STATS 790
Homework 1

Sophie Stelmach | 400172067

20 January, 2023

Contents

Question 1

I completely agree that the so-called "data modelling culture" has been taught and spoken about in my classes much more than the "algorithmic modelling culture". Something that stuck with me in particular was the mention that conclusions regarding model fits are really about the model's mechanism rather than the actual mechanism behind the problem/pattern. I wonder that the Breiman does not give statistical models enough credit, however, as he does repute the notion of models being treated as fact. Though, again, this could be my bias after being taught the usefulness of statistical models for so long.

Question 2

According to Hastie, Tibshirani, and Friedman (2013) the data was generated from two bivariate Gaussian distributions, with 10 means labelled as class *blue* and another 10 as class *orange* Means

with class blue were generated from $N((1,0)^T, \mathbf{I})$, and means with class orange were generated from $N((0,1)^T, \mathbf{I})$. 100 observations were generated for each of the two classes by picking a mean (each with probability 0.1) and generating a normal distribution with that mean and variance $\mathbf{I}/5$.

```
library(MASS)
library(caret)
```

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

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

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'
```

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

```
library(class)
library(FNN)
```

```
##
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```
set.seed(5)
```

First, the data for class *blue* was generated:

```
df_blue <- data.frame(row.names = c("x","y"))

for (i in 1:100){
  #generate 10 means
  blue_mk <- as.data.frame(mvrnorm(n=10
                                   ,mu=c(0, 1)
                                   ,Sigma= diag(2)
  )
  )

  blue_mu = blue_mk %>% sample_frac(0.1)

  blue_sample <- as.data.frame(mvrnorm(n=1
                                       ,mu= c(blue_mu$V1, blue_mu$V2)
                                       ,Sigma= (diag(2)/5)
```

```
  )
  )
  df_blue[[i]] <- as.data.frame(blue_sample)
}

df_blue <- t(df_blue)
blue <- as.data.frame(df_blue)
blue$class <- (rep(0,100))

df_orange <- data.frame(row.names = c("x","y"))
```

The same process was used for data of class *orange*:

```
for (i in 1:100){
 #generate 10 means
  orange_mk <- as.data.frame(mvrnorm(n=10
                             ,mu=c(0, 1)
                             ,Sigma= diag(2)
      )
      )

  or_mu = orange_mk %>% sample_frac(0.1)

  orange_sample <- as.data.frame(mvrnorm(n=1
                                 ,mu= c(or_mu$V1, or_mu$V2)
                                 ,Sigma= (diag(2)/5)
      )
      )
  df_orange[[i]] <- as.data.frame(orange_sample)
}

df_orange <- t(df_orange)
```

```
orange <- as.data.frame(df_orange)
orange$class <- (rep(1,100))


data <- rbind(blue, orange)
```

Then, KNN was performed on the data:

```
# create test and training set
index <- sample(1:nrow(data), 0.8*nrow(data))
train <- data[index,]
test <- data[-index,]

grid <- expand.grid(x = seq(min(test$x), max(test$x), by = 0.1),
                    y = seq(min(test$y), max(test$y), by = 0.1))

knn_pred <- knn(data[,c(1,2)], grid, data$class, k = 15, prob = TRUE)
prob <- attr(knn_pred, "prob")

a <- mutate(grid, prob = prob, class = 0,
              prob_cls = ifelse(knn_pred == class, 1, 0))

b <- mutate(grid, prob = prob, class = 1,
              prob_cls = ifelse(knn_pred == class, 1, 0))
ab <- bind_rows(a, b)
```

The plot similar to Figure 2.2 in Hastie, Tibshirani, and Friedman (2013) was generated using the code below. I followed Jeffrey W. Doser (2022) as a guideline on how to build the plot.

