STATS 101C - Statistical Models and Data Mining - Homework 3

Darren Tsang, Discussion 4B

Produced on Saturday, Oct. 31 2020 @ 12:05:34 AM

Question 1 (Exercise 5 from Section 4.7)

Part A

QDA will be better on the training set, while LDA will be better on the testing set.

Part B

QDA will be better for both the training and testing set.

Part C

QDA's prediction accuracy should increase relative to LDA's prediction accuracy as n increases because of QDA's flexibility.

Part D

False, QDA's flexibility may actually be a bad thing because there will be overfitting, which will result in a higher testing error compared to LDA.

Question 2 (Exercise 13 from Section 4.7)

```
library(caret)
```

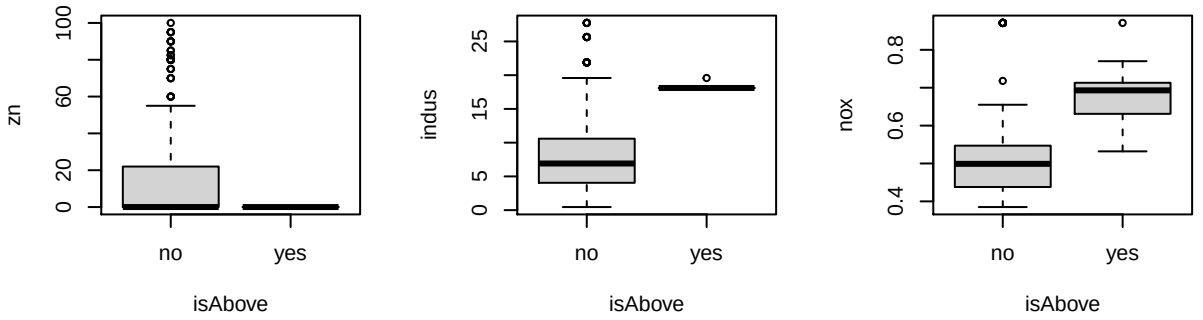
```
## Loading required package: lattice
```

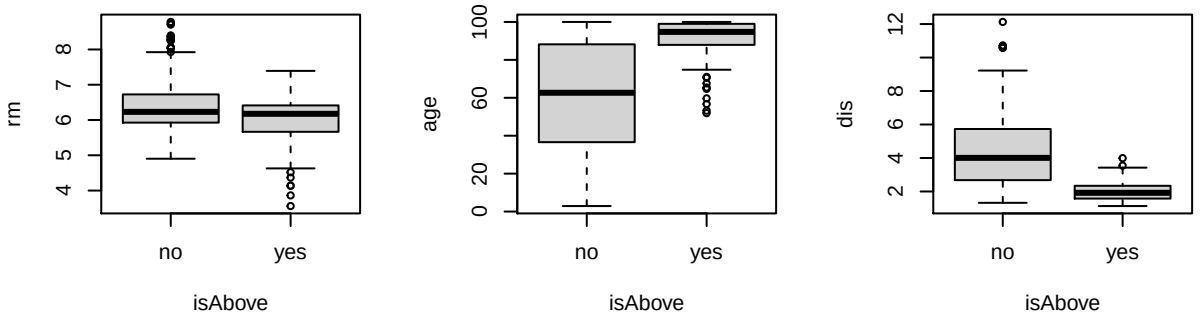
```
## Loading required package: ggplot2
```

```
library(MASS)
data(Boston)

isAbove <- rep("yes", dim(Boston)[1])
isAbove[Boston$crim < mean(Boston$crim)] <- "no"
isAbove <- as.factor(isAbove)
levels(isAbove) <- c("no", "yes")
table(isAbove)
```

