Food Desert Code

Kaitlyn Keebaugh, Matthew Pergolski, Miranda Braman, Victor Yamaykin
IST707

Section 1: Rule Mining

LIBRARIES

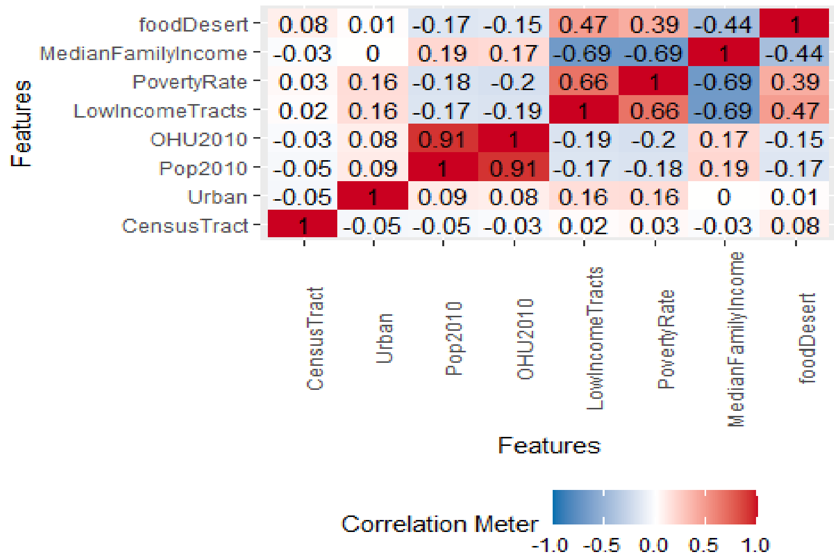
```
library(knitr)
library(magrittr)
library(tidyverse)
library(DataExplorer)
library(ggplot2)
library(arules)
library(arulesViz)
library(ROSE)
library(randomForest)
library(caret)
library(e1071)
library(dplyr)
library(RColorBrewer)
library(neuralnet)
library(arm)
library(rpart.plot)
library(rpart)
library(rattle)
library(plotly)
library(tidyverse)
library(cluster)
library(factoextra)
library(dendextend)
library(DescTools)
```

LOAD PACKAGES & DATA SET

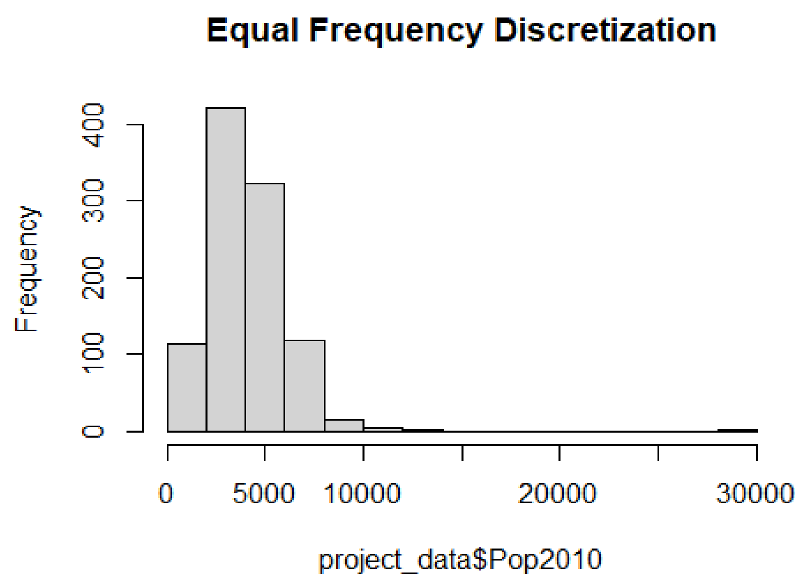
```
    # Load Data Set
    project_data <- read.csv("C:\\Users\\kkeeb\\Documents\\Balanced_FoodDeser
tComparisonData_2006_2019.csv",stringsAsFactors = TRUE)

introduce(project_data)
table(project_data$foodDesert)

# Oversampling test
set.seed(42)
barplot(prop.table(table(project_data$foodDesert)),
        col = rainbow(2),
        ylim = c(0,0.7),
        main = "Class Distribution")
```

```
under_sampled <-ovun.sample(foodDesert~., data=project_data,seed=1, method="u
nder")$data
```

