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.

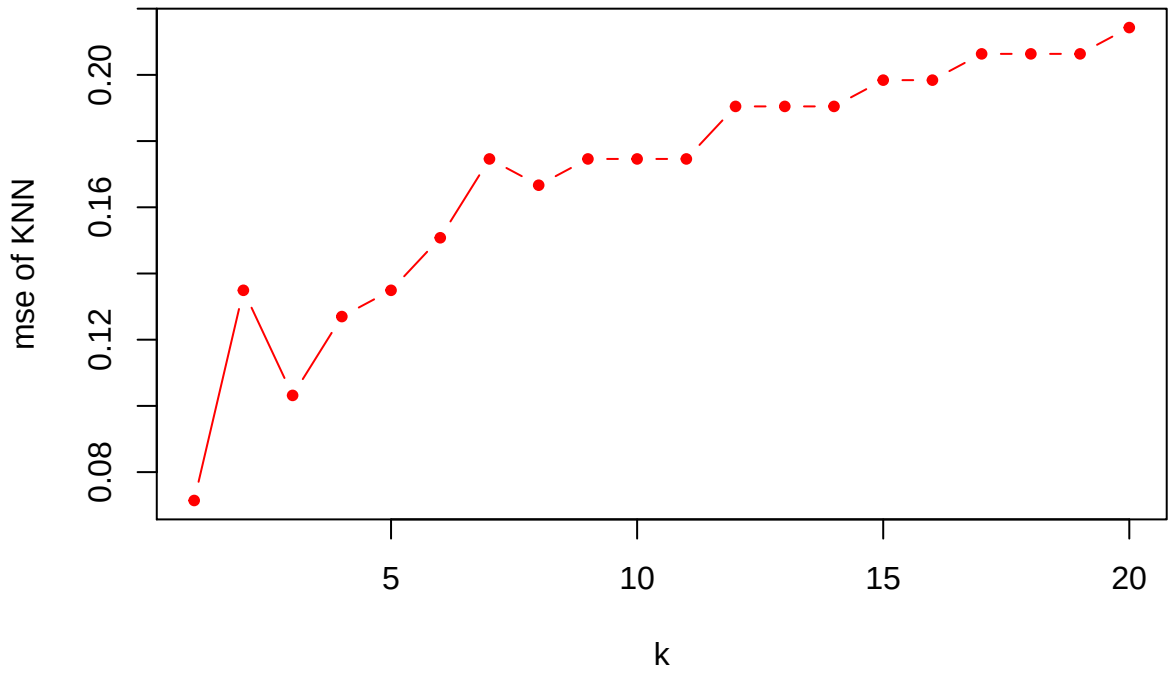
```
suppressWarnings(library(MASS))
library(class)

nobs = dim(Boston)[1] ## number of observations
nvar = dim(Boston)[2] ## number of variables
m.crime = median(Boston$crim)
Bos = Boston
Bos$crim[Boston$crim > m.crime]  = 1 ## crime rate above the median is coded as 1
Bos$crim[Boston$crim <= m.crime] = 0 ## crime rate below the median is coded as 0

set.seed(42)
### 3/4 of the data set is used as the training data set
### the rest is used as the test data
train.idx = sample(seq(1:nobs), round(nobs * 0.75), replace = FALSE)
Bos.train = Bos[ train.idx,]
Bos.test = Bos[-train.idx,]
ytest = Bos.test$crim
ytrain = Bos.train$crim

### Logistic regression
fit.logic = glm(crim ~ ., data = Bos.train, family = 'binomial')
prob.logic = predict(fit.logic, newdata = Bos.test, type = 'response')
pred.logic = 1 * (prob.logic > .5)
mse.logic = mean(abs(ytest - pred.logic))
### misclassification rate of logistic regression
mse.logic
```

```
## [1] 0.1111111
```

```
### LDA
fit.lda = lda(crim ~., data = Bos.train)
pred.lda = as.numeric(predict(fit.lda, newdata = Bos.test)$class) - 1
mse.lda = mean(abs(ytest - pred.lda))
### misclassification rate of LDA
mse.lda
```

```
## [1] 0.1904762
```

```r
### KNN
fit.knn = pred.knn = vector("list", 20)
vmse.knn = rep(NA, 20)
for (k in 1:20){ ## set k from 1 to 20
  fit.knn[[k]] = knn(Bos.train[, -1], Bos.test[, -1], ytrain, k = k)
  pred.knn[[k]] = as.numeric(fit.knn[[k]]) - 1
  vmse.knn[k] = mean(abs(ytest - pred.knn[[k]]))
}
### misclassification rate for each k from 1 to 20
plot(vmse.knn, type = 'b', col = 'red', pch = 20, xlab = 'k', ylab = 'mse of KNN')
```




```r
min(vmse.knn)
```

```
## [1] 0.07142857
```

```r
table(ytest, pred.logic)
```

```
##      pred.logic
## ytest  0  1
##     0 61  5
##     1  9 51
```

```r
table(ytest, pred.lda)
```

```
##      pred.lda
## ytest  0  1
##     0 59  7
##     1 17 43
```

```r
table(ytest, pred.knn[[which.min(vmse.knn)]])
```

```
##
## ytest  0  1
##     0 60  6
##     1  3 57
```