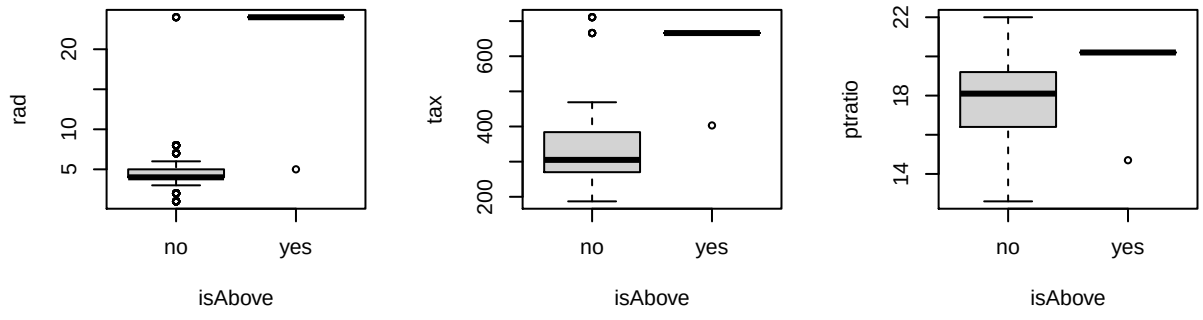
```
## isAbove
##  no yes
## 378 128
```

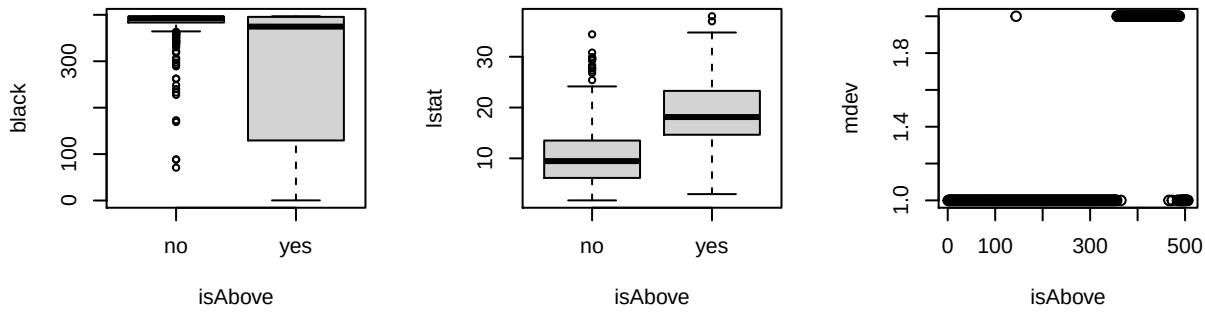
```
Boston <- cbind(Boston, isAbove)

par(mfrow=c(2,3))
plot(Boston$isAbove, Boston$zn, xlab = "isAbove", ylab = "zn")
plot(Boston$isAbove, Boston$indus, xlab = "isAbove", ylab = "indus")
plot(Boston$isAbove, Boston$nox, xlab = "isAbove", ylab = "nox")
plot(Boston$isAbove, Boston$rm, xlab = "isAbove", ylab = "rm")
plot(Boston$isAbove, Boston$age, xlab = "isAbove", ylab = "age")
plot(Boston$isAbove, Boston$dis, xlab = "isAbove", ylab = "dis")
```






```
plot(Boston$isAbove, Boston$rad, xlab = "isAbove", ylab = "rad")
plot(Boston$isAbove, Boston$tax, xlab = "isAbove", ylab = "tax")
plot(Boston$isAbove, Boston$ptratio, xlab = "isAbove", ylab = "ptratio")
plot(Boston$isAbove, Boston$black, xlab = "isAbove", ylab = "black")
plot(Boston$isAbove, Boston$lstat, xlab = "isAbove", ylab = "lstat")
plot(Boston$isAbove, Boston$mdev, xlab = "isAbove", ylab = "mdev")
```





```r
colnames(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
## [15] "isAbove"
```

```r
first_set <- colnames(Boston)[-c(1)]
second_set <- c("indus", "nox", "age", "dis", "rad", "tax", "ptratio", "black", "isAbove")
third_set <- c("rad", "tax", "ptratio", "isAbove")
```

```r
train_control <- trainControl(method="cv", number = 5,
                              classProbs = TRUE,
                              savePredictions = TRUE)
```

Doing logistic regression, LDA, and KNN using first_set (every predictor)

```r
set.seed(1)

LRfit_first <- train(isAbove ~ .,
                     data = Boston[, first_set],
                     method = "glm",
                     family = "binomial",
                     preProc = c("center", "scale"),
                     trControl = train_control)

LDAfit_first <- train(isAbove ~ .,
                      data = Boston[, first_set],
                      method = "lda",
                      family = "binomial",
                      preProc = c("center", "scale"),
                      trControl = train_control)

k_max <- 400
KNNfit_first <- train(isAbove ~ .,
                      data = Boston[, first_set],
                      method = 'knn',
                      preProc = c("center", "scale"),
                      trControl = train_control,
                      tuneGrid = expand.grid(k = seq(1, k_max, by = 1)))
```