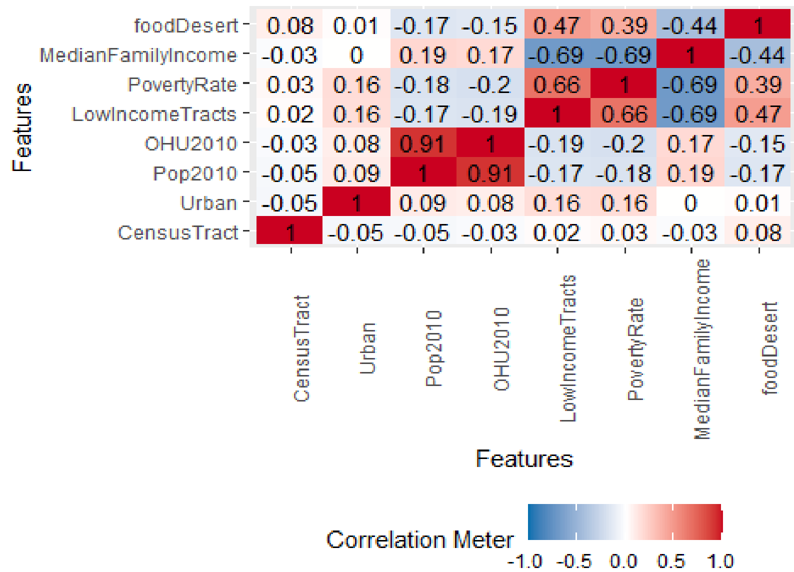

```
table(under_sampled$foodDesert)
plot_correlation(project_data, maxcat=5L)
```



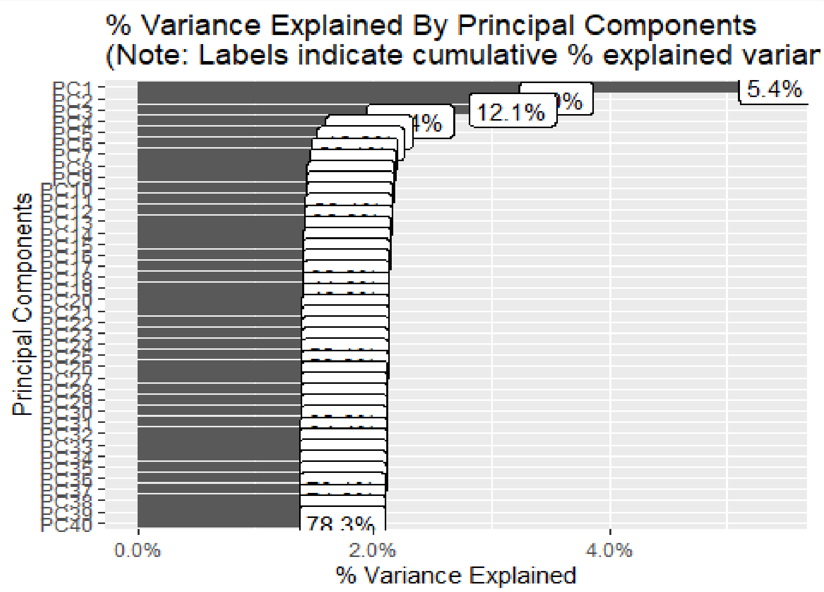

```
table(discretize(project_data$Pop2010, breaks=3))
hist(project_data$Pop2010, breaks = 12, main = "Equal Frequency Discretizatio
n")
```

