HW5

David Schultheiss

10/5/2020

Problem 1

```
library(readr)
tumor <- read_csv("/Users/davidschultheiss/Downloads/tumor.csv")
```

```
## Parsed with column specification:
## cols(
##   Diagnosis = col_character(),
##   Radius = col_double(),
##   Texture = col_double(),
##   Perimeter = col_double(),
##   Area = col_double(),
##   Smoothness = col_double(),
##   Compactness = col_double(),
##   Concavity = col_double(),
##   `Concave Points` = col_double(),
##   Symmetry = col_double(),
##   `Fractal Dimension` = col_double()
## )
```

a)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v dplyr   0.8.3
## v tibble  2.1.3     v stringr 1.4.0
## v tidyr   1.0.0     v forcats 0.4.0
## v purrr   0.3.3

## Warning: package 'ggplot2' was built under R version 3.6.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
tumor = dplyr::select(tumor, Diagnosis, Radius, Texture, Smoothness, Concavity,
                `Fractal Dimension`)
tumor$Diagnosis = factor(tumor$Diagnosis)
is.factor(tumor$Diagnosis)
```

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

b)

```
set.seed(1)
library(MASS)
```

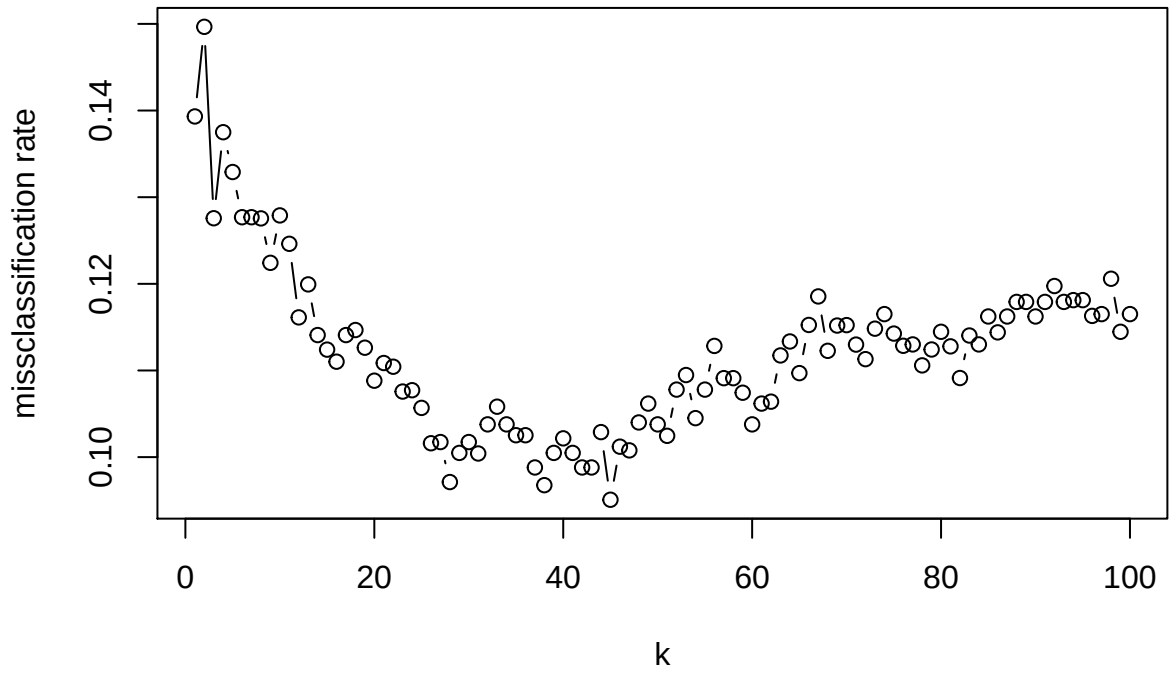
```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
folds = sample(1:10, nrow(tumor), replace= T)
errors = matrix(0, nrow= 10, ncol= 3)
colnames(errors) = c('Log', 'LDA', 'QDA')

