HW3

Aidan Horn

4/12/2022

Section 4.7, page 169-170, question 5

- a. If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?
    - Since QDA is a more flexible model, we expect it to perform better on the training data than LDA due to overfitting any chance non-linearity in the data. However, if the true decision boundary is linear, then LDA should perform better on the test set.
- b. If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?
    - We should expect that QDA should outperform LDA for both the training data and the testing data due to its higher flexibility. However, some nonlinear relationships will be better approximated by a linear function so we can expect that sometimes LDA will perform better.
- c. In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?
    - As the sample size increases, we should expect the test prediction accuracy of the more flexible model (QDA) should increase relative to the test prediction accuracy of LDA since as the sample size grows, the likelihood of the training data not representing the relationship of the overall data decreases.
- d. True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.
    - False. If the sample size is small, QDA will likely lead to overfitting of the training data leading to a higher test error than LDA. It is very unlikely that, given we know the decision boundary is linear, that a randomly sampled test set would result in a boundary better fit by a quadratic decision boundary.

Section 4.7, page 170, question 8

- Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

- We would prefer Logistic regression. For KNN, the test error is the number of missclassified points based on the classification areas determined from the entire dataset. Therefore, for K = 1, all the train points would be correctly predicted, leading to a 0% error rate. If the average error rate is 18% for KNN, the test error rate must be 36% since (0% + 36%)/2 = 18%. Therefore we get better test performance from Logistic Regression (30%) which indicates this model would perform better with new data.

Section 4.7, page 173, question 13

- This question should be answered using the Weekly data set, which is part of the ISLR2 package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.
    - a. Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.0.5
```

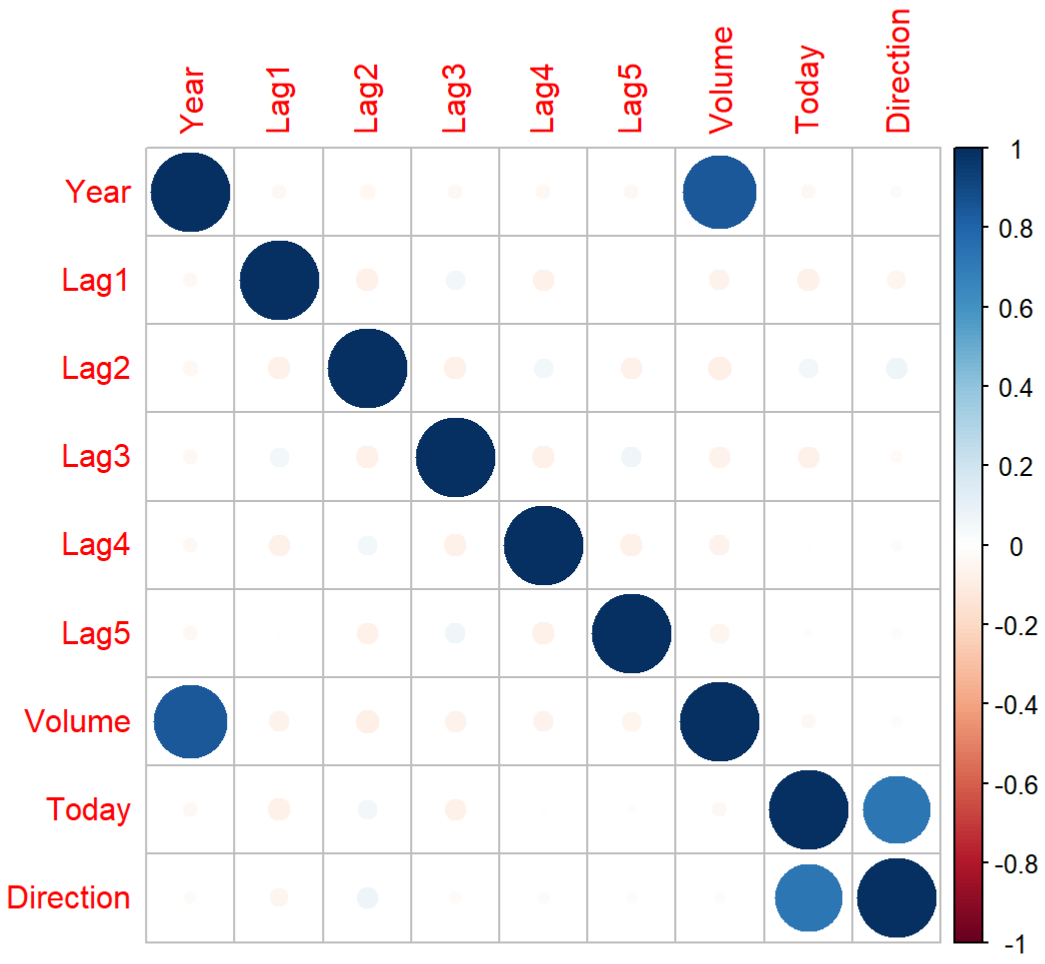
```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