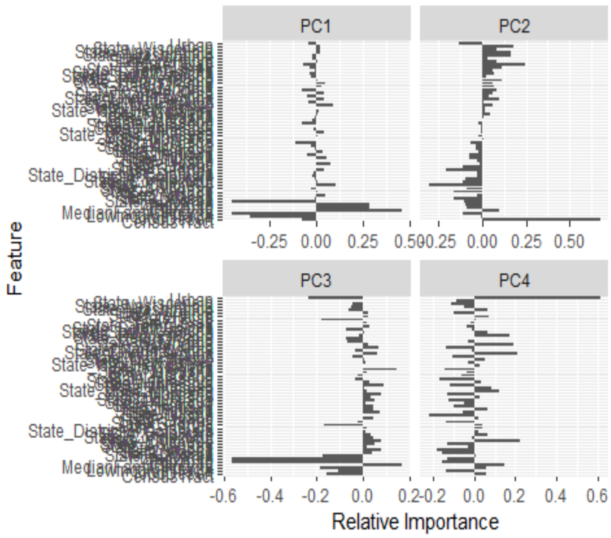



```
##
##      [36,3.13e+03) [3.13e+03,4.73e+03) [4.73e+03,2.97e+04]
##                333                 333                 334


ARM_ready_data <- discretizeDF(project_data, methods = list(
  Pop2010 = list(method = "frequency", breaks = 5,
                 labels = c("very low", "low", "medium", "high", "very high")
),
  OHU2010 = list(method = "frequency", breaks = 3,
                 labels = c("low", "medium", "high")),
  PovertyRate = list(method = "frequency", breaks = 3,
                     labels = c("low", "medium", "high")),
  MedianFamilyIncome = list(method = "frequency", breaks = 3,
                            labels = c("low", "medium", "high"))),default = l
ist(method = "none"))

# Turn the data set to transactions
tid <- as.character(ARM_ready_data$CensusTract)
ARM_ready_data$id <- NULL
transactions <- as(ARM_ready_data, "transactions")
transactionInfo(transactions)[["transactionID"]] <- tid

str(project_data)
plot_str(project_data)
introduce(project_data)

plot_correlation(project_data, maxcat=5L)
```

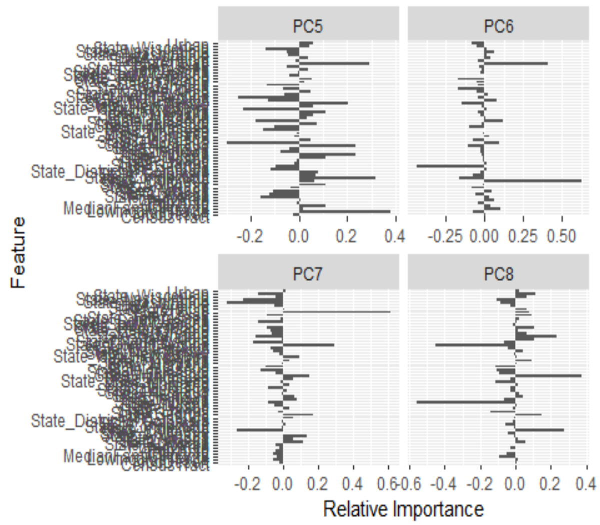



```
plot_prcomp(na.omit(project_data), variance_cap = 0.8, nrow = 2L, ncol = 2L)
```

