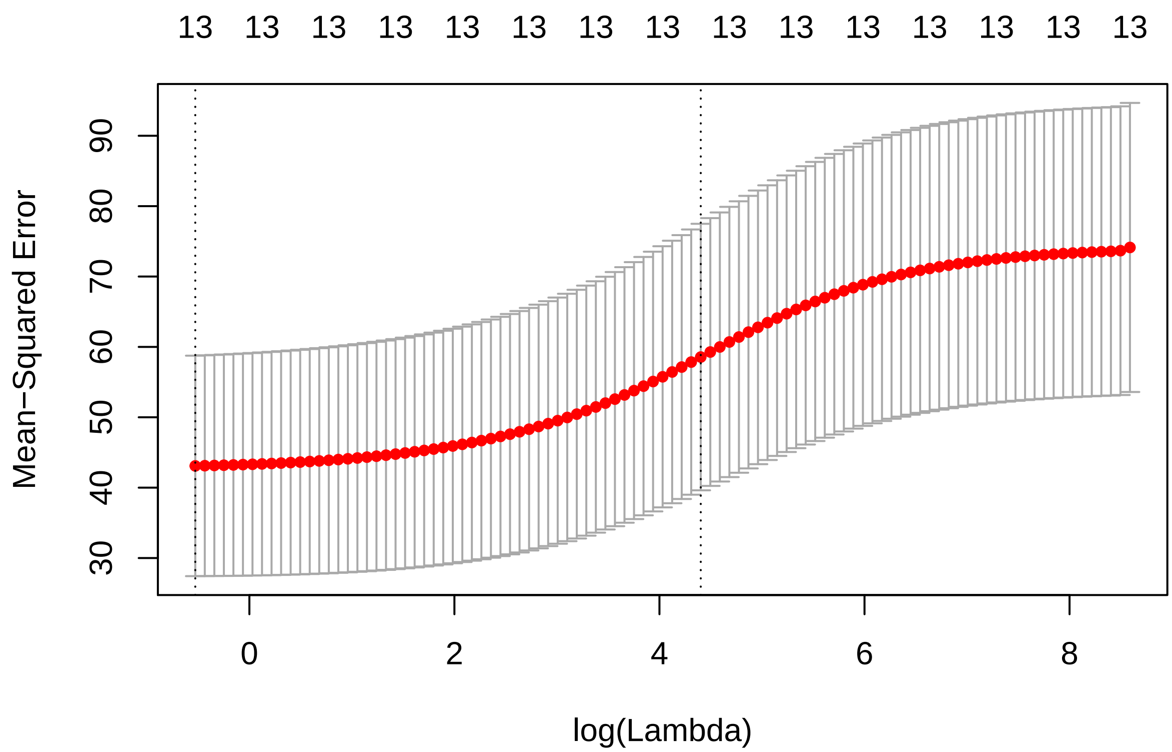
```
## [1] 12
```

```
lasso.coef[lasso.coef!=0] #contains 11 variables in our model
```

```
##   (Intercept)           zn        indus         chas          nox
## 12.666630982  0.036235999 -0.070797081 -0.585263575 -6.966972654
##            rm          dis          rad      ptratio        black
##   0.229498615 -0.787816119  0.514355222 -0.188016174 -0.007548647
```

```
##        lstat         medv
##  0.125278643 -0.157926207
```

```
#Ridge regression
cv.ridge = cv.glmnet(x, y, alpha=0, type.measure = "mse", nfolds=length(y),grouped=FALSE)
plot(cv.ridge)
```




```
bestlam.ridge=cv.ridge$lambda.min #find the best tuning parameter

fit.ridge =glmnet(train.mat, train$crim, alpha = 0, lambda = grid, thresh = 1e-12)
pred.ridge = predict (fit.ridge, s=bestlam.ridge, newx=test.mat)
mean(( pred.ridge - test$crim)^2)
```

```
## [1] 106.0669
```

```
final.ridge=glmnet(x,y,alpha=0) #fit on the full data
ridge.coef=predict(final.ridge,type="coefficients",s=bestlam.ridge)[1:14,]
ridge.coef
```

```
##  (Intercept)           zn        indus         chas          nox
##  8.617905279  0.032352168 -0.081183885 -0.739986141 -5.095661105
##           rm          age          dis          rad          tax
##  0.328170831  0.002074971 -0.683786238  0.414237411  0.003695600
##      ptratio        black        lstat         medv
## -0.127614747 -0.008532788  0.142710654 -0.136308133
```

```
ridge.coef[ridge.coef!=0] #contains all variables in our model
```

```
##  (Intercept)           zn        indus         chas          nox
```

```
##  8.617905279  0.032352168 -0.081183885 -0.739986141 -5.095661105
##           rm          age          dis          rad          tax
##  0.328170831  0.002074971 -0.683786238  0.414237411  0.003695600
##      ptratio        black        lstat         medv
## -0.127614747 -0.008532788  0.142710654 -0.136308133
```