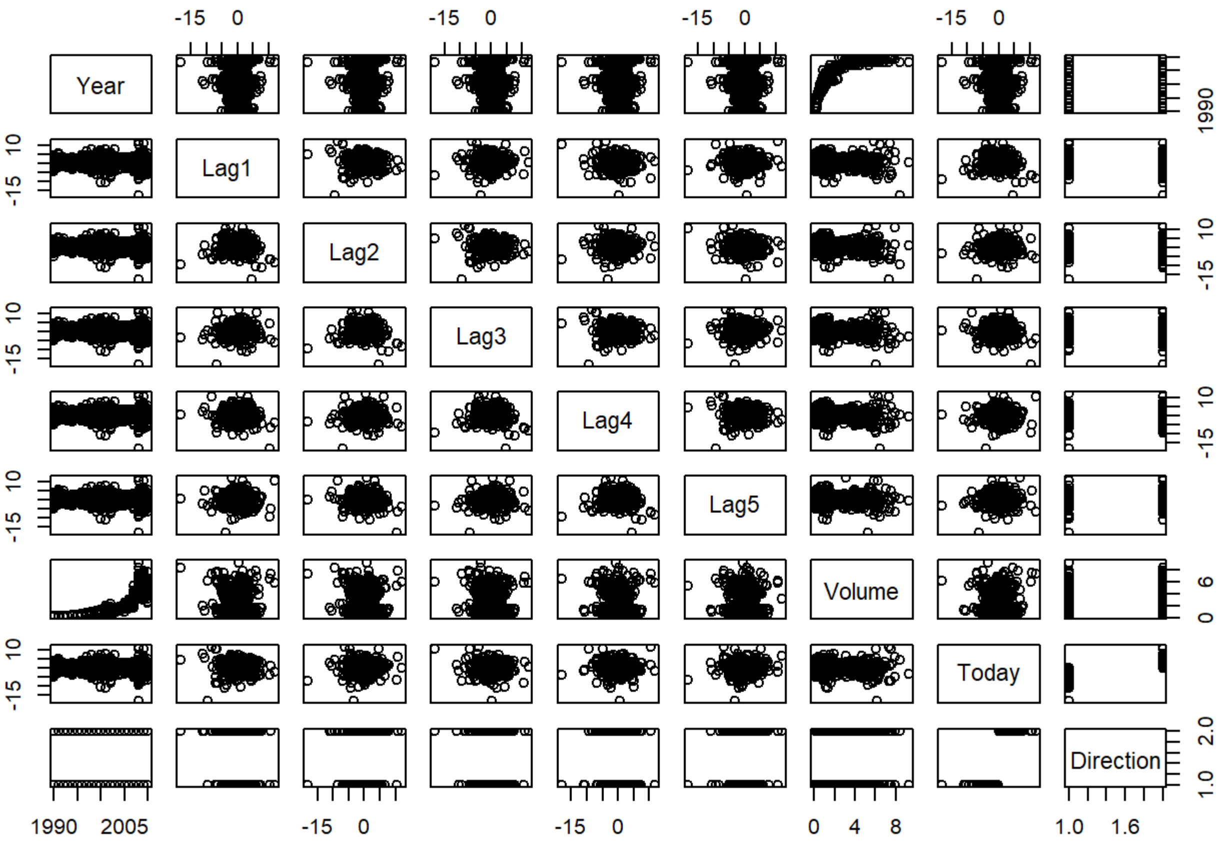
```
attach(Weekly)

data = Weekly

data$Direction = as.numeric(data$Direction)

summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3         
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4               Lag5              Volume            Today         
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
##  Direction 
##  Down:484  
##  Up  :605  
##            
##            
##            
## 
```

```
corrplot(cor(data))
```



```
pairs(Weekly)
```



*

There doesn’t seem to be any strong correlations between any of the lags and the direction of the stock market. The data from Today seems to be the variable with a meaningful correlation with another variable.

