Lab03

Amanda Mentesana

2025-02-16

```
knitr::opts_chunk$set(echo = TRUE)


library(class)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

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

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

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

```
library(ggplot2)
```

Dataset prep and creating train and test sets from the abalone dataset.

```
abalone <- read.csv("C:/Users/amanda/OneDrive/Documents/RPI/Spring 2025/BCBP 4600/R_Project_Data_Analyti

abalone$age.group <- cut(abalone$rings, br=c(0,8,11,35), labels = c("young", 'adult', 'old'))


## alternative way of setting age.group
#abalone$age.group[abalone$rings<=8] <- "young"
#abalone$age.group[abalone$rings>8 & abalone$rings<=11] <- "adult"
#abalone$age.group[abalone$rings>11 & abalone$rings<=35] <- "old"

# sample create list of (80% of 4176, 3340) numbers randomly sampled from 1-3340
n = 4176
s_abalone <- sample(n,n*.8)

## create train & test sets based on sampled indexes
abalone.train <- abalone[s_abalone,]

abalone.test <- abalone[-s_abalone,]
```

1) Train and Evaluate 2 kNN Models as a Predictor of Age

1a) This kNN model is trained on a subset of length, diameter, and height.

```
sqrt(4176) #use sqrt(#observations) for an estimate of k-value size
```

```
## [1] 64.62198
```

```
knn.predicted <- knn(train = abalone.train[,2:4], test = abalone.test[,2:4], cl=abalone.train$age.group

# create contingency table/ confusion matrix
contingency.table <- table(knn.predicted, abalone.test$age.group, dnn=list('predicted','actual'))

contingency.table
```

```
##          actual
## predicted young adult old
##     young   214    69  23
##     adult    78   259 126
##     old       5    28  34
```

```
# calculate classification accuracy
sum(diag(contingency.table))/length(abalone.test$age.group)
```

```
## [1] 0.6064593
```

1b) This kNN model is trained on a subset of whole weight, shucked weight, viscera weight, and shell weight.

```
sqrt(4176) #use sqrt(#observations) for an estimate of k-value size
```

```
## [1] 64.62198
```

```
knn.predicted <- knn(train = abalone.train[,5:8], test = abalone.test[,5:8], cl=abalone.train$age.group

# create contingency table/ confusion matrix
contingency.table <- table(knn.predicted, abalone.test$age.group, dnn=list('predicted','actual'))

contingency.table
```

```
##          actual
## predicted young adult old
##     young   216    58  10
##     adult    78   274 116
##     old       3    24  57
```

```r
# calculate classification accuracy
sum(diag(contingency.table))/length(abalone.test$age.group)
```

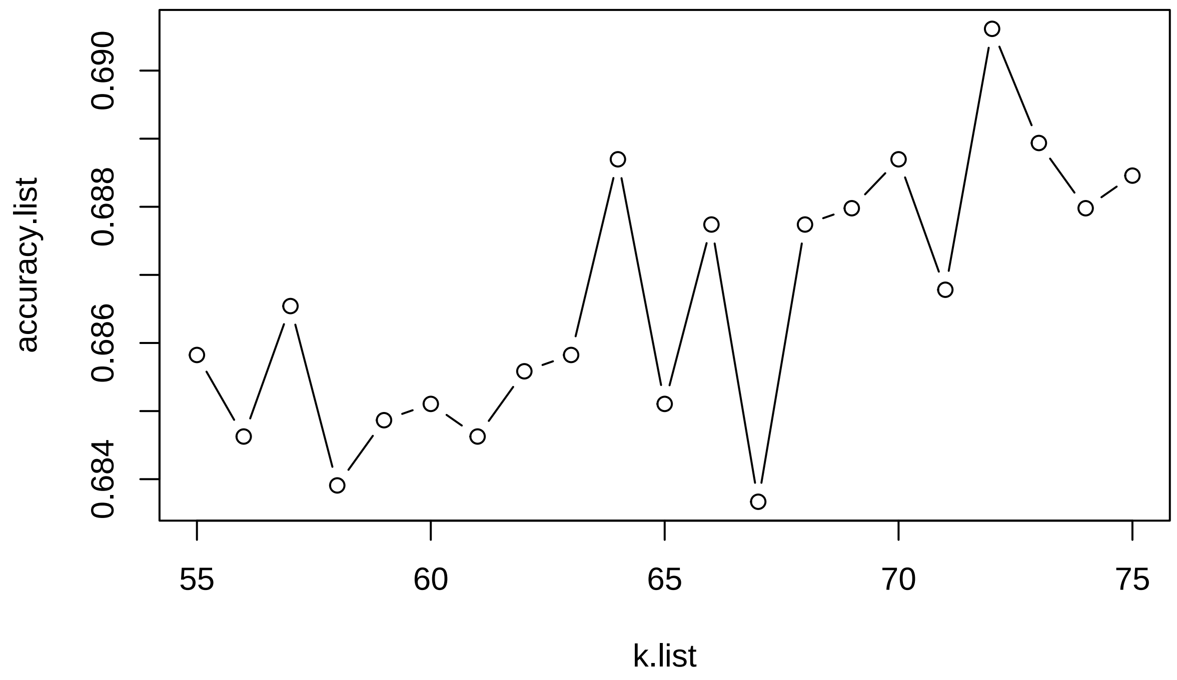
```
## [1] 0.6543062
```

1c) Based on the two models, the second model which uses a subset of the abalone weights is the better preforming model. Based on the contingency tables, the second model with a score of 68.54067% slightly outpreforms the first model with a score of 65.311% accuracy.

1d) Finding the optimal k value for the second model.

Using a for loop and checking for accuracy values this method leads us to an optimal k of 75. However, using the caret function to find its own optimal k, it found it to be 17, which was very distant from our original rule-of-thumb estimate of 65. k = 17 leads to an accuracy of 69.37799% which outperforms higher estimates of k.