Doing logistic regression, LDA, and KNN using second_set

```r
set.seed(1)

LRfit_second <- train(isAbove ~ .,
                      data = Boston[, second_set],
                      method = "glm",
                      family = "binomial",
```

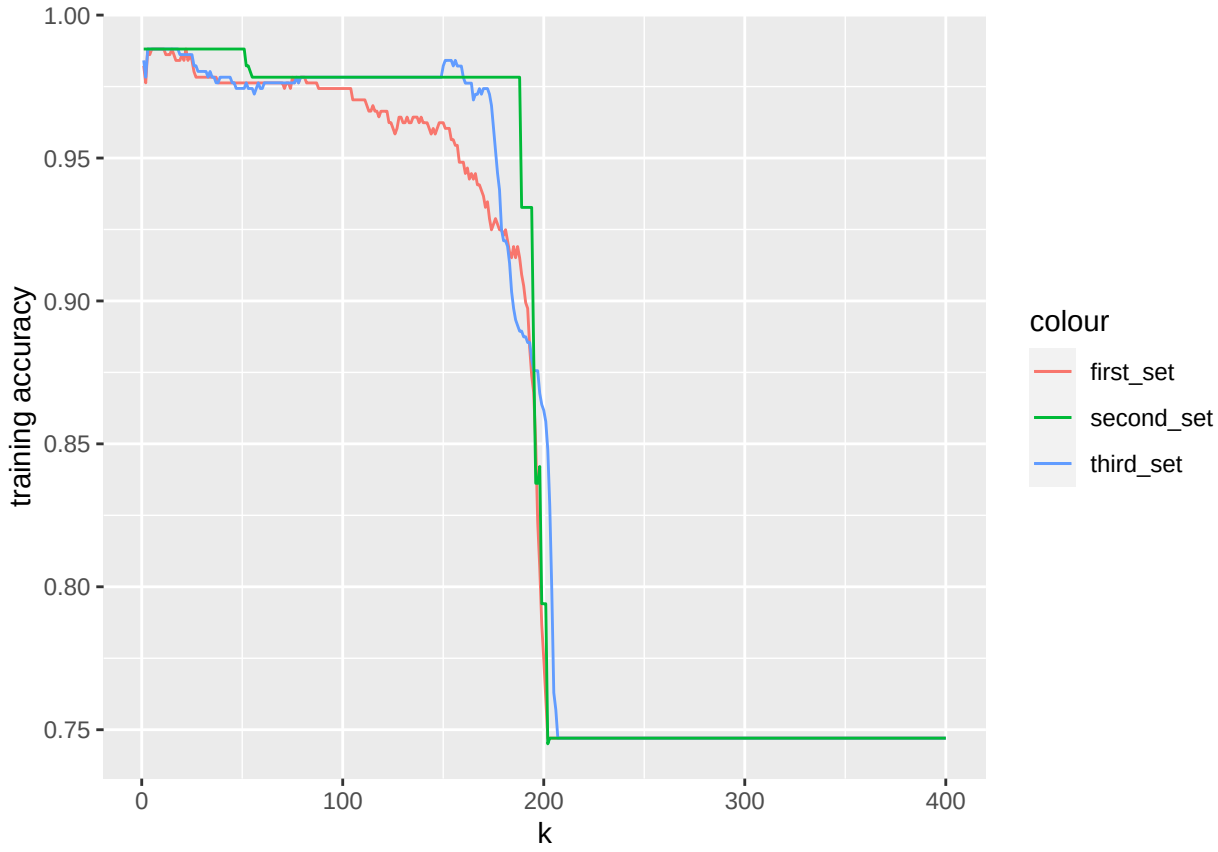
```
                      preProc = c("center", "scale"),
                      trControl = train_control)

LDAfit_second <- train(isAbove ~ .,
                       data = Boston[, second_set],
                       method = "lda",
                       family = "binomial",
                       preProc = c("center", "scale"),
                       trControl = train_control)

KNNfit_second <- train(isAbove ~ .,
                       data = Boston[, second_set],
                       method = 'knn',
                       preProc = c("center", "scale"),
                       trControl = train_control,
                       tuneGrid = expand.grid(k = seq(1, k_max, by = 1)))
```