```
ggplot(ab) +
  geom_point(aes(x = x, y = y, color = class),
             data = mutate(grid, class = knn_pred),
             size = 0.5) +
  geom_point(aes(x = x, y = y, color = as.factor(class)),
```

```
            size = 4, shape = 1, data = data) +
  scale_color_manual(values=c("#ffaf0f", "#48a2f7")) +
  geom_contour(aes(x = x, y = x, z = prob_cls,
            group = as.factor(class),
            color = "black"),
            size = 1, bins = 1, data = ab)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
## Warning: ‘stat_contour()‘: Zero contours were generated
```

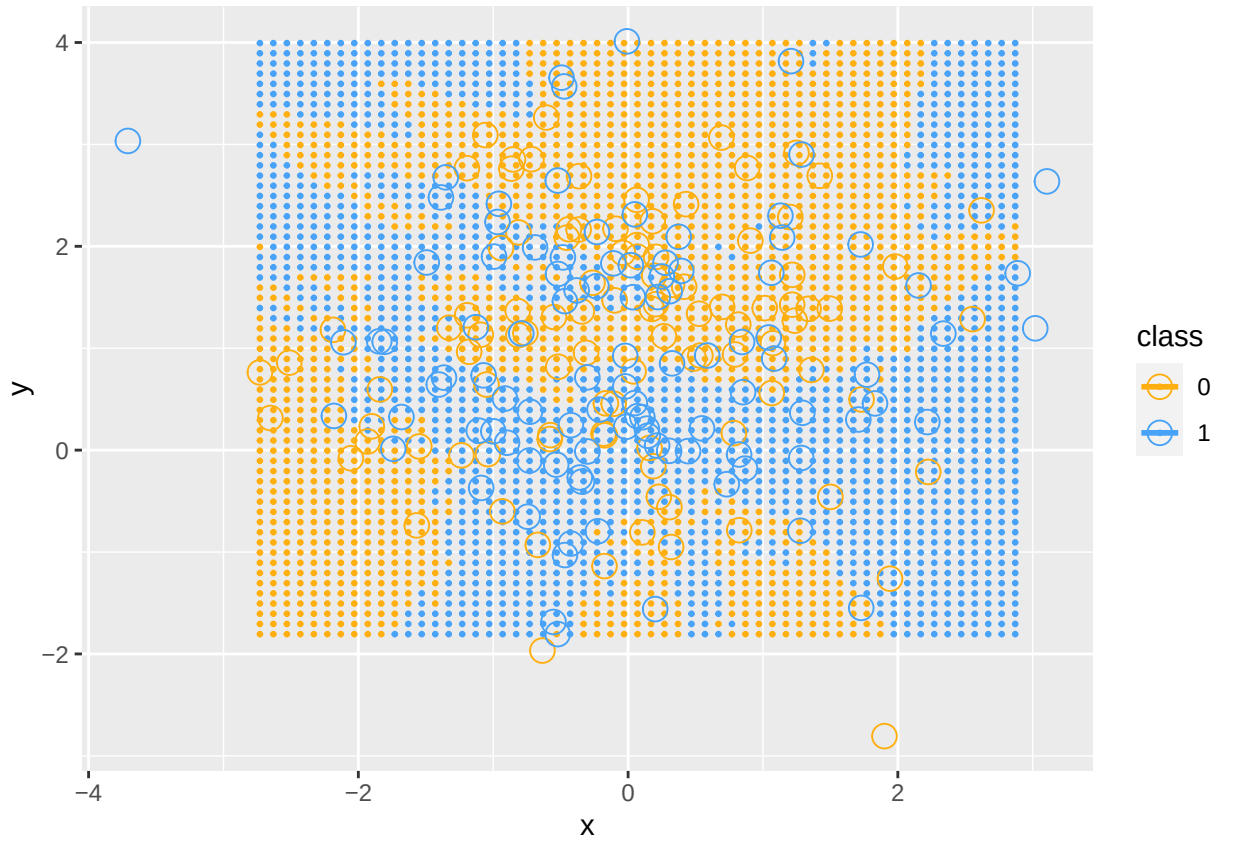
```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```

```
## Warning: ‘stat_contour()‘: Zero contours were generated
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```



Question 3

Let $f(y)$ and $F(y)$ be the probability density function and cumulative density function of a random variable Y. Then,

$$\begin{aligned}
\text{MAE} &= E[|Y-m|] && (1)\\
&= \int_{-\infty}^{\infty} f(y)|Y-m|dy && (2)\\
&= \int_{-\infty}^{m} f(y)|Y-m|dy + \int_{m}^{\infty} f(y)|Y-m|dy && (3)\\
&= \int_{-\infty}^{m} f(y)|Y-m|dy + \int_{m}^{\infty} f(y)|Y-m|dy && (4)\\
& && (5)
\end{aligned}$$

We note that when $m < Y$, $|Y-m| = Y-m$, and when $m > Y$, $|Y-m| = m-Y$. Then, we take the derivative with respect to m.

$$d\text{MAE}/dm = d/dm \int_{-\infty}^{m} f(y)|Y - m|dy + d/dm \int_{m}^{\infty} f(y)|Y - m|dy \tag{6}$$

$$= \int_{-\infty}^{m} f(y)(Y - m)dy + \int_{m}^{\infty} f(y)(m - Y)dy \tag{7}$$

$$= \int_{-\infty}^{m} f(y)dy + \int_{m}^{\infty} f(y)dy \text{by Leibnitz rule} \tag{8}$$

$$0 = \int_{-\infty}^{m} f(y)dy + \int_{m}^{\infty} f(y)dy \tag{9}$$

$$0 = P(X \leq m) + P(X \geq m) = 1 \tag{10}$$

$$\tag{11}$$

Therefore $P(X \leq m) = P(X \geq m) = 1/2$, which means m is the median. Thus, MAE is minimized when m is the median.