```r
## train knn models for multiple values of k and plot accuracies

# list of k values
k.list <- c(55:75)

# empty list for accuracy
accuracy.list <- c()

# loop: train&predict model for each k, compute accuracy and append it to list
for (k in k.list) {

  knn.predicted <- knn(train = abalone[,5:8], test = abalone[,5:8], cl = abalone$age.group, k = k)

  contingency.table <- table(knn.predicted, abalone$age.group, dnn=list('predicted','actual'))

  accuracy <- sum(diag(contingency.table))/length(abalone$age.group)

  accuracy.list <- c(accuracy.list,accuracy)

}


# plot acccuracy with k-values in k.list, limiting values of y axis between .9 & 1
plot(k.list,accuracy.list,type = "b")
```



```r
## based on this method the ideal k value would be 75, not 65


## train and evaluate multiple knn models to find optimal k, using caret library function tunelength=#k
knn.model <- train(abalone[,5:8], abalone$age.group, method = "knn", tuneLength = 10, trControl = train(

# print model outputs
print(knn.model)
```

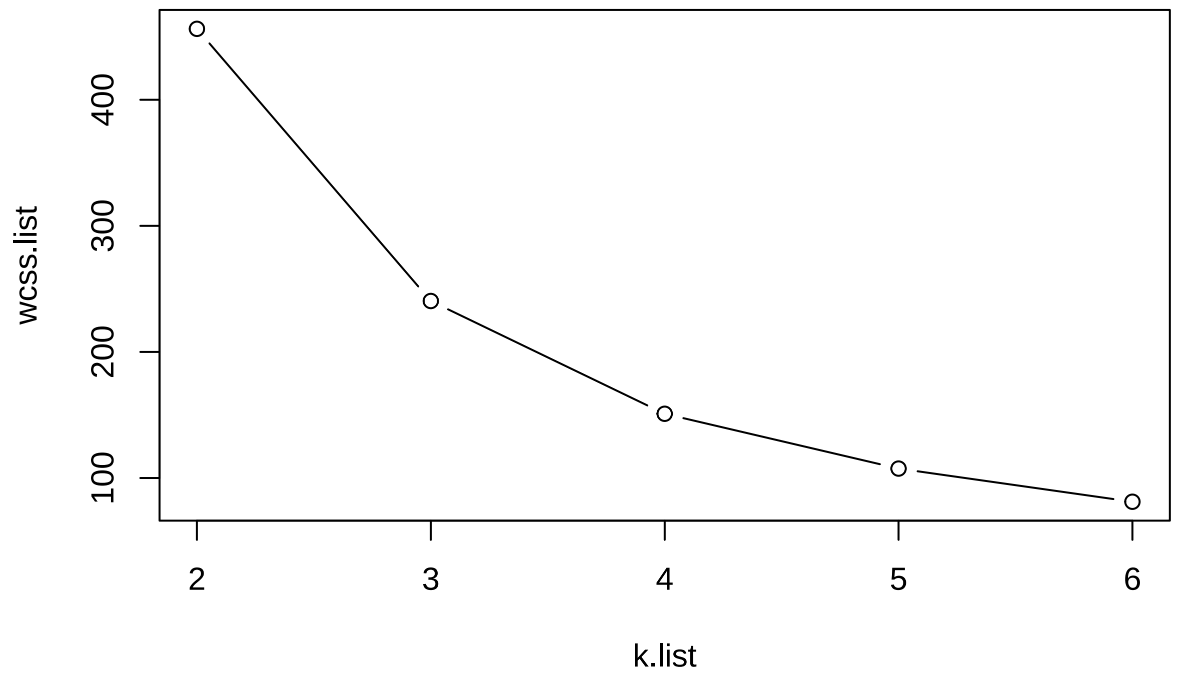
```
## k-Nearest Neighbors
##
## 4176 samples
##    4 predictor
##    3 classes: 'young', 'adult', 'old'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3759, 3758, 3758, 3758, 3759, 3758, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.6496691  0.4503890
##    7  0.6606837  0.4649725
##    