for(f in 1:10) {
  cv.test = tumor[folds == f, ]
  cv.train = tumor[folds != f, ]
    #log
      glm.fit = glm(data= cv.train, Diagnosis ~ ., family= binomial)
      glm.probs = predict(glm.fit, cv.test, type= 'response')
      glm.pred = rep('Benign', nrow(cv.test))
      glm.pred[glm.probs > 0.5] = 'Malignant'
      glm.pred = factor(glm.pred)
      errors[f, 1] = mean(cv.test$Diagnosis != glm.pred)
    #LDA
      lda.fit = lda(data= cv.train, Diagnosis~.)
      lda.pred = predict(lda.fit, cv.test)$class
      errors[f, 2] = mean(cv.test$Diagnosis != lda.pred)
    #QDA
      qda.fit = qda(data= cv.train, Diagnosis~.)
      qda.pred = predict(qda.fit, cv.test)$class
      errors[f, 3] = mean(cv.test$Diagnosis != qda.pred)
}

overall = colMeans(errors)
overall
```

```
##        Log        LDA        QDA
## 0.05972969 0.07744677 0.05265711
```

c)

```
set.seed(1)
folds = sample(1:5, nrow(tumor), replace= T)
missclass = matrix(0, nrow= 5, ncol= 100)

library(class)
for(f in 1:5) {
  cv.test = tumor[folds == f, ]
  cv.train = tumor[folds != f, ]
  for (k in 1:100) {
    knn.pred = knn(cv.train[ , -1], cv.test[ ,-1], cv.train$Diagnosis, k)
    missclass[f, k] = mean(cv.test$Diagnosis != knn.pred)
  }
}

missrates = colMeans(missclass)
plot(missrates, xlab= 'k', ylab= 'missclassification rate', type= 'b')
```




```
which.min(missrates)
```

```
## [1] 45
```

```
missrates[45]
```

```
## [1] 0.09508428
```

d) QDA has the lowest misclassification rate of 5.2%

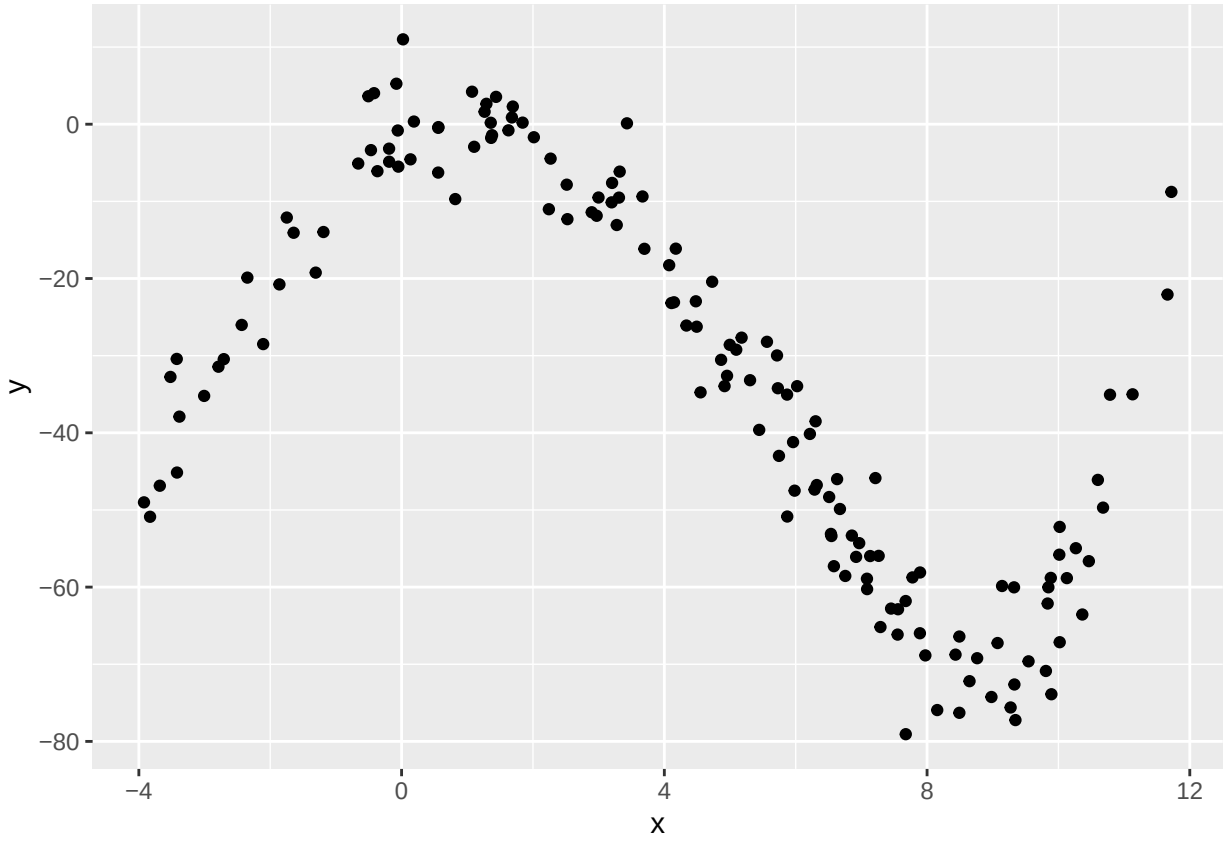
Problem 2

a)