Doing logistic regression, LDA, and KNN using third_set

```
set.seed(1)

LRfit_third <- train(isAbove ~ .,
                     data = Boston[, third_set],
                     method = "glm",
                     family = "binomial",
                     preProc = c("center", "scale"),
                     trControl = train_control)

LDAfit_third <- train(isAbove ~ .,
                      data = Boston[, third_set],
                      method = "lda",
                      family = "binomial",
                      preProc = c("center", "scale"),
                      trControl = train_control)

KNNfit_third <- train(isAbove ~ .,
                      data = Boston[, third_set],
                      method = 'knn',
                      preProc = c("center", "scale"),
                      trControl = train_control,
                      tuneGrid = expand.grid(k = seq(1, k_max, by = 1)))
```

Here are the results summarized:

```
ggplot(data = NULL ,aes(x = 1:k_max)) +
  geom_line(aes(y = KNNfit_first$results$Accuracy, color = "darkblue")) +
  geom_line(aes(y = KNNfit_second$results$Accuracy, color = "darkred")) +
  geom_line(aes(y = KNNfit_third$results$Accuracy, color = "darkorange")) +
  xlab("k") +
  ylab("training accuracy") +
  scale_color_discrete(labels = c("first_set", "second_set", "third_set"))
```




```
first <- c(LRfit_first$results[,"Accuracy"],
           LDAfit_first$results[,"Accuracy"],
           max(KNNfit_first$results$Accuracy))

second <- c(LRfit_second$results[,"Accuracy"],
            LDAfit_second$results[,"Accuracy"],
            max(KNNfit_second$results$Accuracy))

third <- c(LRfit_third$results[,"Accuracy"],
           LDAfit_third$results[,"Accuracy"],
           max(KNNfit_third$results$Accuracy))

results <- rbind("using first_set" = first,
                 "using second_set" = second,
                 "using third_set" = third)
colnames(results) <- c("LR", "LDA", "kNN")
results
```

```
##                         LR       LDA       kNN
## using first_set  0.9841584 0.9881382 0.9881382
## using second_set 0.9920792 0.9881382 0.9881382
## using third_set  0.9881188 0.9881382 0.9881382
```

```
which(KNNfit_first$results$Accuracy == max(KNNfit_first$results$Accuracy))
```

```
##  [1]  3  5  6  7  8  9 10 11 15 22
```

```
which(KNNfit_second$results$Accuracy == max(KNNfit_second$results$Accuracy))
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
```

```
which(KNNfit_third$results$Accuracy == max(KNNfit_third$results$Accuracy))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51
```

From the table above, we can see that for logistic regression (LR), the second set performs best, but we should point out that the other sets are very, very good as well. It is also interesting to note that for LDA and kNN, the best accuracy is the same no matter which subset we used. We also see that there are many values of k that can reach the best accuracy for all three subsets. Ultimately, I would go with the second subset because it's the best for LR and the other two methods are tied with the other two subsets.

Question 3 (Exercise 8 from Section 5.4)