9  0.6664270  0.4727297
##   11  0.6736115  0.4829558
##   13  0.6724194  0.4804426
##   15  0.6788735  0.4905993
##   17  0.6769712  0.4868856
##   19  0.6843817  0.4982368
##   21  0.6817524  0.4937683
##   23  0.6817495  0.4932927
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 19.
```

2) Training the k-means models for the weight subset of the abalone dataset.

2a) Finding optimal k.

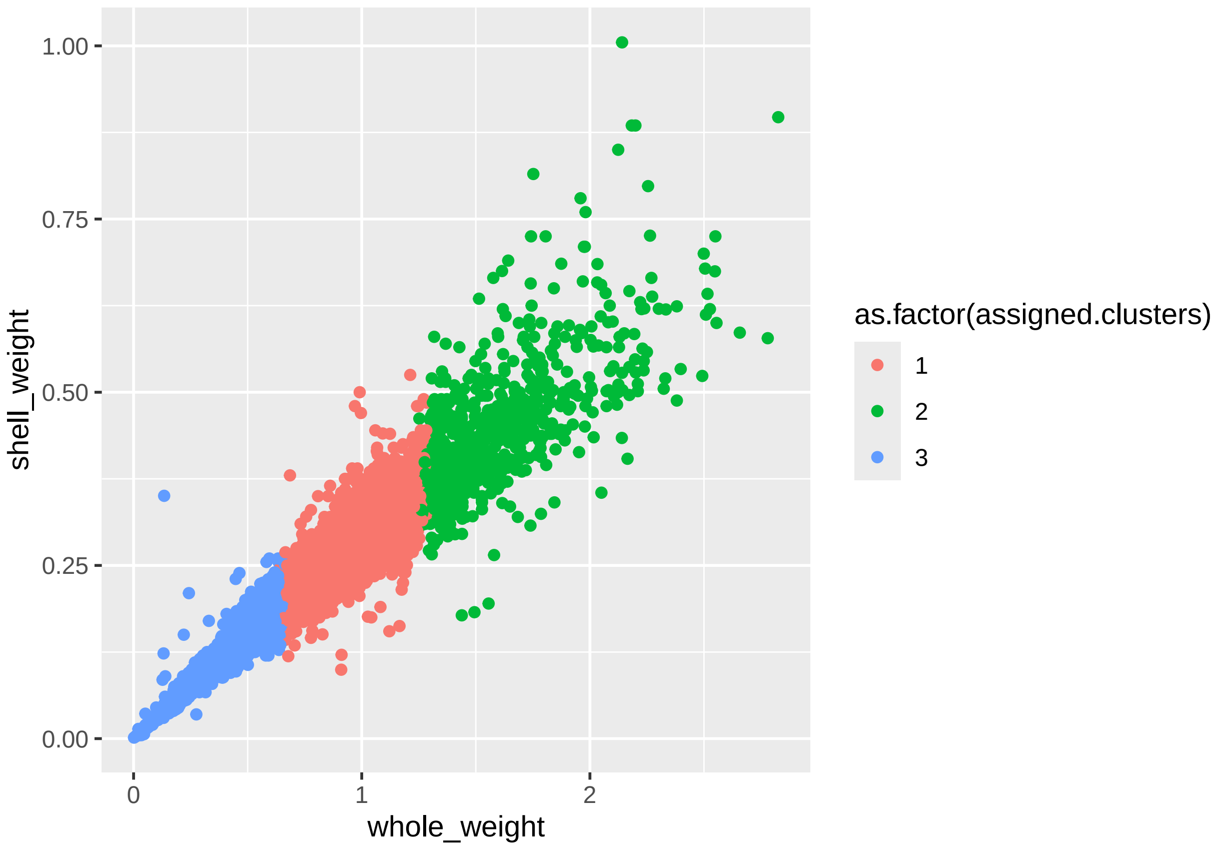
Looking at the clustering output, we can see that there is an 'elbow' at k = 3, which indicates that there are three groups

```r
## run tests with multiple k values and plot WCSS
k.list <- c(2,3,4,5,6)

wcss.list <- c()

for (k in k.list) {

  abalone.km <- kmeans(abalone[,5:8], centers = k)

  wcss <- abalone.km$tot.withinss

  wcss.list <- c(wcss.list,wcss)

  ## get and plot clustering output
  assigned.clusters <- as.factor(abalone.km$cluster)

}


plot(k.list,wcss.list,type = "b")
```



2b) Using k = 3 and plotting whole weight vs shell weight.

```
## run kmeans
k = 3
abalone.km <- kmeans(abalone[,5:8], centers = k)

## get and plot clustering output

assigned.clusters <- as.factor(abalone.km$cluster)

ggplot(abalone, aes(x = whole_weight, y = shell_weight, colour = as.factor(assigned.clusters))) +
  geom_point()
```


1.00
0.75
0.50
0.25
0.00
shell_weight
0
1
2
whole_weight
as.factor(assigned.clusters)
1
2
3