- b. Use the full data set to perform a logistic regression withDirection as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = bin
omial)

summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

- Lag 2 seems to be the most significant predictor. Of the 6 predictors, Lag 2 is the only significant predictor.
- c. Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
glm.probs = predict(glm.fit, type = "response")

glm.pred = rep("Down", 1089)

glm.pred[glm.probs > .5] = "Up"

table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
print(paste("Fraction of correct predictions: ", mean(glm.pred == Direction)))
```

```
## [1] "Fraction of correct predictions:  0.561065197428834"
```

- The confusion matrix indicates a high rate of false positives meaning that logistic regression was biassed towards prediciting “Up”
- d. Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train = (Year < 2009)
Weekly.2009 = Weekly[!train,]
Direction.2009 = Direction[!train]

glm.fit2 = glm(Direction ~ Lag2, data = Weekly, subset = train, family = binomial)

glm.probs2 = predict(glm.fit2, Weekly.2009, type = "response")

glm.pred2 = rep("Down", length(Direction.2009))

glm.pred2[glm.probs2 > .5] = "Up"

table(glm.pred2, Direction.2009)
```

```
##          Direction.2009
## glm.pred2 Down Up
##      Down    9  5
##      Up     34 56
```

```
print(paste("Fraction of correct predictions: ", mean(glm.pred2 == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.625"
```

- e. Repeat using LDA

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```r
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)

lda.pred = predict(lda.fit, Weekly.2009)

lda.class = lda.pred$class

table(lda.class, Direction.2009)
```

```
##          Direction.2009
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```r
print(paste("Fraction of correct predictions: ", mean(lda.class == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.625"
```

- f. Repeat using QDA

```r
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)

qda.class = predict(qda.fit, Weekly.2009)$class

table(qda.class, Direction.2009)
```

```
##          Direction.2009
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```r
print(paste("Fraction of correct predictions: ", mean(qda.class == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.586538461538462"
```

- g. Repeat using KNN with K = 1

```r
library(class)

train.X <- cbind(Lag2, Lag2)[train , ]
test.X <- cbind(Lag2, Lag2)[!train , ]
train.Direction <- Direction[train]

knn.pred = knn(train.X, test.X, train.Direction, k = 1)

table(knn.pred, Direction.2009)
```

```
##         Direction.2009
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
print(paste("Fraction of correct predictions: ", mean(knn.pred == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.5"
```

- h. Repeat using naive Bayes

```
library(e1071)

nb.fit = naiveBayes(Direction ~ Lag2, data = Weekly, subset = train)

nb.class = predict(nb.fit, Weekly.2009)

table(nb.class, Direction.2009)
```

```
##         Direction.2009
## nb.class Down Up
##     Down    0  0
##     Up     43 61
```

```
print(paste("Fraction of correct predictions: ", mean(nb.class == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.586538461538462"
```

- i. Which of these methods appears to provide the best results on this data?
- The best performing methods were logstic regression and lda with equal performance (62.5%)

- j. Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.
- First we will use best subset selection to determine the most useful combination of 3 predictors. Then we will try to improve naive bayes classification using kernal transformations to see if we get better results. Finally, we will try some different values of k to see the best performance.

```
library(leaps)

regfit_full = regsubsets(Direction ~., data = Weekly, subset = train)
summary(regfit_full)
```