Based on the results above we have the following: Lasso: selects model with 11 variables: zn + indus + chas + nox + rm + dis + rad + ptratio + black + lstat + medv

Forward selection: selects model with 3 variables: rad + black + lstat

Ridge: selects all parameters - we will ignore this as we get smaller models with better Test MSE with Lasso.

```r
# do some data processing to prepare the categorical variable
crim_median <- rep(0, length(crim))
crim_median[crim > median(crim)] <- 1

# add new column to the data frame
Boston <- data.frame(Boston, crim_median)
```

```r
train <- 1:(length(crim) * 0.7)
test <- (length(train)+ 1):length(crim)
Boston.train <- Boston[train, ]
Boston.test <- Boston[test, ]
crim_median.test <- crim_median[test]

# Logistic Regression: Model with all parameters
fit.glm <- glm(crim_median ~ . - crim_median - crim, data = Boston, family = binomial, subset = train)
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
conf_matrix = table(pred.glm, crim_median.test)
conf_matrix
```

```
##         crim_median.test
## pred.glm   0   1
##        0   7   0
##        1  10 135
```

```r
cm=confusionMatrix(data = pred.glm, reference = crim_median.test)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.41176471           1.00000000           1.00000000
##       Neg Pred Value            Precision               Recall
##           0.93103448           1.00000000           0.41176471
##                   F1           Prevalence       Detection Rate
##           0.58333333           0.11184211           0.04605263
## Detection Prevalence    Balanced Accuracy
##           0.04605263           0.70588235
```

```r
mean(pred.glm == crim_median.test)
```

```
## [1] 0.9342105
```

We see that the results appear good. However specificity is 41%

```r
# Logistic Regression: Model with parameters selected by lasso
fit.glm <- glm(crim_median ~ zn + indus + chas + nox + rm + dis + rad + ptratio + black + lstat + medv,
```

```r
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim_median.test)
```

```
##         crim_median.test
## pred.glm   0   1
##        0   7   0
##        1  10 135
```

```r
cm=confusionMatrix(data = pred.glm, reference = crim_median.test)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.41176471           1.00000000           1.00000000
##       Neg Pred Value            Precision               Recall
##           0.93103448           1.00000000           0.41176471
##                   F1           Prevalence       Detection Rate
##           0.58333333           0.11184211           0.04605263
## Detection Prevalence    Balanced Accuracy
##           0.04605263           0.70588235
```

```r
mean(pred.glm == crim_median.test)
```

```
## [1] 0.9342105
```

Results similar to when we use all params.

```r
# Logistic Regression: Model with parameters selected by forward selection
fit.glm <- glm(crim_median ~ rad + black + lstat, data = Boston, family = binomial, subset = train)
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim_median.test)
```

```
##         crim_median.test
## pred.glm   0   1
##        0  14   2
##        1   3 133
```

```r
cm_lr=confusionMatrix(data = pred.glm, reference = crim_median.test)
cm_lr$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.82352941           0.98518519           0.87500000
##       Neg Pred Value            Precision               Recall
##           0.97794118           0.87500000           0.82352941
##                   F1           Prevalence       Detection Rate
##           0.84848485           0.11184211           0.09210526
## Detection Prevalence    Balanced Accuracy
##           0.10526316           0.90435730
```

```r
mean(pred.glm == crim_median.test)
```

```
## [1] 0.9671053
```

