ML Assignment 4

Dionna Attinson

2/17/2020

### (Part I): Implementing a Simple Prediction Pipeline

##### 1. Below I perform basic data cleaning and note which features are continuous, which are categorical and ensure they are being stored that way in the R dataset.

Data =   
 read.csv("./class4\_p1.csv") %>%   
 janitor::clean\_names() %>%   
 rename (PATCID = "x")

Data =   
Data%>%   
 mutate(  
 chronic1 = recode(chronic1,   
 "2" = "0",   
 "1" = "1"),  
 chronic3 = recode(chronic3,   
 "2" = "0",  
 "1" = "1"),  
 chronic4 = recode(chronic4,   
 "2" = "0",  
 "1" = "1"),  
 chronic1 = factor(chronic1, levels = c("0", "1")),  
 chronic3 = factor(chronic3, levels = c("0", "1")),  
 chronic4 = factor(chronic4, levels = c("0", "1")),  
 tobacco1 = recode(tobacco1,   
 "3" = "1",  
 "2" = "2",  
 "1" = "3"),  
 tobacco1 = factor(tobacco1, levels = c("1", "2", "3")),  
 alcohol1 = recode(alcohol1,   
 "3" = "1",  
 "2" = "2",  
 "1" = "3"),  
 alcohol1 = factor(alcohol1, levels = c("1", "2", "3")),  
 habits5 = recode(habits5,   
 "4" = "1",  
 "3" = "2",  
 "2" = "3",  
 "1" = "4"),  
 habits5 = factor(habits5, levels = c("1", "2", "3", "4")),  
 habits7 = recode(habits7,   
 "5" = "1",  
 "4" = "2",  
 "3" = "3",  
 "2" = "4",  
 "1" = "5"),  
 habits7 = factor(habits7, levels = c("1", "2", "3", "4", "5")),   
 agegroup = factor(agegroup, levels = c("1", "2", "3", "4")),   
 dem3 = recode(dem3,   
 "2" = "0",   
 "1" = "1"),  
 dem3 = factor(dem3, levels = c("0", "1")),  
 dem4 = recode(dem4,   
 "2" = "0",   
 "1" = "1"),  
 dem4 = factor(dem4, levels = c("0", "1")),  
 dem8 = recode(dem8,   
 "2" = "0",   
 "1" = "1"),  
 dem8 = factor(dem8, levels = c("0", "1")),   
 povertygroup = recode(povertygroup,  
 "1" = "1",  
 "2" = "2",  
 "3" = "3",   
 "4" = "4",  
 "5" = "5"),  
 povertygroup = factor(povertygroup, levels = c("1", "2", "3", "4", "5"))) %>%   
 drop\_na()  
 str(Data)

## 'data.frame': 2012 obs. of 17 variables:  
## $ PATCID : int 1324 1325 1326 1327 1328 1330 1331 1332 1333 1334 ...  
## $ chronic1 : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 1 1 1 ...  
## $ chronic3 : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 1 1 1 ...  
## $ chronic4 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...  
## $ bmi : num 19.9 33.7 24.2 22.7 25.7 ...  
## $ tobacco1 : Factor w/ 3 levels "1","2","3": 1 1 1 2 1 1 1 1 1 1 ...  
## $ alcohol1 : Factor w/ 3 levels "1","2","3": 1 1 1 2 1 1 1 2 1 2 ...  
## $ gpaq8totmin : int 0 0 30 0 0 0 0 0 0 0 ...  
## $ gpaq11days : int 7 7 4 6 4 5 0 7 7 6 ...  
## $ habits5 : Factor w/ 4 levels "1","2","3","4": 3 3 3 3 3 4 3 3 2 4 ...  
## $ habits7 : Factor w/ 5 levels "1","2","3","4",..: 4 1 3 2 2 4 4 3 2 5 ...  
## $ agegroup : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 3 2 2 2 3 3 ...  
## $ dem3 : Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 2 1 2 ...  
## $ dem4 : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 2 2 1 ...  
## $ dem8 : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 2 2 1 1 ...  
## $ povertygroup: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 5 2 4 3 2 2 2 ...  
## $ healthydays : int 30 27 30 30 23 30 30 30 0 27 ...

