Final Assignment

Holly Finertie - HF2379

Due: 5/1/2020

## Part 1: Neighborhood Environments in NYC

### Question 1: Run Unsupervised Learning Analysis

#### Load Data

nyc\_data = read\_csv("./nyc\_env\_correct.csv") %>%   
 janitor::clean\_names() %>%   
 dplyr::rename(old\_housing = pct\_pre1960\_housing,   
 npl\_site = npl\_site\_count\_wt\_by\_dist)

## Parsed with column specification:  
## cols(  
## `Pct Low Inc` = col\_double(),  
## `Diesel Conc` = col\_double(),  
## `NPL Site Count, wt by dist` = col\_double(),  
## `Pct pre1960 Housing` = col\_double()  
## )

#### Scale Data

# mean and sd are different  
summary(nyc\_data)

## pct\_low\_inc diesel\_conc npl\_site old\_housing   
## Min. :0.0000 Min. :0.6697 Min. :0.04633 Min. :0.0000   
## 1st Qu.:0.2129 1st Qu.:2.2887 1st Qu.:0.08758 1st Qu.:0.5526   
## Median :0.3488 Median :4.5973 Median :0.11812 Median :0.7338   
## Mean :0.3664 Mean :3.9260 Mean :0.15732 Mean :0.6720   
## 3rd Qu.:0.5010 3rd Qu.:5.0290 3rd Qu.:0.16970 3rd Qu.:0.8506   
## Max. :1.0000 Max. :6.1438 Max. :5.35836 Max. :1.0000

apply(nyc\_data, 2, sd, na.rm = TRUE)

## pct\_low\_inc diesel\_conc npl\_site old\_housing   
## 0.1932175 1.5077004 0.1724710 0.2312597

# scaling  
nyc\_data = scale(nyc\_data)

#### K-Means Clustering

# Visually test out 5 clusters  
clusters = kmeans(nyc\_data, 5, nstart = 25)  
str(clusters)

## List of 9  
## $ cluster : int [1:2140] 2 2 2 3 2 1 3 2 3 1 ...  
## $ centers : num [1:5, 1:4] 0.947 -0.879 1.078 -1.233 -0.615 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:5] "1" "2" "3" "4" ...  
## .. ..$ : chr [1:4] "pct\_low\_inc" "diesel\_conc" "npl\_site" "old\_housing"  
## $ totss : num 8556  
## $ withinss : num [1:5] 515 569 335 144 1064  
## $ tot.withinss: num 2627  
## $ betweenss : num 5929  
## $ size : int [1:5] 645 406 246 2 841  
## $ iter : int 3  
## $ ifault : int 0  
## - attr(\*, "class")= chr "kmeans"

fviz\_cluster(clusters, data = nyc\_data)



# Find optimal using gap statistic  
gap\_stat = clusGap(nyc\_data, FUN = kmeans, nstart = 25, K.max = 9, B = 50)

## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations  
  
## Warning: did not converge in 10 iterations

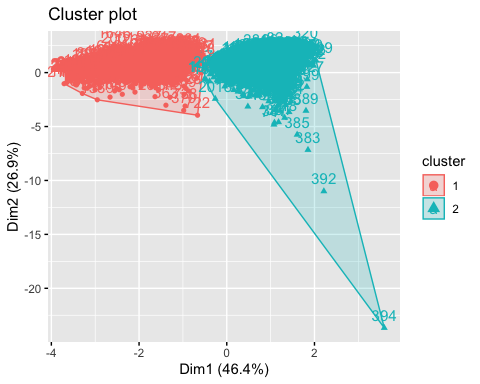
print(gap\_stat, method = "firstmax")

## Clustering Gap statistic ["clusGap"] from call:  
## clusGap(x = nyc\_data, FUNcluster = kmeans, K.max = 9, B = 50, nstart = 25)  
## B=50 simulated reference sets, k = 1..9; spaceH0="scaledPCA"  
## --> Number of clusters (method 'firstmax'): 2  
## logW E.logW gap SE.sim  
## [1,] 7.153008 8.817573 1.664565 0.005276067  
## [2,] 6.834087 8.523391 1.689304 0.006127729  
## [3,] 6.682157 8.363763 1.681606 0.005761747  
## [4,] 6.660705 8.216858 1.556153 0.004971593  
## [5,] 6.596593 8.129959 1.533367 0.003994658  
## [6,] 6.491211 8.054785 1.563574 0.005484282  
## [7,] 6.426818 8.005569 1.578751 0.004638145  
## [8,] 6.372681 7.959739 1.587057 0.004723523  
## [9,] 6.346023 7.918382 1.572359 0.004766930