```
## Subset selection object
## Call: regsubsets.formula(Direction ~ ., data = Weekly, subset = train)
## 8 Variables  (and intercept)
##        Forced in Forced out
## Year       FALSE      FALSE
## Lag1       FALSE      FALSE
## Lag2       FALSE      FALSE
## Lag3       FALSE      FALSE
## Lag4       FALSE      FALSE
## Lag5       FALSE      FALSE
## Volume     FALSE      FALSE
## Today      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          Year Lag1 Lag2 Lag3 Lag4 Lag5 Volume Today
## 1  ( 1 ) " "  " "  " "  " "  " "  " "  " "    "*"  
## 2  ( 1 ) " "  " "  " "  " "  " "  "*"  " "    "*"  
## 3  ( 1 ) " "  " "  " "  "*"  " "  "*"  " "    "*"  
## 4  ( 1 ) " "  " "  " "  "*"  "*"  "*"  " "    "*"  
## 5  ( 1 ) " "  " "  "*"  "*"  "*"  "*"  " "    "*"  
## 6  ( 1 ) " "  " "  "*"  "*"  "*"  "*"  "*"    "*"  
## 7  ( 1 ) "*"  " "  "*"  "*"  "*"  "*"  "*"    "*"  
## 8  ( 1 ) "*"  "*"  "*"  "*"  "*"  "*"  "*"    "*"
```

```
qda.fit = qda(Direction ~ Today + Lag3 + Lag5, data = Weekly, subset = train)

qda.class = predict(qda.fit, Weekly.2009)$class

table(qda.class, Direction.2009)
```

```
##          Direction.2009
## qda.class Down Up
##      Down   39  1
##      Up      4 60
```

```
print(paste("Fraction of correct predictions: ", mean(qda.class == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.951923076923077"
```

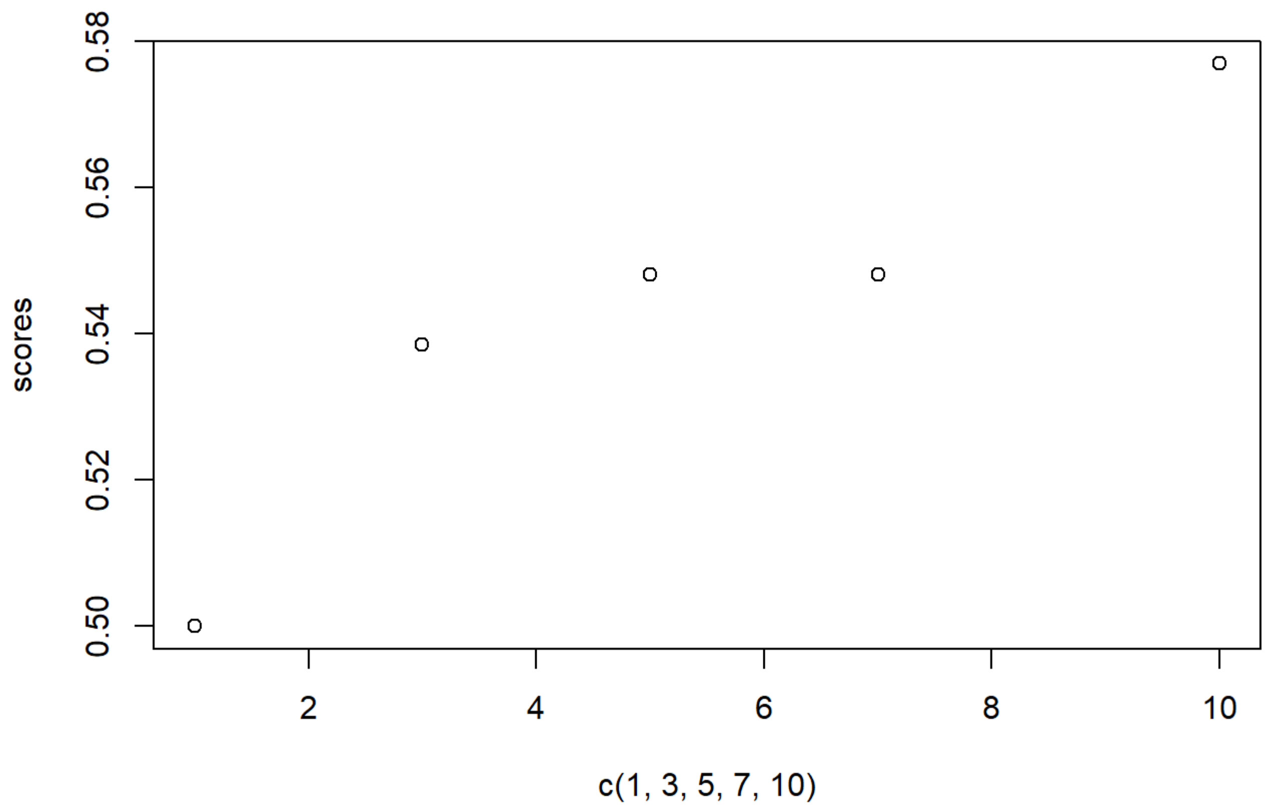
```
nb.fit = naiveBayes(Direction ~ ., data = Weekly, subset = train, type = 'C-classification', ker
nel = 'radial')

nb.class = predict(nb.fit, Weekly.2009)

table(nb.class, Direction.2009)
```

```
##          Direction.2009
## nb.class Down Up
##     Down   43 15
##     Up      0 46
```