### Next, I created the test/training datasets with a 70:30 split

set.seed(100)  
train.indices<-createDataPartition(y=Data$healthydays, p=0.7, list=FALSE)  
  
training <- Data[train.indices,]  
testing <- Data[-train.indices,]

### Next, I fit two prediction models using different subsets of the features in the training data.

set.seed(100)  
model.1 <- lm(healthydays ~ chronic1 + chronic3 + chronic4 + povertygroup + bmi + agegroup + dem3 + dem4 + dem8, data=training)  
summary(model.1)

##   
## Call:  
## lm(formula = healthydays ~ chronic1 + chronic3 + chronic4 + povertygroup +   
## bmi + agegroup + dem3 + dem4 + dem8, data = training)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -29.8680 -0.2972 1.8736 3.7997 11.0876   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 29.97275 1.27180 23.567 < 2e-16 \*\*\*  
## chronic11 -1.65163 0.49797 -3.317 0.000934 \*\*\*  
## chronic31 -1.27323 0.65198 -1.953 0.051036 .   
## chronic41 -3.96203 0.85075 -4.657 3.51e-06 \*\*\*  
## povertygroup2 1.31253 0.64398 2.038 0.041721 \*   
## povertygroup3 2.63851 0.64161 4.112 4.15e-05 \*\*\*  
## povertygroup4 3.14436 0.67464 4.661 3.45e-06 \*\*\*  
## povertygroup5 3.13381 0.64646 4.848 1.39e-06 \*\*\*  
## bmi -0.06765 0.03476 -1.947 0.051785 .   
## agegroup2 -0.92684 0.90436 -1.025 0.305609   
## agegroup3 -2.65670 0.91431 -2.906 0.003722 \*\*   
## agegroup4 -3.64931 0.97213 -3.754 0.000181 \*\*\*  
## dem31 -0.65344 0.40759 -1.603 0.109126   
## dem41 -1.28320 0.51655 -2.484 0.013103 \*   
## dem81 -0.24308 0.43665 -0.557 0.577826   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 7.39 on 1394 degrees of freedom  
## Multiple R-squared: 0.1167, Adjusted R-squared: 0.1078   
## F-statistic: 13.15 on 14 and 1394 DF, p-value: < 2.2e-16

model.2<-lm(healthydays ~ tobacco1 + alcohol1 + habits5 + habits7 + gpaq8totmin + gpaq11days + agegroup + bmi, data=training)  
summary(model.2)

##   
## Call:  
## lm(formula = healthydays ~ tobacco1 + alcohol1 + habits5 + habits7 +   
## gpaq8totmin + gpaq11days + agegroup + bmi, data = training)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -30.0264 -0.4868 1.9766 3.5870 13.4947   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 20.439645 1.827782 11.183 < 2e-16 \*\*\*  
## tobacco12 -1.271939 0.907808 -1.401 0.161404   
## tobacco13 -0.336466 0.653184 -0.515 0.606553   
## alcohol12 0.539660 0.433399 1.245 0.213276   
## alcohol13 2.142061 1.167660 1.834 0.066795 .   
## habits52 2.069646 1.047989 1.975 0.048480 \*   
## habits53 4.772394 1.001887 4.763 2.10e-06 \*\*\*  
## habits54 4.090671 1.047345 3.906 9.84e-05 \*\*\*  
## habits72 2.161896 1.032648 2.094 0.036481 \*   
## habits73 5.213446 0.997715 5.225 2.00e-07 \*\*\*  
## habits74 5.497660 1.035974 5.307 1.30e-07 \*\*\*  
## habits75 6.504926 1.110151 5.859 5.79e-09 \*\*\*  
## gpaq8totmin -0.001680 0.002087 -0.805 0.420990   
## gpaq11days 0.097021 0.075015 1.293 0.196098   
## agegroup2 -0.699521 0.890982 -0.785 0.432521   
## agegroup3 -3.077448 0.890972 -3.454 0.000569 \*\*\*  
## agegroup4 -4.331885 0.931573 -4.650 3.63e-06 \*\*\*  
## bmi -0.021217 0.034589 -0.613 0.539707   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 7.297 on 1391 degrees of freedom  
## Multiple R-squared: 0.1405, Adjusted R-squared: 0.13   
## F-statistic: 13.38 on 17 and 1391 DF, p-value: < 2.2e-16

### Using RMSE as a measure of fit, the first model is the preferred model of prediction.

* This model would be useful when trying to understand in a clinical setting how chronic disease could predict quality of life (healthy days).

rmse(model.1, testing)

## [1] 7.216415

rmse(model.2, testing)

## [1] 7.255706

## (Part II): Conducting an Unsupervised Analysis

### Using the dataset from the Group assignment Part IIb (USArrests), we identify clusters using hierarchical analysis.

library(devtools)

## Loading required package: usethis

library(ggbiplot)

## Loading required package: plyr

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following object is masked from 'package:purrr':  
##   
## compact

## Loading required package: scales

##   
## Attaching package: 'scales'

## The following object is masked from 'package:viridis':  
##   
## viridis\_pal

## The following object is masked from 'package:purrr':  
##   
## discard

## The following object is masked from 'package:readr':  
##   
## col\_factor

## Loading required package: grid

library(stats)  
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(cluster)  
  
data(USArrests)

### First, we loaded in the dataset, checked means and SDs to determine if scaling is necessary. However, scaling was not necessary.