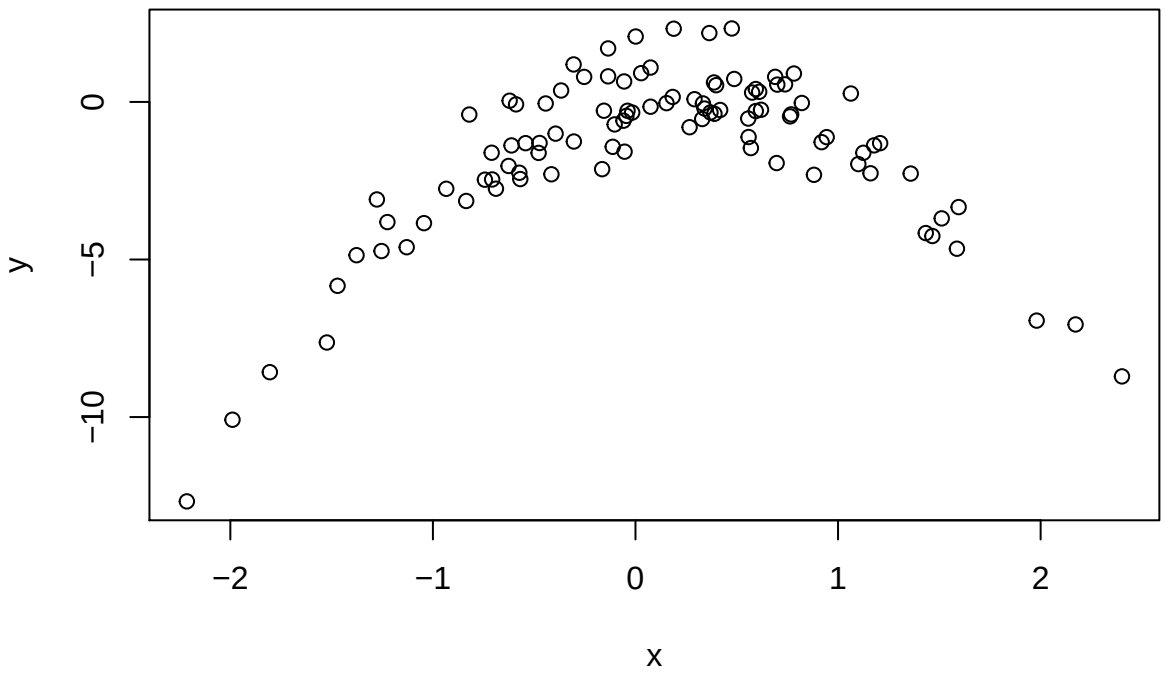
Part A

```
set.seed(1)
x <- rnorm(100)
y <- x - 2*x*x + rnorm(100)
```

n is 100, p is 2. The model used to generate the data is $Y = -2X^2 + X + \epsilon$.

Part B

```
plot(x, y)
```



There seems to be a quadratic relationship between x and y. The curve peak occurs around $x = 0$.

Part C

```
xy <- data.frame(x, y)
```

```
model_fitter <- function(df, deg, seed){
  set.seed(seed)
  rows <- nrow(df)
  error <- rep(NULL, rows)
  for (i in seq(1, rows)){
    curr_lm <- lm(y ~ poly(x, deg), data = df[-i, ])
```

```
    pred <- predict(curr_lm, newdata = df[i, ])
    error[i] <- (pred - df[i, ]$y)^2
  }

  return(mean(error))
}
```

(i) $Y = \beta_0 + \beta_1 X + \epsilon$

```
print(model_fitter(xy, 1, 10))
```

```
## [1] 7.288162
```

(ii) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

```
print(model_fitter(xy, 2, 10))
```

```
## [1] 0.9374236
```

(iii) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

```
print(model_fitter(xy, 3, 10))
```

```
## [1] 0.9566218
```

(iv) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

```
print(model_fitter(xy, 4, 10))
```

```
## [1] 0.9539049
```

Part D

```
print(model_fitter(xy, 1, 100))
```

```
## [1] 7.288162
```

```
print(model_fitter(xy, 2, 100))
```

```
## [1] 0.9374236
```

```
print(model_fitter(xy, 3, 100))
```

```
## [1] 0.9566218
```

```r
print(model_fitter(xy, 4, 100))
```

```
## [1] 0.9539049
```

My results the same as what I got in Part C. This makes sense because LOOCV allows every single observation to serve as validation, so there is no random selection at play.

Part E

We see that $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$ had the lowest error; this is not surprising because as we saw in Part B, the data had a quadratic shape.

Part F

```r
summary(lm(y ~ poly(x, 4), data = xy))
```

```
##
## Call:
## lm(formula = y ~ poly(x, 4), data = xy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0550 -0.6212 -0.1567  0.5952  2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9591 on 95 degrees of freedom
## Multiple R-squared:  0.8753, Adjusted R-squared:  0.8701
## F-statistic: 166.7 on 4 and 95 DF,  p-value: < 2.2e-16
```

We can see that only the intercept, linear term, and quadratic term are statistically significant, which agrees with our results from doing LOOCV.