Better results and model is simpler. Moving to LDA:

```r
# LDA: Model with all parameters
fit.lda <- lda(crim_median ~ . - crim_median  - crim, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##       0   1
##   0   5   0
##   1  12 135
```

```r
mean(pred.lda$class == crim_median.test)
```

```
## [1] 0.9210526
```

```r
cm=confusionMatrix(data = pred.lda$class, reference = crim_median.test)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.29411765           1.00000000           1.00000000
##       Neg Pred Value            Precision               Recall
##           0.91836735           1.00000000           0.29411765
##                   F1           Prevalence       Detection Rate
##           0.45454545           0.11184211           0.03289474
## Detection Prevalence    Balanced Accuracy
##           0.03289474           0.64705882
```

Poor results when we use all params.

```r
# LDA: Model with parameters selected by lasso
fit.lda <- lda(crim_median ~ zn + indus + chas + nox + rm + dis + rad + ptratio + black + lstat + medv,
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##       0   1
##   0   4   0
##   1  13 135
```

```r
mean(pred.lda$class == crim_median.test)
```

```
## [1] 0.9144737
```

```r
cm=confusionMatrix(data = pred.lda$class, reference = crim_median.test)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.23529412           1.00000000           1.00000000
##       Neg Pred Value            Precision               Recall
##           0.91216216           1.00000000           0.23529412
##                   F1           Prevalence       Detection Rate
##           0.38095238           0.11184211           0.02631579
## Detection Prevalence    Balanced Accuracy
##           0.02631579           0.61764706
```

Poor results when we use Lasso params. Logistic Reg performed better.

```r
# LDA: Model with parameters selected by forward selection
fit.lda <- lda(crim_median ~ rad + black + lstat, data = Boston, subset = train)
```

```r
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##       0   1
##   0  13   2
##   1   4 133
```

```r
mean(pred.lda$class == crim_median.test)
```

```
## [1] 0.9605263
```

```r
cm_lda=confusionMatrix(data = pred.lda$class, reference = crim_median.test)
cm_lda$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.76470588           0.98518519           0.86666667
##       Neg Pred Value            Precision               Recall
##           0.97080292           0.86666667           0.76470588
##                   F1           Prevalence       Detection Rate
##           0.81250000           0.11184211           0.08552632
## Detection Prevalence    Balanced Accuracy
##           0.09868421           0.87494553
```

Simplest model performed best with LDA. For both LR and LDA, simplest model performed the best. Between LDA and LR, LR has performed slightly better.

Moving to KNN.

```r
# KNN: Model with all parameters
train.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0 17 93
##        1  0 42
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.3881579
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn   0   1
##        0  12   4
##        1   5 131
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.9407895
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn   0   1
##        0  15 115
##        1   2  20
```

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

```
## [1] 0.2302632
```

When using all params we get the best results when k =10

```
# KNN: Model with parameters selected by lasso
train.X <- cbind(zn, indus, chas, nox, rm, dis, rad, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, dis, rad, ptratio, black, lstat, medv)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0  6 55
##        1 11 80
```

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

```
## [1] 0.5657895
```

```
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn   0   1
##        0   8  25
##        1   9 110
```

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

```
## [1] 0.7763158
```

```
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0 15 81
##        1  2 54
```

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

```
## [1] 0.4539474
```

When using lasso params results are similar. The k=10 mean actually falls.

```
# KNN: Model with parameters selected by forward selection
train.X <- cbind(rad, black, lstat)[train, ]
test.X <- cbind(rad, black, lstat)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0 12 63
##        1  5 72
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.5526316
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0 14 50
##        1  3 85
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.6513158
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##         crim_median.test
## pred.knn  0  1
##        0 17 91
##        1  0 44
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.4013158
```

```r
cm_lr$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.82352941           0.98518519           0.87500000
##       Neg Pred Value            Precision               Recall
##           0.97794118           0.87500000           0.82352941
##                   F1           Prevalence       Detection Rate
##           0.84848485           0.11184211           0.09210526
## Detection Prevalence    Balanced Accuracy
##           0.10526316           0.90435730
```

```r
cm_lda$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.76470588           0.98518519           0.86666667
##       Neg Pred Value            Precision               Recall
##           0.97080292           0.86666667           0.76470588
##                   F1           Prevalence       Detection Rate
##           0.81250000           0.11184211           0.08552632
## Detection Prevalence    Balanced Accuracy
##           0.09868421           0.87494553
```

All in all it seems that the simple Logistic Regression model with parameters selected by forward selection performs best closely followed by LDA model with parameters selected by forward selection.