USArrests.nomiss <-na.omit(USArrests)  
  
colMeans(USArrests.nomiss, na.rm=TRUE)

## Murder Assault UrbanPop Rape   
## 7.788 170.760 65.540 21.232

apply(USArrests.nomiss, 2, sd, na.rm=TRUE)

## Murder Assault UrbanPop Rape   
## 4.355510 83.337661 14.474763 9.366385

### Hierarchical Clustering: Agglomeration Method

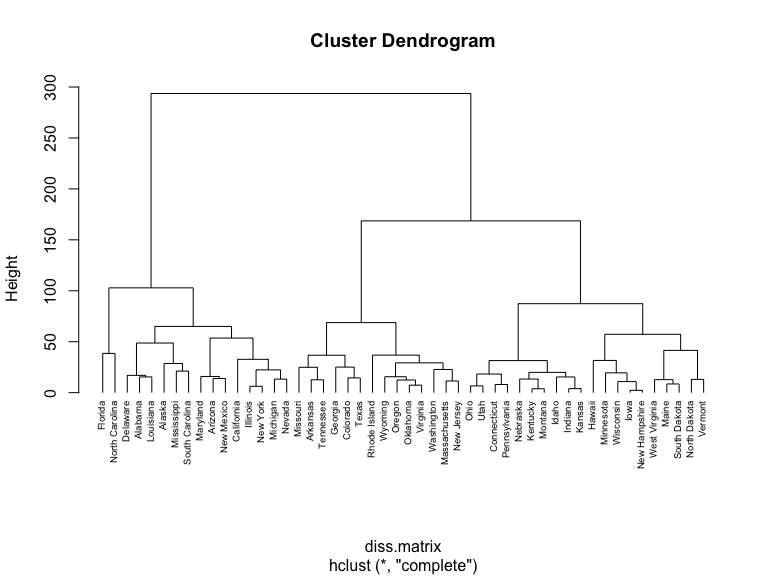
I used Euclidean distance for the dissimilarity matrix and completed 3 difference agglomeration methods: complete, average, and single.

# Create Dissimilarity matrix  
diss.matrix <- dist(USArrests.nomiss, method = "euclidean")

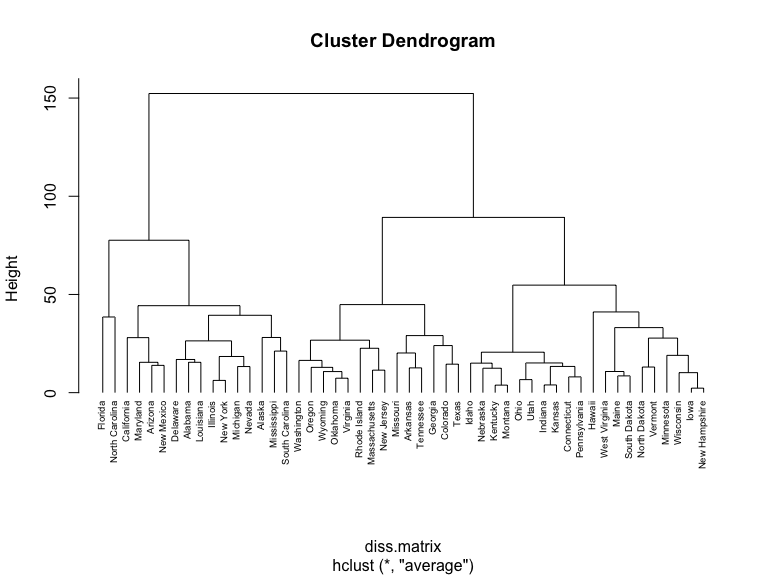
#### Complete Agglomeration Method

For each pair of clusters, the algorithm computes and merges them to minimize the maximum distance between the clusters (in other words, the distance of the farthest elements). It tends to produce more compact clusters.