```
print(paste("Fraction of correct predictions: ", mean(nb.class == Direction.2009)))
```

```
## [1] "Fraction of correct predictions:  0.855769230769231"
```

```
scores = c(mean(knn.pred == Direction.2009))

for (k_i in c(3,5,7,10)){
  knn.pred = knn(train.X, test.X, train.Direction, k = k_i)

  scores[[length(scores)+1]] = mean(knn.pred == Direction.2009)
}

plot(c(1,3,5,7,10), scores)
```



Section 5.4, page 197, question 2

- We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

  a. What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.
  - There are n observations. The probability of selecting any sample is equal. The proability that the jth observation is selected as the first observation is $\frac{1}{n}$. Therefore, the probability that the jth observation is not the first bootstrap observation is $1-\frac{1}{n}$

  b. What is the probability that the second bootstrap observation is not the jth observation from the original sample?
  - Since bootstrapping samples with replacement, the chances are equivalent of the jth observation not being selected for any observation, $1-\frac{1}{n}$

  c. Argue that the probability that the jth observation is not in the bootstrap sample is $(1-\frac{1}{n})^n$
  - Since the event of choosing a bootstrapped sample is independent, the probability that the jth observation will not be in the sample of size n is equal to the product of the probability of each observation not being the jth sample, which we have determined to be $1-\frac{1}{n}$, therefore, the for a sample of size n, there are n events with this probability, thus the probability that the jth observation is not in the bootstrap sample is $(1-\frac{1}{n})^n$

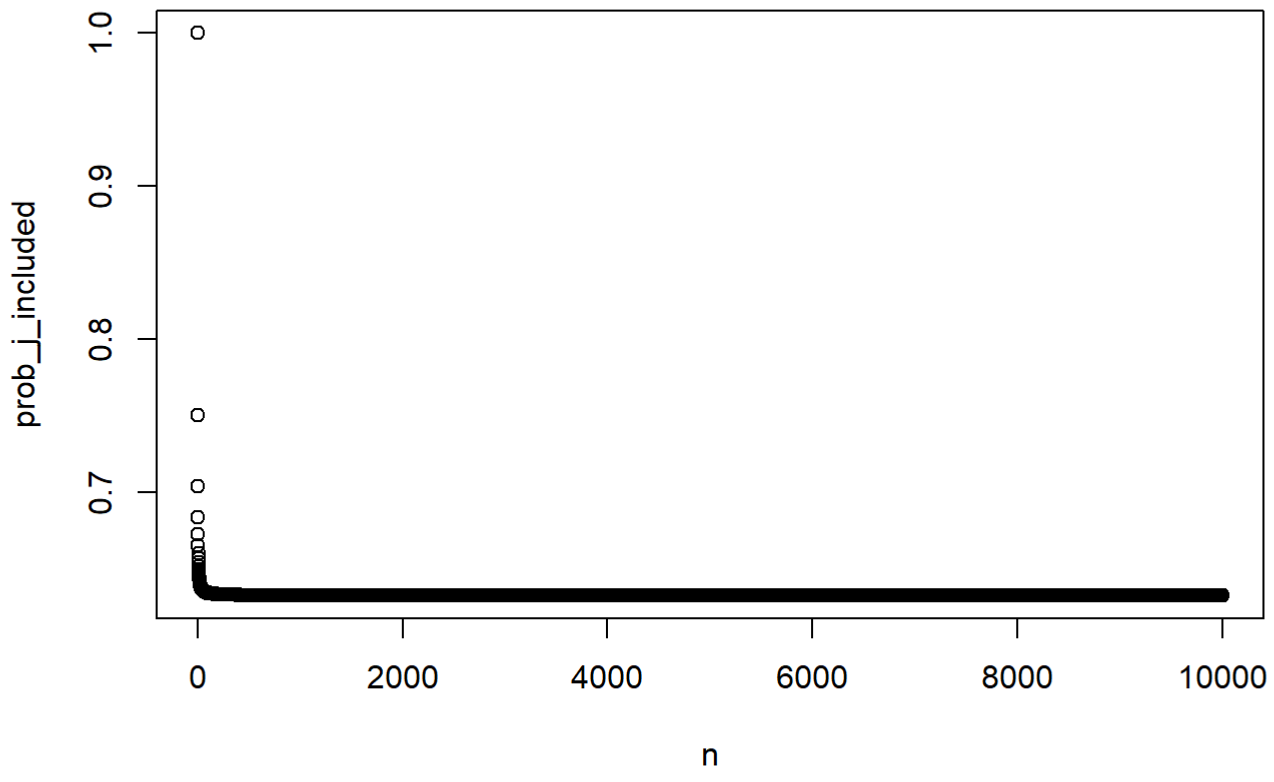
  d. When n = 5, what is the probability that the jth observation is in the bootstrap sample?
  - This is the probability that the jth observation is not **not** in the set, which is $1-(1-\frac{1}{5})^5)$ which equals 0.672

  e. When n = 100, what is the probability that the jth observation is in the bootstrap sample?
  - $1-(1-\frac{1}{100})^{100})=0.634$

  f. When n = 10,000, what is the probability that the jth observation is in the bootstrap sample?
  - $1-(1-\frac{1}{10000})^{10000})=0.632$

  g.