2. Now repeat the exercise but classify those neighborhoods in the bottom 10% percentile with lowest crime rates. What differences do you notice between this and the previous classification task (hint: look at the confusion matrix)? Why may it be deceiving to only look at misclassification rate? What other measures can you consider?

We will setup the test and train data as described. The difference here will be that test data will only have 0 values for crime above or below median [always below]

```
test = which(Boston$crim <= quantile(Boston$crim, 0.1))
train = which(Boston$crim > quantile(Boston$crim, 0.1))

Boston.train <- Boston[train, ]
Boston.test <- Boston[test, ]
crim_median.test <- crim_median[test]

# Logistic Regression: Model with all parameters
fit.glm <- glm(crim_median ~ . - crim_median - crim, data = Boston, family = binomial, subset = train)
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
conf_matrix = table(pred.glm, crim_median.test)
conf_matrix
```

```
##         crim_median.test
## pred.glm  0
##        0 47
##        1  4
```

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

```
## [1] 0.9215686
```

We can see the problem right here that - the column for true positive is missing.
We classification rate is high and doesn't tell the complete story. We get that on seeing the confusion matrix.

```
# Logistic Regression: Model with parameters selected by lasso
fit.glm <- glm(crim_median ~ zn + indus + chas + nox + rm + dis + rad + ptratio + black + lstat + medv,
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim_median.test)
```

```
##         crim_median.test
## pred.glm  0
##        0 47
##        1  4
```

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

```
## [1] 0.9215686
```

Results similar to when we use all params.

```
# Logistic Regression: Model with parameters selected by forward selection
fit.glm <- glm(crim_median ~ rad + black + lstat, data = Boston, family = binomial, subset = train)
probs <- predict(fit.glm, Boston.test, type = "response")
```

```r
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim_median.test)
```

```
##         crim_median.test
## pred.glm  0
##        0 51
```

```r
mean(pred.glm == crim_median.test)
```

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

This model classifies all test rows correctly and we see that the confusion matrix has a single entry.

```r
# LDA: Model with all parameters
fit.lda <- lda(crim_median ~ . - crim_median  - crim, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##      0
##   0 51
##   1  0
```

```r
mean(pred.lda$class == crim_median.test)
```

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

This model classifies all test rows correctly.

```r
# LDA: Model with parameters selected by lasso
fit.lda <- lda(crim_median ~ zn + indus + chas + nox + rm + dis + rad + ptratio + black + lstat + medv,
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##      0
##   0 51
##   1  0
```

```r
mean(pred.lda$class == crim_median.test)
```

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

This model classifies all test rows correctly.

```r
# LDA: Model with parameters selected by forward selection
fit.lda <- lda(crim_median ~ rad + black + lstat, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim_median.test)
```

```
##    crim_median.test
##      0
##   0 51
##   1  0
```

```r
mean(pred.lda$class == crim_median.test)
```

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

Simplest model also classified everything correctly. For LR simplest model performed the best. For LDA all performed fine.

Moving to KNN.

```
# KNN: Model with all parameters
train.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##        0 51
##        1  0
```

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

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

```
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##        0 51
##        1  0
```

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

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

```
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##        0 51
##        1  0
```

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

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

When using all params all classified correctly for all k.

```
# KNN: Model with parameters selected by lasso
train.X <- cbind(zn, indus, chas, nox, rm, dis, rad, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, dis, rad, ptratio, black, lstat, medv)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##        0 50
##        1  1
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.9803922
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##         0 48
##         1  3
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.9411765
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##         0 51
##         1  0
```

```r
mean(pred.knn == crim_median.test)
```

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

When using lasso params k=100 peformed best. Othe vlaues of k classified a few incorrectly.

```r
# KNN: Model with parameters selected by forward selection
train.X <- cbind(rad, black, lstat)[train, ]
test.X <- cbind(rad, black, lstat)[test, ]
train.crim_median <- crim_median[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim_median, k = 1)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##         0 43
##         1  8
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.8431373
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 10)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##         0 48
##         1  3
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.9411765
```

```r
pred.knn <- knn(train.X, test.X, train.crim_median, k = 100)
table(pred.knn, crim_median.test)
```

```
##          crim_median.test
## pred.knn  0
##        0 49
##        1  2
```

```r
mean(pred.knn == crim_median.test)
```

```
## [1] 0.9607843
```

When using forward selection params all classified incorrectly for all k.