We should use MSE rather than MAE to measure error, since in MSE the weight that distance from observations takes is lower for smaller distances (smaller error) and higher for greater distances (greater error). This is due to the difference between predicted and observed values being squared in the MSE.

Question 4

Linear smoothers estimate the regression function with: $\hat{\mu}(x) = \sum_i y_i \hat{w}(x_i, x)$

For global mean:

$$\mu(x) = 1/n \sum_i y_i$$

So $\hat{w}(x_i, x)$ is a diagonal matrix with elements $1/n$ by definition.

The effective degrees of freedom:

$$\begin{aligned} \text{tr}\mathbf{w} &= \sum_{i=1}^{n} 1/n \\ &= n * 1/n \\ &= n/n \\ &= 1 \end{aligned}$$

Question 5

By equation 1.55 in Shalizi (2013):

$$\hat{w}(x_i, x) = \begin{cases} 1/k & x_i \text{ is one of k nearest neighbours of } x \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We define $\hat{\mathbf{w}}$ by $\hat{w}(x_i, x)$, i.e. the nxn,

$$\begin{bmatrix} 1/k & 0 & ... & 0 \\ 0 & 1/k & ... & 0 \\ 0 & ... & ... & 1/k \end{bmatrix}$$

To find the effective degrees of freedom:

$$\text{tr}\mathbf{w} = \sum_{i=1}^{n} 1/k \tag{13}$$

$$= n * 1/k \tag{14}$$

$$= n/k \tag{15}$$

$$= 1, \text{ if } k = n. \tag{16}$$

Question 6

The data was downloaded as instructed in Hastie, Tibshirani, and Friedman (2013) and observations from classes 2 and 3 were selected from the training and test data. There was an extra column in the

training data that contained all NA values (probably a mishap that occurred while downloading or loading the data), so this was removed. KNN classification was applied for values of $k = 1, 3, 5, 7, 15$.

I would like to note that I created a for loop which would perform KNN classification on all the k value listed, however it would only output error values for $k = 1, 5, 15$. I decided to produce the error values manually, but have provided the code for the for-loop at the end.