# Visualize optimal clusters  
clusters\_new = kmeans(nyc\_data, 2, nstart = 25)  
str(clusters\_new)

## List of 9  
## $ cluster : int [1:2140] 1 1 1 1 1 2 1 1 1 2 ...  
## $ centers : num [1:2, 1:4] -0.1414 0.0618 -1.4423 0.6306 -0.0588 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:2] "1" "2"  
## .. ..$ : chr [1:4] "pct\_low\_inc" "diesel\_conc" "npl\_site" "old\_housing"  
## $ totss : num 8556  
## $ withinss : num [1:2] 1516 3572  
## $ tot.withinss: num 5088  
## $ betweenss : num 3468  
## $ size : int [1:2] 651 1489  
## $ iter : int 1  
## $ ifault : int 0  
## - attr(\*, "class")= chr "kmeans"

fviz\_cluster(clusters\_new, data = nyc\_data)



### Question 2: Describe Outputs

First, I imported 2,140 rows of data on diesel exhaust, prosimity to NPL sites, proportion of housing built before 1960, and proprotion of the neighborhood with household income lower than the city median for each NYC census tract. Because the means and standard deviations varied, I scaled the data to make it compatible for clustering analysis.

Once the data was imported and cleaned, I decided to run K-Means clustering analysis. I chose this method instead of hierarchical clustering because our data was large and, visually, the K-Means plot would be more useful than a dendogram.

Initially, we visually assessed 5 clusters across the 2,140 inputs with their 4 corresponding variables. Then, we ran a gap statistic analysis to determine the optimal number of clusters. This analysis determined that 2 clusters best fit the data. While there are a few notalbe outliers (observation 392 and 394), we can visually confirm that 2 clusters fit the data appropriately in our Cluster Plot.

Overall, this analysis shows us that there are 2 main clusters of census tracts based on our 4 variables of interests.

### Question 3: Research Question

I would include the cluster assignement as a confounding variable in an explanatory analysis to see if race explains adverse health outcomes.

**Questions**:

* Preliminary Analysis: Is race associated with adverse health events (yes/no) in NYC?
* Secondary Analysis: Is race associated with adverse health events in NYC if adjusted by a confounding variable created with information on diesel exhaust, proximity to NPL sites, housing before 1960, and low household income by census tract?

## Part 2: Supervised Adventure (B)

### Choice B

Below I used two different algorithms for feature selection: LASSO and Random Forrest. The two algorithms provided very different results for variable importance; however, their overall performance was similar.

LASSO selected all variable except weight in kg and BMI in kg/m2. In total, LASSO identified 11 variables as important in the model and had an RMSE = 19.95 and RSquare = 0.10 using the test data.

RF provided two different variable importance graphs: %IncMSE and IncNodePurity. Both had varying results. Overall, RF had an RMSE = 20.33 and RSquare = 0.076.

* %IncMSE: Identified waist circumference, BMI, age, serum copper, weight, height, gender and race as important variables.
* IncNodePurtiy: Identified waist circumference, age, and all serums as important variables.

The IncNodePurity variable selection better matched LASSO becasue it did not have BMI or weight as one of the top variables.

#### Load Data

glucose\_data = read\_csv("./glucose.csv") %>%   
 janitor::clean\_names() %>%   
 drop\_na() %>%  
 mutate(  
 riagendr = as.factor(riagendr),   
 ridreth1 = as.factor(ridreth1)  
 ) %>%   
 select(-seqn)

## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:  
## cols(  
## X1 = col\_double(),  
## SEQN = col\_double(),  
## LBXSCU = col\_double(),  
## LBXSSE = col\_double(),  
## LBXMG = col\_double(),  
## LBXLYC = col\_double(),  
## LBXSZN = col\_double(),  
## BMXWT = col\_double(),  
## BMXHT = col\_double(),  
## BMXBMI = col\_double(),  
## BMXWAIST = col\_double(),  
## RIAGENDR = col\_double(),  
## RIDAGEYR = col\_double(),  
## RIDRETH1 = col\_double(),  
## LBXSGL = col\_double()  
## )

#### Testing and Training Data Sets

train = glucose\_data %>% sample\_frac(.7)  
test = anti\_join(glucose\_data, train, by = 'x1') %>%   
 select(-x1)  
  
train = train %>%   
 select(-x1)

#### Algorithm 1: LASSO

lambda = 10^seq(-3,1, length = 100)  
  
LASSO = train(  
 lbxsgl ~.,   
 data = train,  
 method = "glmnet",  
 trControl = trainControl("cv", number = 10),   
 tuneGrid = expand.grid(alpha = 1, lambda = lambda))

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =  
## trainInfo, : There were missing values in resampled performance measures.

