DSE 6211 Module 08 Lab 08

Joseph Annand

2024-03-03

## Load libraries

library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(caret)

## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: ggplot2

## Loading required package: lattice

library(NbClust)

## Warning: package 'NbClust' was built under R version 4.3.1

## Data pre-processing

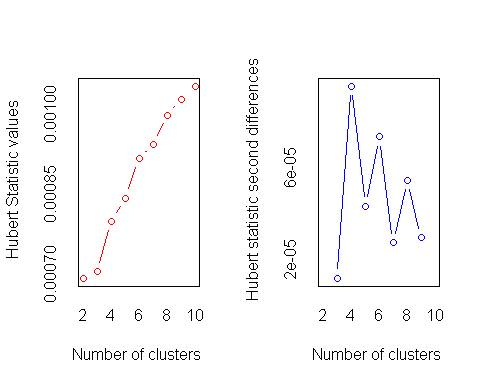
data <- read.csv("lab\_8\_data/lab\_8\_data.csv")  
training\_ind <- createDataPartition(data$lodgepole\_pine,  
 p = 0.75,  
 list = F,  
 times = 1)  
  
training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
top\_20\_soil\_types <- training\_set %>%  
 group\_by(soil\_type) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 select(soil\_type) %>%  
 top\_n(20)

## Selecting by soil\_type

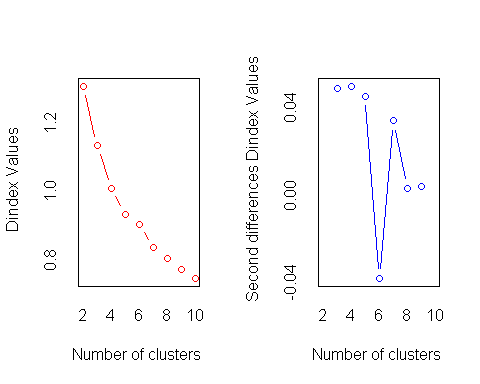
training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 training\_set$soil\_type,  
 "other")  
  
training\_set$wilderness\_area <- factor(training\_set$wilderness\_area)  
training\_set$soil\_type <- factor(training\_set$soil\_type)  
  
  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,  
 training\_set[, c("wilderness\_area", "soil\_type")],  
 levelsOnly = T,  
 fullRank = T)  
  
onehot\_enc\_training <- predict(onehot\_encoder,  
 training\_set[, c("wilderness\_area", "soil\_type")])  
  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
  
  
test\_set$soil\_type <- ifelse(test\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 test\_set$soil\_type,  
 "other")  
  
test\_set$wilderness\_area <- factor(test\_set$wilderness\_area)  
test\_set$soil\_type <- factor(test\_set$soil\_type)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("wilderness\_area", "soil\_type")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)  
  
  
test\_set[, -c(11:13)] <- scale(test\_set[, -c(11:13)],  
 center = apply(training\_set[, -c(11:13)], 2, mean),  
 scale = apply(training\_set[, -c(11:13)], 2, sd))  
training\_set[, -c(11:13)] <- scale(training\_set[, -c(11:13)])  
  
  
training\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(training\_set), 33))  
training\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(training\_set)))  
  
test\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(test\_set)))

## K-means clustering

set.seed(123)  
nc <- NbClust(training\_features[sample(nrow(training\_features), 1000), c(4, 6, 10)],  
 min.nc = 2, max.nc = 10, method = "kmeans")



## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.   
##



## \*\*\* : The D index is a graphical method of determining the number of clusters.   
## In the plot of D index, we seek a significant knee (the significant peak in Dindex  
## second differences plot) that corresponds to a significant increase of the value of  
## the measure.   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## \* Among all indices:   
## \* 5 proposed 2 as the best number of clusters   
## \* 4 proposed 3 as the best number of clusters   
## \* 8 proposed 4 as the best number of clusters   
## \* 1 proposed 5 as the best number of clusters   
## \* 3 proposed 7 as the best number of clusters   
## \* 1 proposed 8 as the best number of clusters   
## \* 1 proposed 10 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 4   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

km\_clusters <- kmeans(training\_features[, c(4, 6, 10)], centers = 4)  
  
cluster\_number <- data.frame(cluster\_number = km\_clusters$cluster)  
training\_features <- cbind(training\_features, cluster\_number)  
head(training\_features)