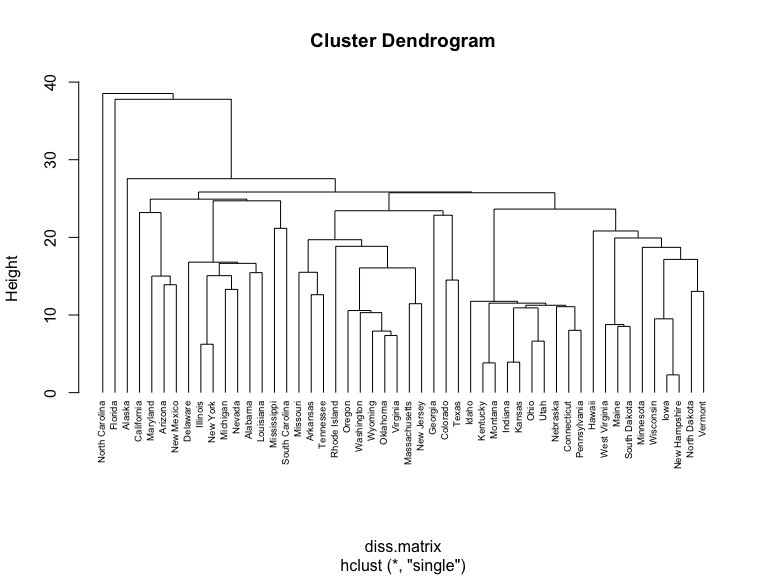
set.seed(100)  
# Hierarchical clustering using Complete Linkage  
hc\_complete <- hclust(diss.matrix, method = "complete")  
# Plot the obtained dendrogram  
plot(hc\_complete, cex = 0.6, hang = -1)

 #### Average Agglomeration Method It’s similar to complete linkage, but in this case, the algorithm uses the average distance between the pairs of clusters

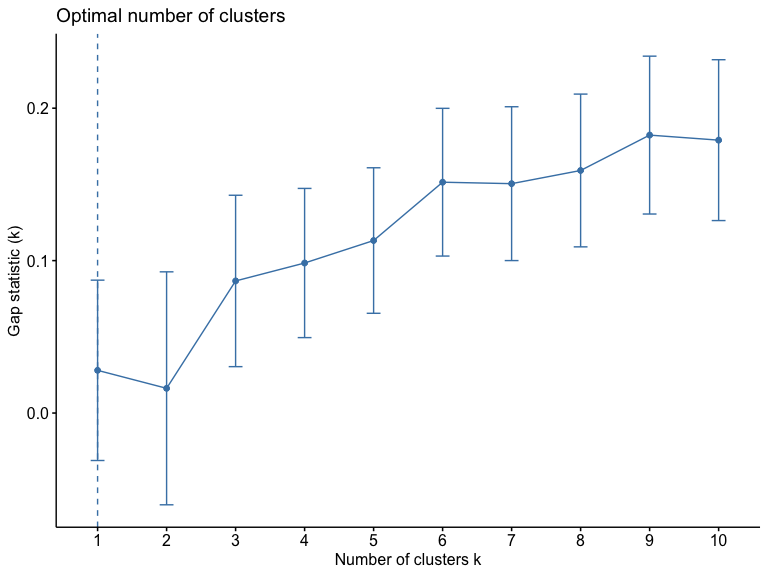
set.seed(100)  
# Hierarchical clustering using Average Linkage  
hc\_average <- hclust(diss.matrix, method = "average")  
# Plot the obtained dendrogram  
plot(hc\_average, cex = 0.6, hang = -1)

 #### Single Agglomeration Method It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the smallest of these dissimilarities as a linkage criterion. It tends to produce long, “loose” clusters.

set.seed(100)  
# Hierarchical clustering using Single Linkage  
hc\_single <- hclust(diss.matrix, method = "single")  
# Plot the obtained dendrogram  
plot(hc\_single, cex = 0.6, hang = -1)



gap\_stat <- clusGap(USArrests.nomiss, FUN = hcut, nstart = 25, K.max = 10, B = 50)  
fviz\_gap\_stat(gap\_stat)



#### Difference Between Dendrograms

It is noticeable to mention that the single agglomeration method produced the most clusters, while the average and complete agglomeration methods produced less and more compact clusters. #### Using the gap clusters below, we see that 4 is the optimal number of clusters.

#### Below, I describe one research question that can be addressed using the newly identified clusters.

What states should be prioritized to receive a community-based violence prevention intervention based on their rates of murder, assault, and rape.