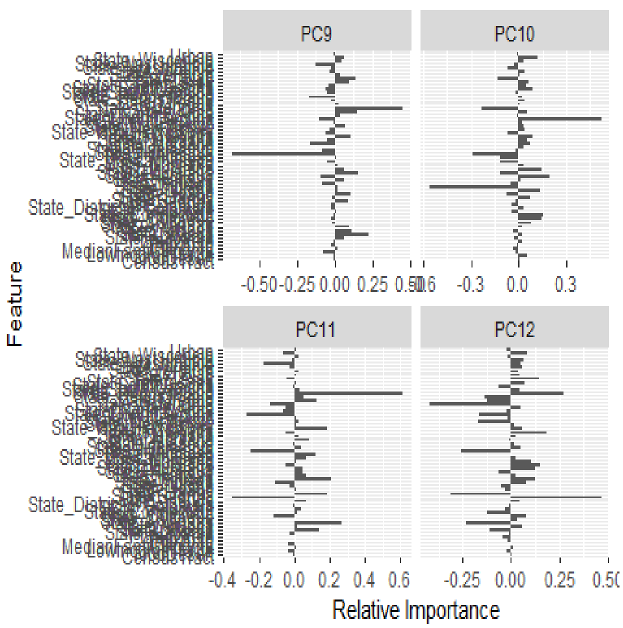




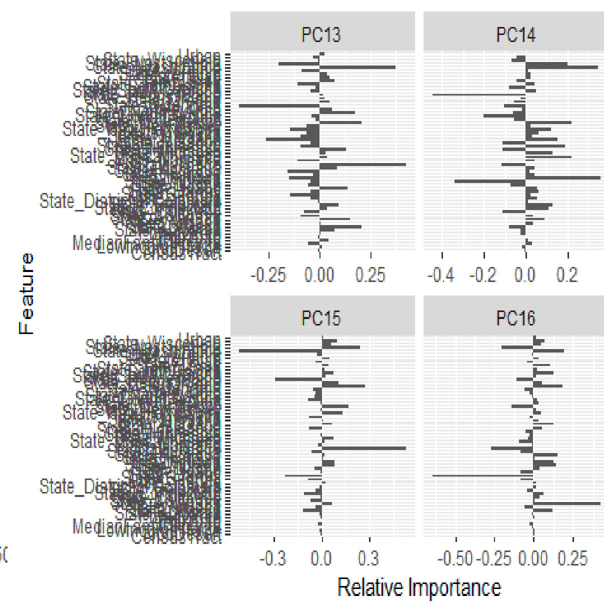
Page 1



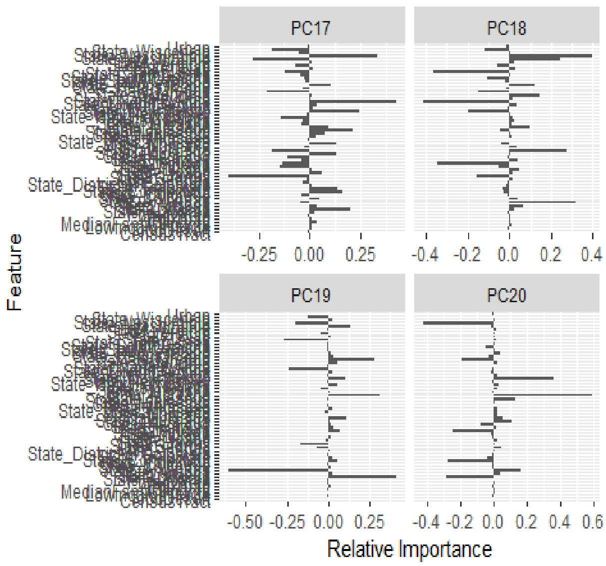
Page 2



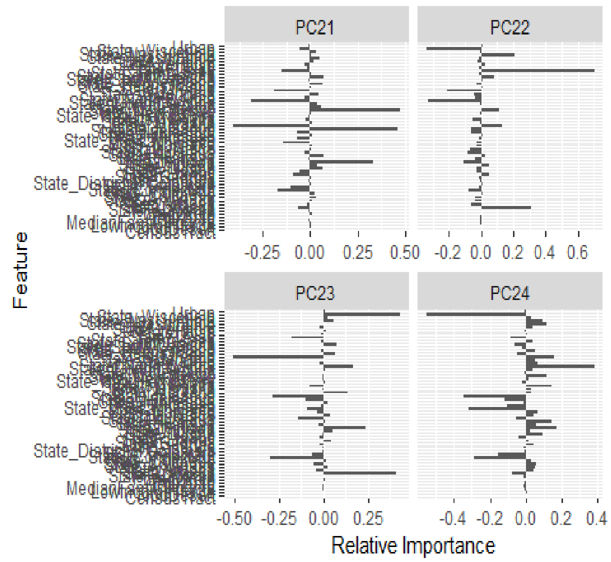
Page 3



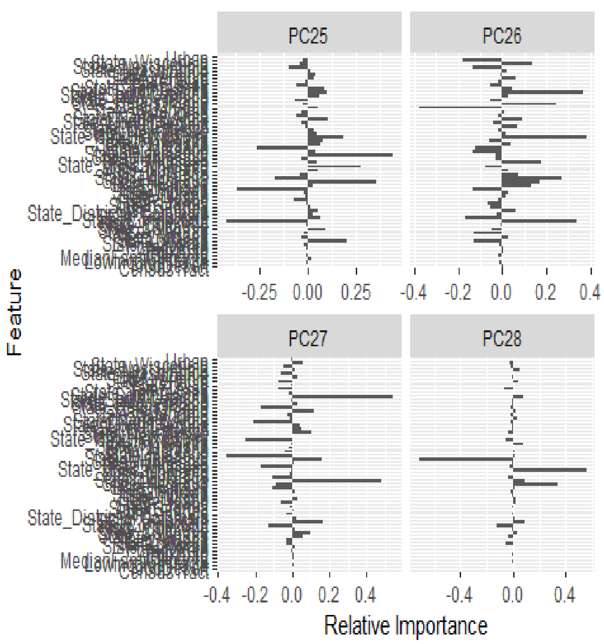
Page 4



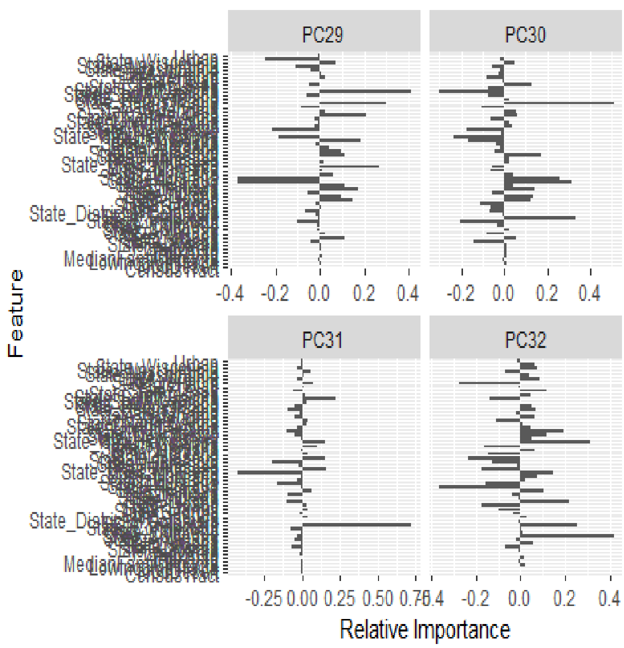
Page 5



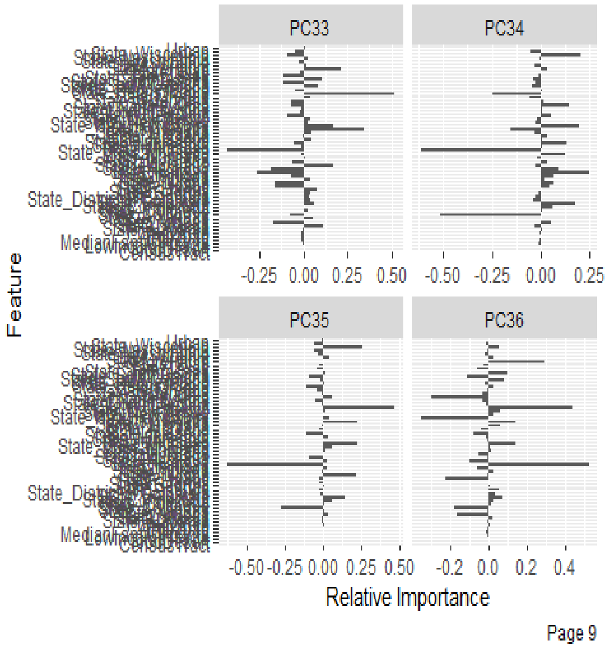
Page 6

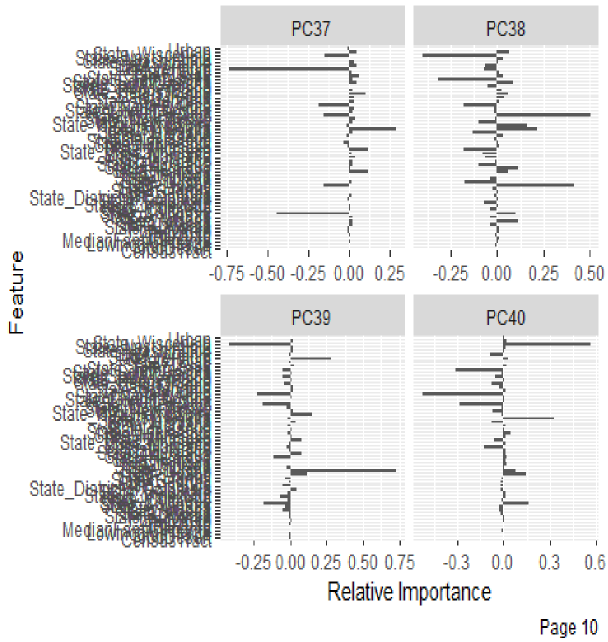


Page 7



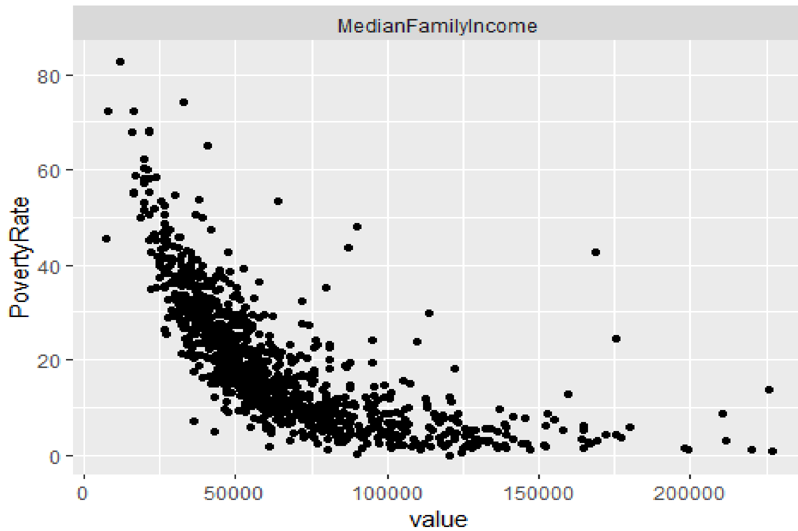
Page 8





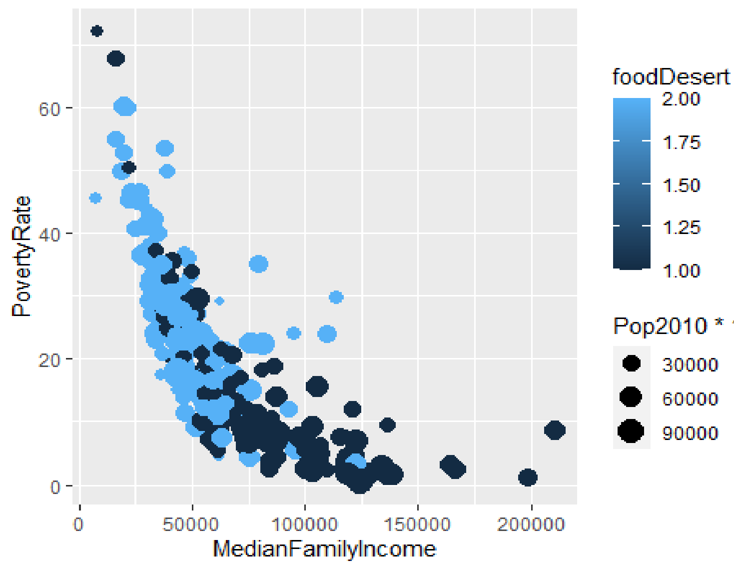

```
plot(project_data$PovertyRate, project_data$MedianFamilyIncome)