## 1 2 3 4 5 6 7  
## 1 -0.2931498 -0.6018545 -0.2952543 -1.1286719 -0.7493397 1.2960405 0.9333750  
## 2 -0.8273881 -0.1796831 0.2505435 -0.1964266 0.5485391 -0.4941846 1.1996304  
## 3 -0.5075720 -0.7276077 0.7963413 -0.8461733 -0.4590247 0.3426281 0.9714115  
## 4 -0.7874111 -0.2784892 0.6598919 -1.1286719 -0.7834944 0.2479310 1.3517764  
## 5 -1.3361866 -1.3114617 -0.8410522 -0.8697149 -0.8859585 -1.3026526 -0.0555739  
## 6 -1.3180152 1.5628966 -0.9775016 -1.1286719 -0.8005717 -1.1435358 -0.3218294  
## 8 9 10 11 12 13  
## 1 -0.10198705 -0.7522046 3.0168104 -0.2215892 -0.9009345 -0.2528934  
## 2 0.40736380 -0.8043398 -0.8364033 -0.2215892 -0.9009345 -0.2528934  
## 3 -1.27349399 -1.5603008 0.2127314 -0.2215892 -0.9009345 -0.2528934  
## 4 -0.05105196 -1.2474894 0.2012362 -0.2215892 -0.9009345 -0.2528934  
## 5 0.10175329 0.2383650 -0.5290965 -0.2215892 -0.9009345 -0.2528934  
## 6 0.50923397 0.6815146 -0.3735272 -0.2215892 -0.9009345 -0.2528934  
## 14 15 16 17 18 19  
## 1 -0.04795369 -0.03030767 -0.4989099 -0.09542723 4.2361634 -0.2200655  
## 2 -0.04795369 -0.03030767 -0.4989099 -0.09542723 4.2361634 -0.2200655  
## 3 -0.04795369 -0.03030767 2.0040633 -0.09542723 -0.2360265 -0.2200655  
## 4 -0.04795369 -0.03030767 2.0040633 -0.09542723 -0.2360265 -0.2200655  
## 5 -0.04795369 -0.03030767 2.0040633 -0.09542723 -0.2360265 -0.2200655  
## 6 -0.04795369 -0.03030767 -0.4989099 -0.09542723 -0.2360265 -0.2200655  
## 20 21 22 23 24 25  
## 1 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 2 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 3 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 4 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 5 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 6 -0.3149516 -0.2945671 -0.04463561 -0.06439589 -0.01749279 -0.01236832  
## 26 27 28 29 30 31  
## 1 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 2 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 3 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 4 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 5 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 6 -0.1542795 -0.1424332 -0.1418732 -0.1258904 -0.04953015 -0.1009841  
## 32 33 cluster\_number  
## 1 -0.01749279 -0.04632411 4  
## 2 -0.01749279 -0.04632411 1  
## 3 -0.01749279 -0.04632411 1  
## 4 -0.01749279 -0.04632411 1  
## 5 -0.01749279 -0.04632411 1  
## 6 -0.01749279 -0.04632411 1

## Exercises

### Exercise 1

It is good practice to center and scale the data before trying k-means clustering because features with higher magnitudes may dominate the distance calculations and have greater influence on the clusters. Centering and scaling ensures all features have equal influence on the determination of clusters by the model.

### Exercise 2

print(km\_clusters$size)

## [1] 3203 1293 1309 732

print(km\_clusters$centers)

## [,1] [,2] [,3]  
## 1 -0.4915537 -0.6082597 -0.39493456  
## 2 1.4698853 -0.3155127 -0.05481461  
## 3 -0.1849670 1.2632169 -0.17785123  
## 4 -0.1147452 0.9599216 2.14297536

### Exercise 3

for (x in c(4,6,10)) {  
 f\_means <- aggregate(training\_features[, x],  
 by=list(training\_features$cluster\_number),  
 FUN=mean)  
 print(paste("Means for feature", x))  
 print(f\_means)  
}

## [1] "Means for feature 4"  
## Group.1 x  
## 1 1 -0.4915537  
## 2 2 1.4698853  
## 3 3 -0.1849670  
## 4 4 -0.1147452  
## [1] "Means for feature 6"  
## Group.1 x  
## 1 1 -0.6082597  
## 2 2 -0.3155127  
## 3 3 1.2632169  
## 4 4 0.9599216  
## [1] "Means for feature 10"  
## Group.1 x  
## 1 1 -0.39493456  
## 2 2 -0.05481461  
## 3 3 -0.17785123  
## 4 4 2.14297536