```
train_zip <- read.table("ziptrain.txt", sep=" ")
test_zip <- read.table("ziptest.txt", sep=" ")

train_zip <- as.data.frame(subset(train_zip, V1 == 2 | V1 == 3))
train_zip <- train_zip[,-258]

test_zip <- as.data.frame(subset(test_zip, V1 == 2 | V1 == 3))
```

```
set.seed(5)
knn_pred_test1 <- knn(train = train_zip,
                      test = test_zip,
                      cl= train_zip$V1,
                      k = 1)
knn_pred_train1 <- knn(train = train_zip,
                       test = train_zip,
                       cl= train_zip$V1,
                       k = 1)

test_error1 <- mean(test_zip$V1 != knn_pred_test1)
train_error1 <- mean(train_zip$V1 != knn_pred_train1)
message("Errors for k=1")
```

```
## Errors for k=1
```

```
list("test error" = test_error1, "train error" = train_error1)
```

```
## $`test error`
```

```
## [1] 0.02472527
##
## $`train error`
## [1] 0
```

```
knn_pred_test3 <- knn(train = train_zip,
                      test = test_zip,
                      cl= train_zip$V1,
                      k = 3)
knn_pred_train3 <- knn(train = train_zip,
                       test = train_zip,
                       cl= train_zip$V1,
                       k = 3)

test_error3 <- mean(test_zip$V1 != knn_pred_test3)
train_error3 <- mean(train_zip$V1 != knn_pred_train3)
message("Errors for k=3")
```

```
## Errors for k=3
```

```
list("test error" = test_error3, "train error" = train_error3)
```

```
## $`test error`
## [1] 0.03021978
##
## $`train error`
## [1] 0.004319654
```

```
knn_pred_test5 <- knn(train = train_zip,
                      test = test_zip,
                      cl= train_zip$V1,
                      k =5)
knn_pred_train5 <- knn(train = train_zip,
```

```
                      test = train_zip,
                      cl= train_zip$V1,
                      k = 5)

test_error5 <- mean(test_zip$V1 != knn_pred_test5)
train_error5 <- mean(train_zip$V1 != knn_pred_train5)
message("Errors for k=5")
```

```
## Errors for k=5
```

```
list("test error" = test_error5, "train error" = train_error5)
```

```
## $`test error`
## [1] 0.03021978
##
## $`train error`
## [1] 0.005759539
```

```
knn_pred_test7 <- knn(train = train_zip,
                      test = test_zip,
                      cl= train_zip$V1,
                      k = 7)
knn_pred_train7 <- knn(train = train_zip,
                       test = train_zip,
                       cl= train_zip$V1,
                       k = 7)

test_error7 <- mean(test_zip$V1 != knn_pred_test7)
train_error7 <- mean(train_zip$V1 != knn_pred_train7)
message("Errors for k=7")
```

```
## Errors for k=7
```

```
list("test error" = test_error7, "train error" = train_error7)
```

```
## $`test error`
## [1] 0.03021978
##
## $`train error`
## [1] 0.005759539
```

```
knn_pred_test15 <- knn(train = train_zip,
                       test = test_zip,
                       cl= train_zip$V1,
                       k = 15)
knn_pred_train15 <- knn(train = train_zip,
                        test = train_zip,
                        cl= train_zip$V1,
                        k = 15)

test_error15 <- mean(test_zip$V1 != knn_pred_test15)
train_error15 <- mean(train_zip$V1 != knn_pred_train15)
message("Errors for k=15")
```

```
## Errors for k=15
```

```
list("test error" = test_error15, "train error" = train_error15)
```

```
## $`test error`
## [1] 0.03846154
##
## $`train error`
## [1] 0.009359251
```

For loop:

```r
ks <- c(1,3,5,7,15)
ks <- as.numeric(ks)

for (i in ks){
  knn_pred_test <- knn(train = train_zip,
                       test = test_zip,
                       cl= train_zip$V1,
                       k = ks[i])
  knn_pred_train <- knn(train = train_zip,
                        test = train_zip,
                        cl= train_zip$V1,
                        k = ks[i])

  test_error <- mean(test_zip$V1 != knn_pred_test)
  train_error <- mean(train_zip$V1 != knn_pred_train)
  print(round(cbind(test.error = test_error,
                    train.error= train_error,
                    k=ks[i]),8))
}
```

Bibliography

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2013. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY: Springer.

Jeffrey W. Doser, Andrew O. Finley. 2022. "R Programming for Data Sciences." *Jeff Doser.* https://www.jeffdoser.com/files/for875/_book/index.html.

Shalizi, Cosma. 2013. *Advanced Data Analysis from an Elementary Point of View.* Citeseer.