```
data2 <- read_csv("/Users/davidschultheiss/Downloads/HW5data.csv")
```

```
## Parsed with column specification:
## cols(
##   x = col_double(),
##   y = col_double()
## )
```

```
ggplot(data= data2, mapping= aes(x= x, y= y)) +
  geom_point()
```

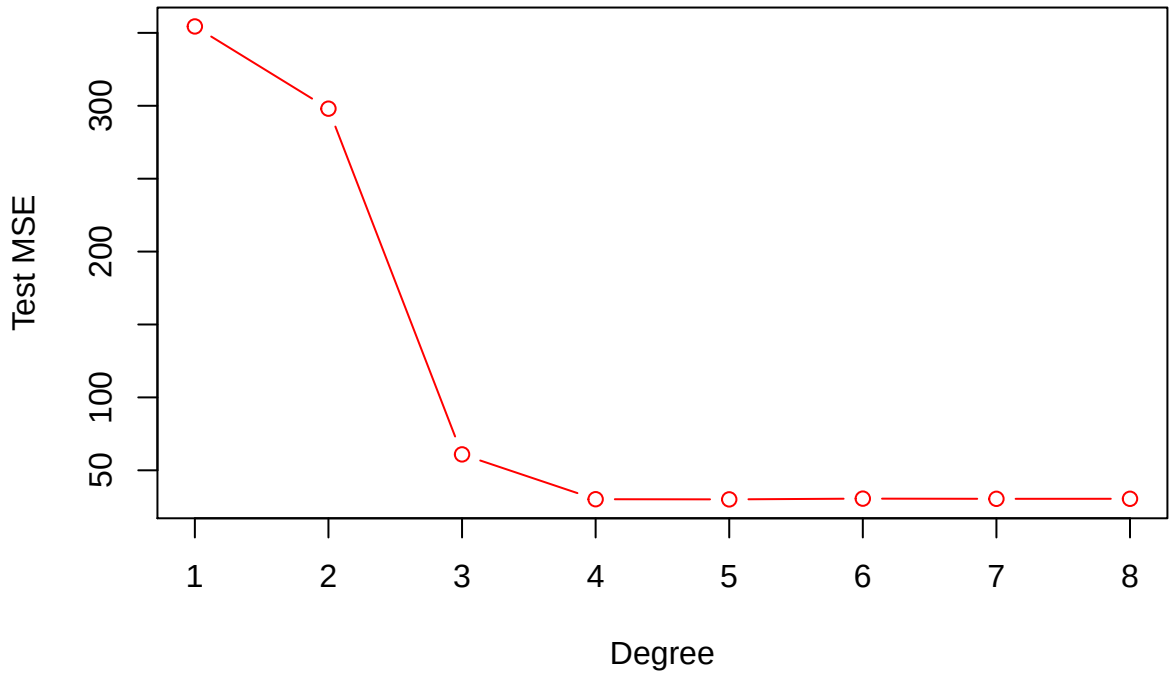


b)