### Exercise 4

library(clue)

## Warning: package 'clue' was built under R version 4.3.2

predictions <- cl\_predict(km\_clusters, newdata = test\_features[, c(4,6,10)])  
test\_features <- cbind(test\_features, predictions)  
head(test\_features)

##   
## [1,] -0.2931498 1.1714628 0.01939781 -0.6018545 -0.9521669 0.13469982  
## [2,] -0.8273881 0.4736821 0.80076682 -0.1796831 -0.2335773 -0.50304844  
## [3,] -0.5075720 0.6517616 0.66993294 -0.7276077 -0.9701317 0.46704751  
## [4,] -0.7874111 1.5130846 -0.42398368 -0.2784892 -1.3563736 1.55391427  
## [5,] -1.3361866 0.6117845 0.63722447 -1.3114617 -0.1886655 0.92514837  
## [6,] -1.3180152 0.3864595 1.03336038 1.5628966 1.6347556 -0.02698284  
##   
## [1,] -0.2952543 0.25054350 -0.02235542 -1.1286719 -0.2811762 2.7180170  
## [2,] 0.2505435 -0.02235542 -1.11395110 -0.1964266 -0.8697149 -1.2699212  
## [3,] 0.7963413 -1.38685002 -0.02235542 -0.8461733 -0.7049240 0.2555711  
## [4,] 0.6598919 1.34213919 1.47858865 -1.1286719 -0.1399269 -0.9544644  
## [5,] -0.8410522 0.38699296 -0.56815326 -0.8697149 -0.4224255 1.0983585  
## [6,] -0.9775016 -0.15880488 0.11409404 -1.1286719 -1.1286719 1.8140216  
##   
## [1,] -0.7493397 0.1728373 0.7022352 1.2960405 1.3321156 0.37290536  
## [2,] 0.5485391 -0.5102567 -0.8005717 -0.4941846 1.3417785 -0.09929147  
## [3,] -0.4590247 -0.7151850 0.3094561 0.3426281 0.2891597 -0.27451322  
## [4,] -0.7834944 1.0267049 -0.5102567 0.2479310 0.8953496 0.16160855  
## [5,] -0.8859585 -1.2445828 0.3436108 -1.3026526 1.4950976 -0.77183376  
## [6,] -0.8005717 -0.7151850 1.4024066 -1.1435358 1.1279049 0.08881422  
##   
## [1,] 0.9333750 0.4389005 0.7051560 -0.10198705 -1.0188186 1.1204550  
## [2,] 1.1996304 1.1615939 0.6671195 0.40736380 0.3564287 0.4073638  
## [3,] 0.9714115 0.3247910 -0.0555739 -1.27349399 0.4073638 1.4770006  
## [4,] 1.3517764 -1.2347052 -2.3758001 -0.05105196 -1.8337799 -1.0188186  
## [5,] -0.0555739 1.2376669 -0.5880848 0.10175329 0.3054936 1.1713901  
## [6,] -0.3218294 -0.8923768 0.9714115 0.50923397 -0.1019870 0.7639094  
##   
## [1,] -0.7522046 -0.85647507 0.003756417 3.0168104 -0.4969098 -0.2348176  
## [2,] -0.8043398 -0.75220458 -0.178716928 -0.8364033 -0.7436748 -0.8463659  
## [3,] -1.5603008 0.10802690 0.811852659 0.2127314 -0.7045909 -0.9383280  
## [4,] -1.2474894 -0.07444644 1.202866969 0.2012362 -0.1129678 -1.1996538  
## [5,] 0.2383650 -0.93467793 1.176799349 -0.5290965 -0.5735449 -0.6861985  
## [6,] 0.6815146 0.75971742 -0.413325514 -0.3735272 -0.9383280 0.4709918  
## predictions  
## [1,] -0.2215892 -0.2215892 -0.2215892 1  
## [2,] -0.2215892 -0.2215892 -0.2215892 1  
## [3,] -0.2215892 -0.2215892 -0.2215892 3  
## [4,] -0.2215892 -0.2215892 -0.2215892 3  
## [5,] -0.2215892 -0.2215892 -0.2215892 3  
## [6,] -0.2215892 -0.2215892 -0.2215892 2