LASSO$bestTune

## alpha lambda  
## 54 1 0.1384886

coef(LASSO$finalModel, LASSO$bestTune$lambda)

## 16 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 87.0164835403  
## lbxscu -0.0186163206  
## lbxsse 0.0253901052  
## lbxmg .   
## lbxlyc 0.0206617120  
## lbxszn -0.0201688208  
## bmxwt 0.0005250333  
## bmxht -0.1131300926  
## bmxbmi .   
## bmxwaist 0.2321088263  
## riagendr2 -2.6160028300  
## ridageyr 0.1612409099  
## ridreth12 1.2439200175  
## ridreth13 -1.7972712403  
## ridreth14 -1.6664278679  
## ridreth15 0.8325838284

LASSO\_predict = LASSO %>% predict(test)  
  
# Model prediction performance  
data.frame(  
 RMSE = RMSE(LASSO\_predict, test$lbxsgl),  
 Rsquare = R2(LASSO\_predict, test$lbxsgl))

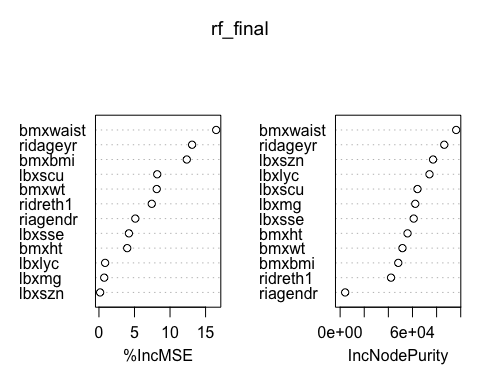
## RMSE Rsquare  
## 1 19.94964 0.1029936

#### Algorithm 2: Random Forrest

mtry\_options = c((ncol(train) - 1),   
 (ncol(train) - 1)/2,   
 sqrt(ncol(train) - 1))  
  
mtry\_grid = expand.grid(mtry = mtry\_options)  
  
tree\_options = seq(200, 600, by = 200)  
  
rf\_results = list()  
for (ntree in tree\_options) {  
 rf\_glucose = train(lbxsgl ~ .,   
 data = train,  
 method = "rf",   
 tuneGrid = mtry\_grid,   
 importance = TRUE,   
 metric = "RMSE",  
 ntree = ntree)  
  
 index = toString(ntree)  
 rf\_results[[index]] = rf\_glucose$results  
}  
  
rf\_output = bind\_rows(rf\_results, .id = "ntrees")  
  
best\_tune = rf\_output[which.max(rf\_output[,"RMSE"]),]  
  
rf\_final = randomForest(lbxsgl ~ .,   
 data = train,   
 mtry = best\_tune$mtry,   
 importance = TRUE,   
 ntree = as.numeric(best\_tune$ntrees))  
  
rf\_final

##   
## Call:  
## randomForest(formula = lbxsgl ~ ., data = train, mtry = best\_tune$mtry, importance = TRUE, ntree = as.numeric(best\_tune$ntrees))   
## Type of random forest: regression  
## Number of trees: 400  
## No. of variables tried at each split: 12  
##   
## Mean of squared residuals: 309.5113  
## % Var explained: 4.77

varImpPlot(rf\_final)



varImp(rf\_final)

## Overall  
## lbxscu 8.1989259  
## lbxsse 4.2185132  
## lbxmg 0.7393213  
## lbxlyc 0.8672409  
## lbxszn 0.1652377  
## bmxwt 8.1287945  
## bmxht 3.9744363  
## bmxbmi 12.3575321  
## bmxwaist 16.4900885  
## riagendr 5.1141907  
## ridageyr 13.1032940  
## ridreth1 7.4304444

RF\_predict = rf\_final %>% predict(test)  
  
data.frame(  
 RMSE = RMSE(RF\_predict, test$lbxsgl),  
 Rsquare = R2(RF\_predict, test$lbxsgl))

## RMSE Rsquare  
## 1 20.33759 0.07556304

## Part 3: Ethical Considerations

### Question 1:

Underrepresentation of certain populations is a huge potential risk. If researchers use social media data to develope intervetions, then they might only be targeting those who are frequent users of social media. Using the most extreme example of Twitter, we can see that using social media data will not be generalizable to the many populations.

A Pew Research Center [publication](https://www.pewresearch.org/internet/2019/04/24/sizing-up-twitter-users/) in 2019 described the demographics of Twitter. In general, Twitter users are younger, more educated, and wealthier. Those are just the users. If you consider that 10% of users generate 90% of the content, then the differences are more extreme.

If we use algorithms based on social media data to identify ways to increase early detection and prevention stratgies, these interventions will work best on younger, educated, and weathly populations. This will further increase disparities and health inequities.

### Question 2:

Solving underrepresentation and bias in data is extremely difficult. If researchers had access to demographic data for each social media user, they could possibly create a data set that is representative of the population and attempt to balance the data. However, this is most likely not feasible as most social media users do not report their demographic data.

Instead, researchers should use multiple data sources (Facebook, Twitter, Tumblr, etc) because each site has different user populations. By creating a diversified data set, there will be less underrepresentation. However, this solution comes with its own issues. Trying to uniformly create one data set from multiple social media sources could be next to impossible due to their differing nature.

Overall, it is almost impossible to solve underrepresentation in data; however, researchers can diversify their data sources and know that there will be limitations.