```
loocv.error = rep(0 ,8)

library(boot)

for(i in 1:8) {
  glm.fit = glm(data= data2, y ~ poly(x, i, raw= T))
  loocv.error[i] = cv.glm(data2, glm.fit)$delta[1]
}

loocv.error
```

```
## [1] 354.39929 298.02124  60.91170  30.16478  30.06661  30.56880  30.44234
## [8]  30.46173
```

```
plot(loocv.error, type= 'b', xlab= 'Degree', ylab= 'Test MSE', col= 'red')
```




```
which.min(loocv.error)
```

```
## [1] 5
```

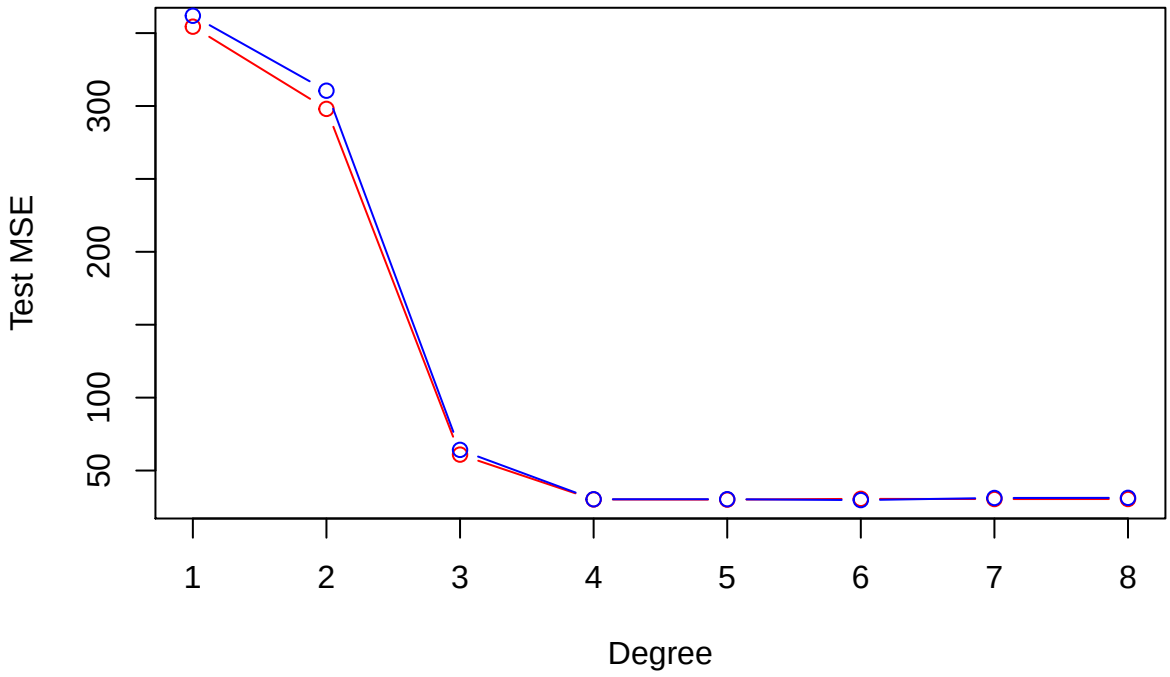
A polynomial of degree 5 has the lowest test MSE. However, the difference is negligable from a degree 4 polynomial. Considering this, I think a 4th degree polynomial would make the best fit.

c)

```
cv.error = rep(0,8)
for(i in 1:8) {
  glm.fit = glm(data= data2, y ~ poly(x, i, raw= T))
  cv.error[i] = cv.glm(data2, glm.fit, K= 10)$delta[1]
}

cv.error
```

```
## [1] 361.88617 310.48646  64.16768  30.29548  30.33852  29.72492  31.18284
## [8]  31.26445
```