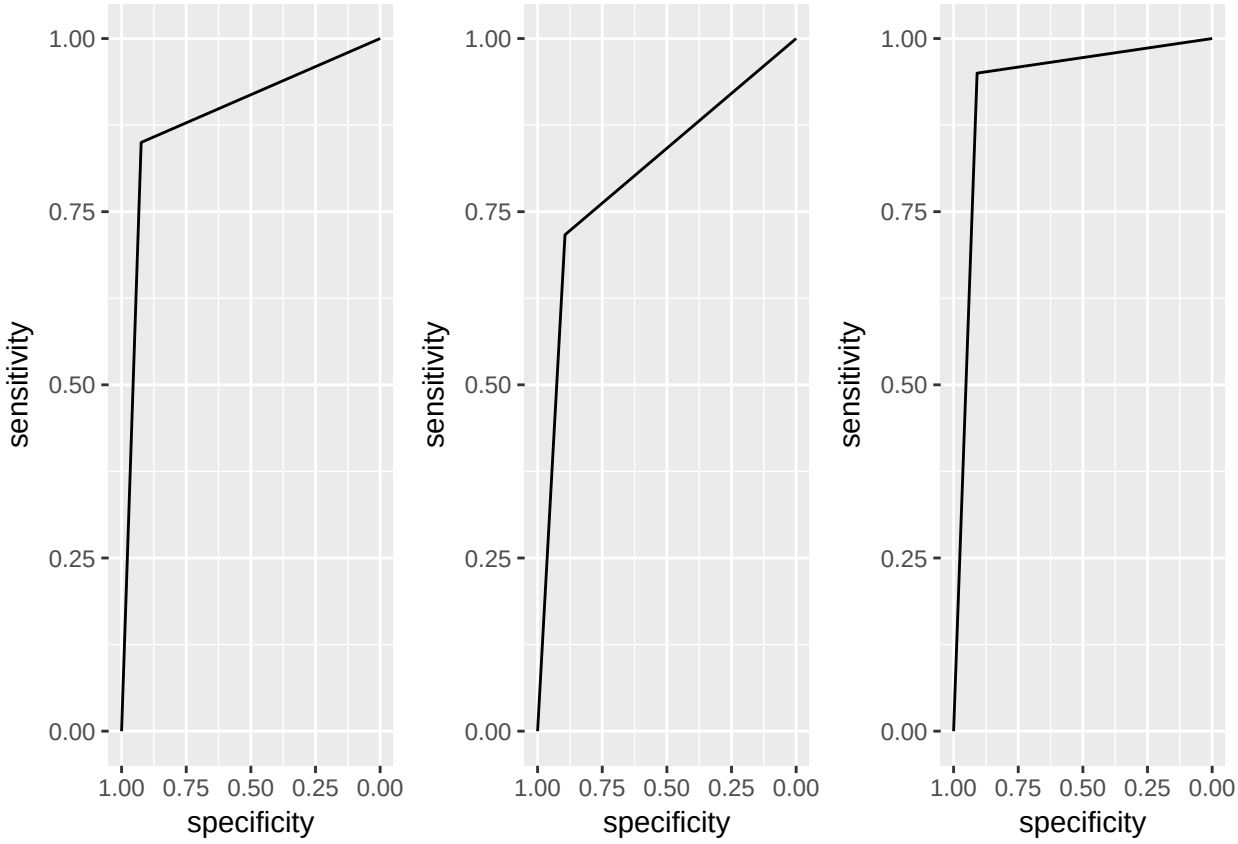
```
suppressWarnings(library(gridExtra))
suppressWarnings(library(pROC, quietly = T))
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
grid.arrange(ggroc(roc(ytest, pred.logic)), ggroc(roc(ytest, pred.lda)),
             ggroc(roc(ytest, pred.knn[[which.min(vmse.knn)]])), ncol = 3, nrow = 1)
```



Remarks

1. There are 506 observations and 14 variables in the data set Boston. About 3/4 of the data set is used as the training data and the rest 1/4 is used as the test data.
2. KNN classification with $k = 3$ has the best performance, but there might be an overfitting issue in this case. Logistic regression has the best performance next to KNN classification with $k = 3$ and the logistic regression should be more robust. LDA has the worst performance, probably because the normal assumption is violated in this case.
3. Dimension reduction techniques such as variable selection and PCA can be further applied to see if there is any improvment of misclassification rate.
4. Training/test or CV can be used to select the number of k in KNN classification.