```
n <- 1:10000

df = data.frame(n, prob_j_included = 1 - (1-1/n)^n)

plot(df)
```



- The curve flattens out to around 0.632 or 1 - 1/e

h. We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

```r
store=rep(NA, 10000)

for(i in 1:10000) {
  store[i]=sum(sample (1:100, rep=TRUE)==4) > 0
}

mean(store)
```

```
## [1] 0.6339
```

- Here we generate 10000 bootstrap samples, then determine the proportion that contain 4. We should expect that 63.2% should contain 4. Our test returns a very close result to the expected value.

Section 5.4, page 200-201, question 8 (do not do part (f))

- We will now perform cross-validation on a simulated data set.

a. Generate a simulated data set as follows:

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

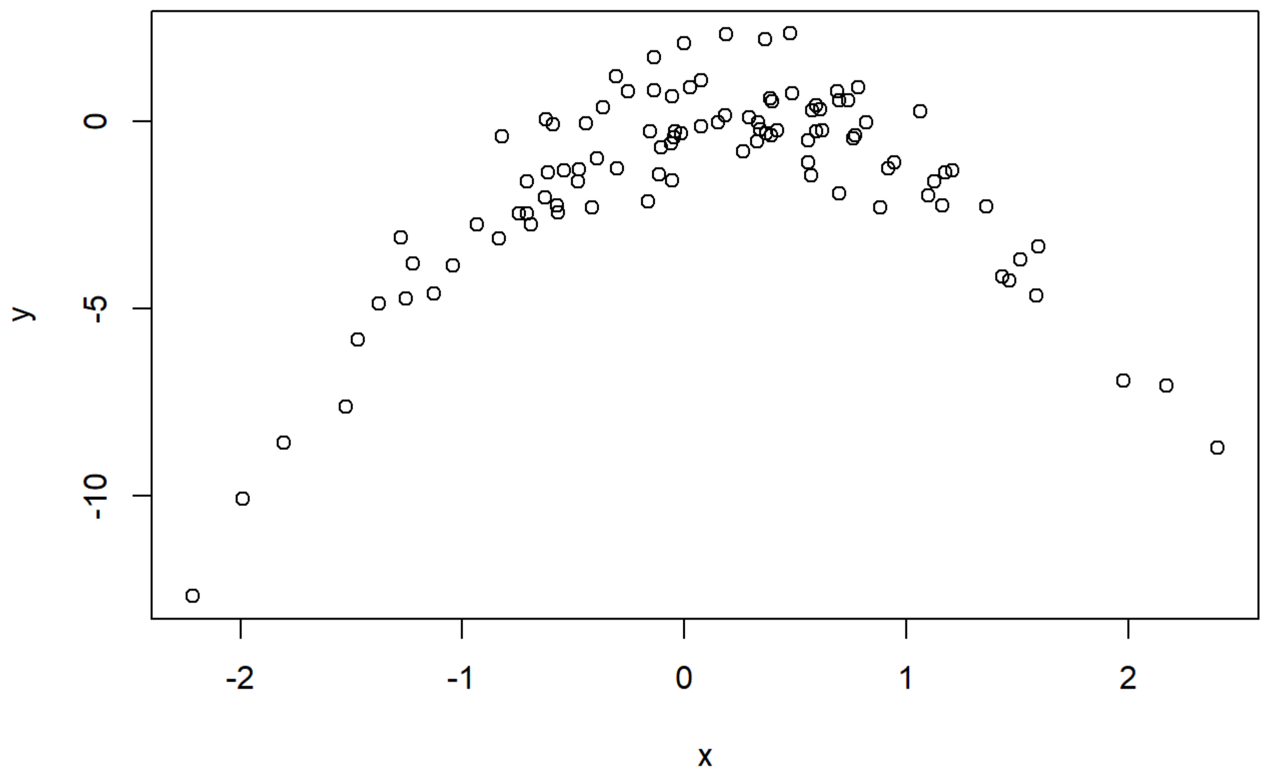
- In this data set, what is n and what is p? Write out the model used to generate the data in equation form.
- n = 100, p = 2.

$$Y = X - 2X^2 + eps$$

$$epsN(0, 1)$$

b. Create a scatterplot of X against Y . Comment on what you find.

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



- The data is parabolic which is to be expected due to the X^2 term. Most of the points are centered around the mean 0 of the normal distribution used to simulate noise.

c. Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