```
plot(loocv.error, type= 'b', xlab= 'Degree', ylab= 'Test MSE', col= 'red')
lines(cv.error, type='b', col='blue')
```



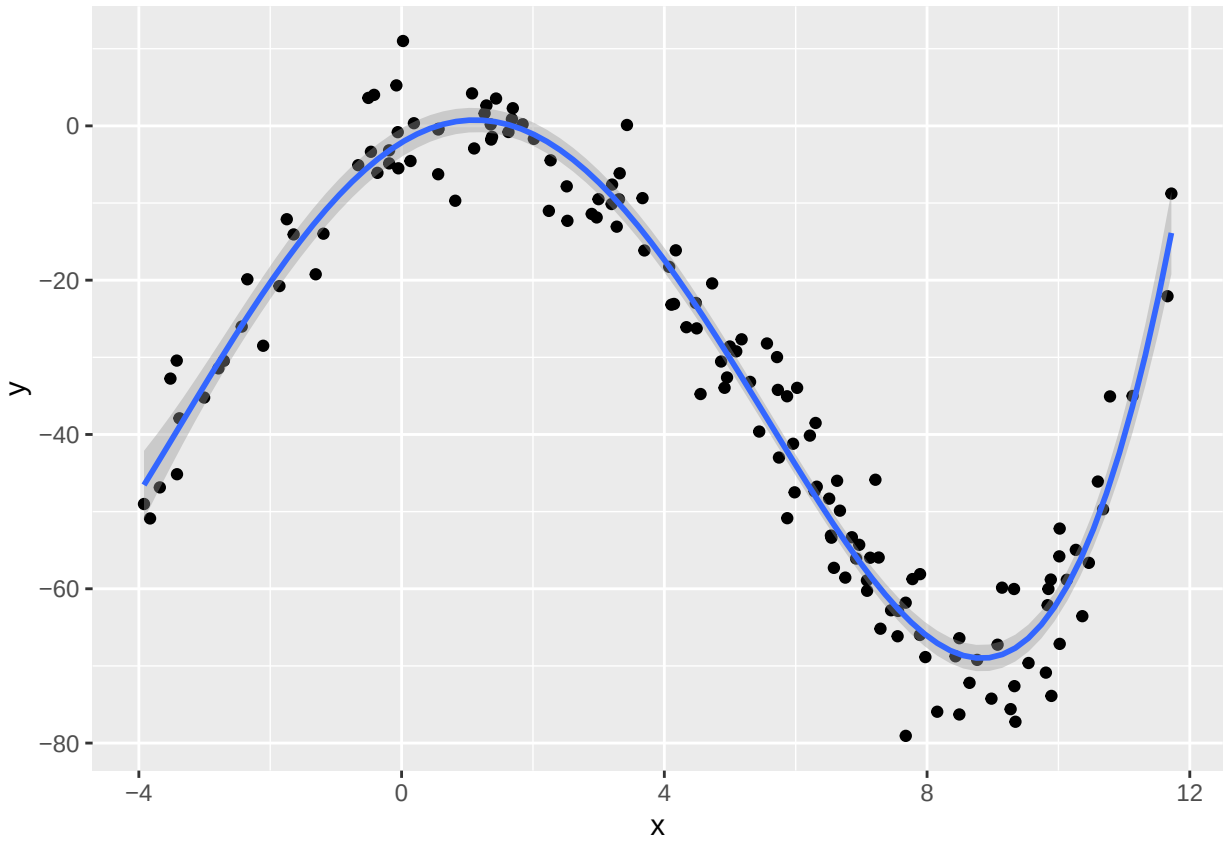

```
which.min(cv.error)
```

```
## [1] 6
```

A 4th degree polynomial fits best. We do get slightly different Test MSE from LOOCV. I also didn't notice a speed difference because the data set wasn't particularly large. With more data, bootstrapping would be more noticably faster.

d)

```
ggplot(data= data2, mapping= aes(x= x, y= y)) +
  geom_point() +
  stat_smooth(method= 'glm', formula = y ~ poly(x, 4, raw=T))
```



It looks like a great fit!