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?

```
library(MASS)
library(class)

nobs = dim(Boston)[1] ## number of observations
nvar = dim(Boston)[2] ## number of variables
m.crime = quantile(Boston$crim, 0.10)
Bos = Boston
Bos$crim[Boston$crim <= m.crime]  = 1 ## crime rate above the 90% percentile is coded as 1
Bos$crim[Boston$crim > m.crime] = 0 ## crime rate below the 90% percentile is coded as 0

set.seed(623)
### 3/4 of the data set is used as the training data set
### the rest is used as the test data
train.idx = sample(seq(1:nobs), round(nobs * 0.75), replace = FALSE)
Bos.train = Bos[ train.idx,]
Bos.test  = Bos[-train.idx,]
ytest  = Bos.test$crim
ytrain = Bos.train$crim

### Logistic regression
fit.logic = glm(crim ~ ., data = Bos.train, family = 'binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
prob.logic = predict(fit.logic, newdata = Bos.test, type = 'response')
pred.logic = 1 * (prob.logic > .5)
mse.logic = mean(abs(ytest - pred.logic))
### misclassification rate of logistic regression
mse.logic
```

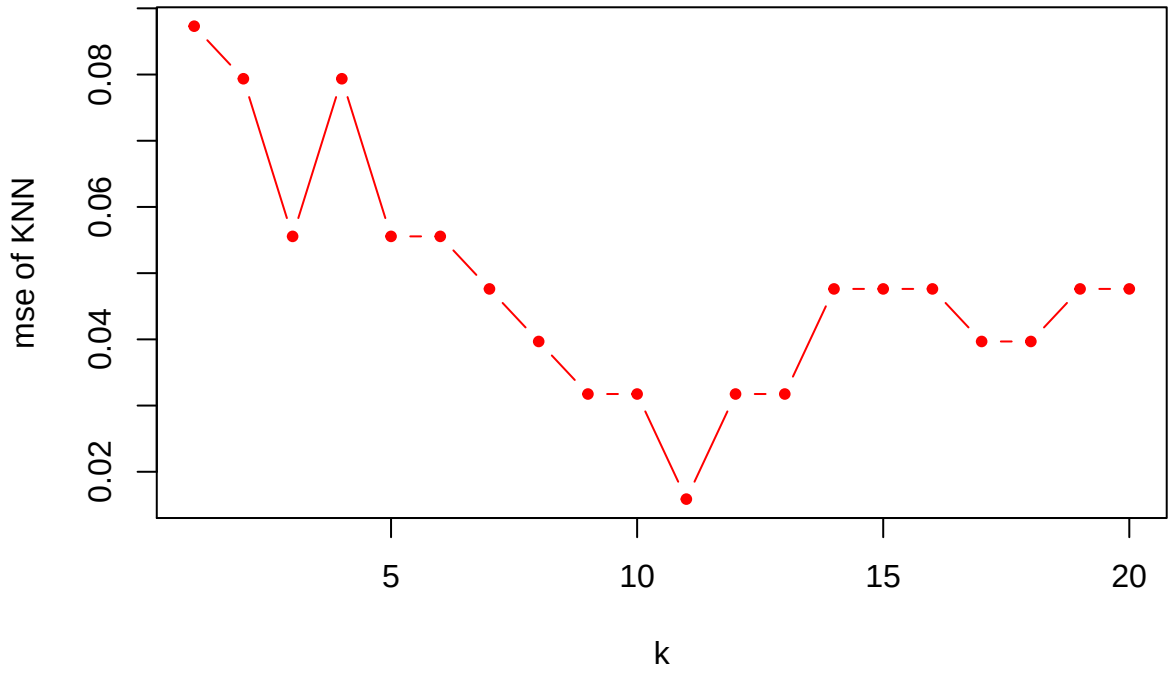
```
## [1] 0.03968254
```

```
### LDA
fit.lda = lda(crim ~., data = Bos.train)
pred.lda  = as.numeric(predict(fit.lda, newdata = Bos.test)$class) - 1
mse.lda = mean(abs(ytest - pred.lda))
### misclassification rate of LDA
mse.lda
```

```
## [1] 0.04761905
```

```
### KNN
### KNN
fit.knn = pred.knn = vector("list", 20)
vmse.knn = rep(NA, 20)
for (k in 1:20){ ## set k from 1 to 20
  fit.knn[[k]] = knn(Bos.train[, -1], Bos.test[, -1], ytrain, k = k)
  pred.knn[[k]] = as.numeric(fit.knn[[k]]) - 1
  vmse.knn[k] = mean(abs(ytest - pred.knn[[k]]))
}
### misclassification rate for each k from 1 to 20
```

```r
plot(vmse.knn, type = 'b', col = 'red', pch = 20, xlab = 'k', ylab = 'mse of KNN')
```




```r
min(vmse.knn)
```

```
## [1] 0.01587302
```

```r
table(ytest, pred.logic)
```

```
##      pred.logic
## ytest   0   1
##     0 116   2
##     1   3   5
```