```
i. Y=β0+β1X+ϵ
```

```
library(boot)

set.seed(103)

data <- data.frame(x = x, y = y)

glm_1 <- glm(y ~ x, data = data)
cv_glm_1 <- cv.glm(data, glm_1)
cv_glm_1$delta
```

```
## [1] 7.288162 7.284744
```

```
ii. Y=β0+β1X+β2X^2+ϵ
```

```
glm_2 <- glm(y ~ x + I(x^2), data = data)
cv_glm_2 <- cv.glm(data, glm_2)
cv_glm_2$delta
```

```
## [1] 0.9374236 0.9371789
```

```
iii. Y=β0+β1X+β2X^2+β3X^3+ϵ
```

```
glm_3 <- glm(y ~ x + I(x^2) + I(x^3), data = data)
cv_glm_3 <- cv.glm(data, glm_3)
cv_glm_3$delta
```

```
## [1] 0.9566218 0.9562538
```

```
iv. Y=β0+β1X+β2X^2+β3X^3+β4X^4+ϵ
```

```
glm_4 <- glm(y ~ x + I(x^2) + I(x^3) + I(x^4), data = data)
cv_glm_4 <- cv.glm(data, glm_4)
cv_glm_4$delta
```

```
## [1] 0.9539049 0.9534453
```

d. Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```r
library(boot)

set.seed(5714)

data <- data.frame(x = x, y = y)

glm_1 <- glm(y ~ x, data = data)
cv_glm_1 <- cv.glm(data, glm_1)
cv_glm_1$delta
```

```
## [1] 7.288162 7.284744
```

```r
glm_2 <- glm(y ~ x + I(x^2), data = data)
cv_glm_2 <- cv.glm(data, glm_2)
cv_glm_2$delta
```

```
## [1] 0.9374236 0.9371789
```

```r
glm_3 <- glm(y ~ x + I(x^2) + I(x^3), data = data)
cv_glm_3 <- cv.glm(data, glm_3)
cv_glm_3$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
glm_4 <- glm(y ~ x + I(x^2) + I(x^3) + I(x^4), data = data)
cv_glm_4 <- cv.glm(data, glm_4)
cv_glm_4$delta
```

```
## [1] 0.9539049 0.9534453
```

- These results are the same because there is no randomness in loocv. If the order of the observations remains unchanged the train sets and corresponding test observations will be the same since we compute the loocv error on the same n models.

e. Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

- The quadratic model had the smallest loocv error. This is expected since we know the underlying relationship is quadratic. The benefit of using higher order polynomials is only seen when going from first to second, whereas going to third or fourth order polynomials does not increase the performance since they can approximate a quadratic relationship but have slightly higher variance.