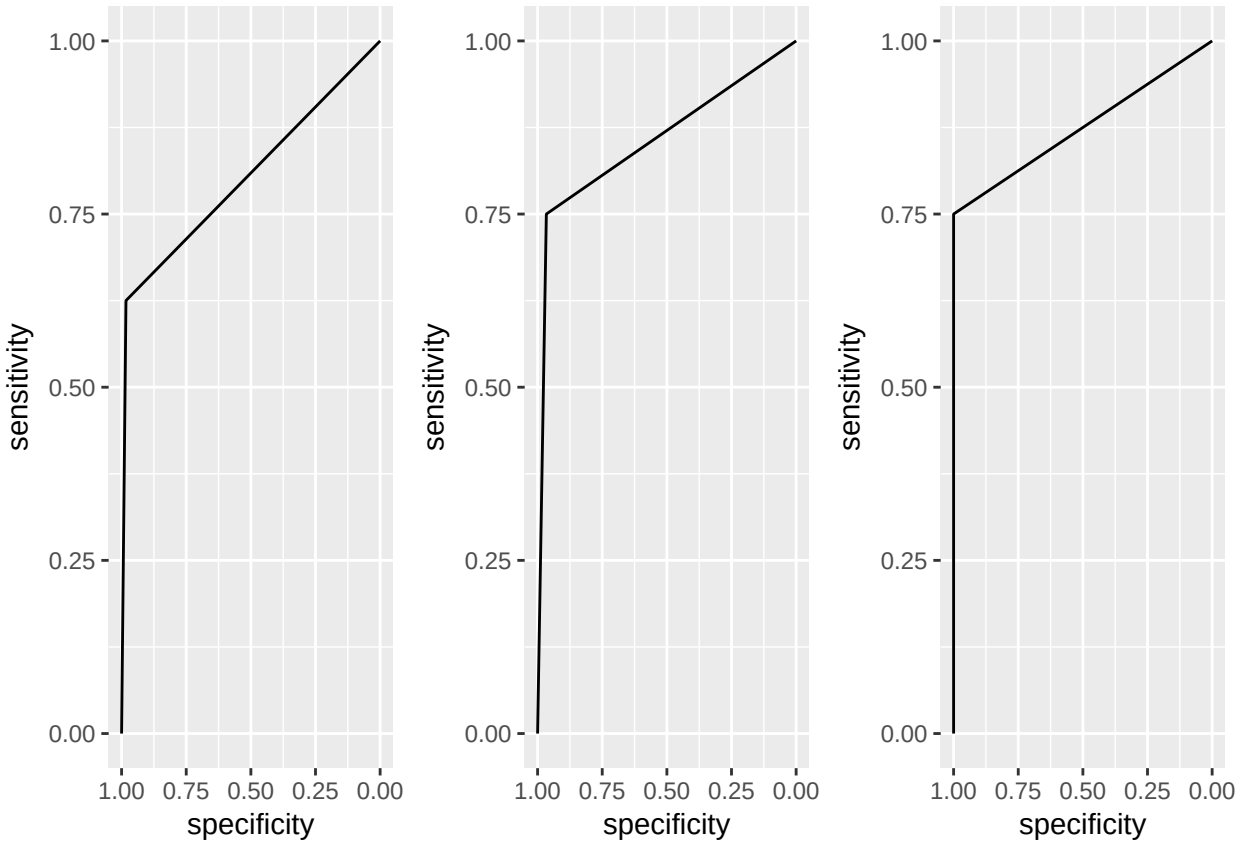
```r
table(ytest, pred.lda)
```

```
##      pred.lda
## ytest   0   1
##     0 114   4
##     1   2   6
```

```r
table(ytest, pred.knn[[which.min(vmse.knn)]])
```

```
##
## ytest   0   1
##     0 118   0
##     1   2   6
```

```r
grid.arrange(ggroc(roc(ytest, pred.logic)), ggroc(roc(ytest, pred.lda)),
             ggroc(roc(ytest, pred.knn[[which.min(vmse.knn)]])), ncol = 3, nrow = 1)
```



Remarks

1. The misclassification rate is lower for all methods, but we see from the confusion matrix this is because there are many more zeros. Our correct classification of 1s is much lower percentage now, and this is confirmed by looking at the ROC curves (except LDA). For KNN, optimal number of clusters is now 11. Generally low incidence classification problems are harder!