```


```
plot_scatterplot(project_data[, c("PovertyRate", "MedianFamilyIncome")], by="
PovertyRate", sampled_rows = 1000L)
```

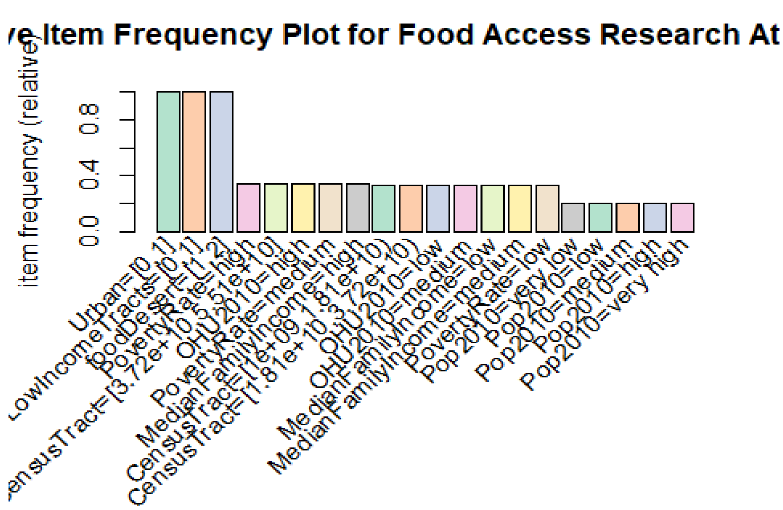



```
p <- ggplot(project_data[sample(nrow(project_data), 250), ], aes(MedianFamily
Income, PovertyRate, color=foodDesert, size = Pop2010*10,
                                                                 main="Food")
) + geom_point(na.rm = T)
##   CensusTract          State             County Urban   Pop2010 OHU2010
## 1 17031381800       Illinois        Cook County     1  very low     low
## 2 42003483800   Pennsylvania    Allegheny County     1       low  medium
## 3 13117130612        Georgia      Forsyth County     1 very high  medium
## 4 45091060905 South Carolina        York County     1 very high    high
## 5  6037501504     California Los Angeles County     1    medium  medium
## 6 48141010336          Texas     El Paso County     1 very high    high
##   LowIncomeTracts PovertyRate MedianFamilyIncome foodDesert
## 1               1      medium             medium          1
## 2               1        high                low          1
## 3               0         low               high          1
## 4               0      medium             medium          1
## 5               1        high                low          1
## 6               0      medium             medium          1

## 'data.frame':    1000 obs. of  10 variables:
##  $ CensusTract       : num  1.70e+10 4.20e+10 1.31e+10 4.51e+10 6.04e+09 .
..
##  $ State             : chr  "Illinois" "Pennsylvania" "Georgia" "South Car
olina" ...
##  $ County            : chr  "Cook County" "Allegheny County" "Forsyth Coun
ty" "York County" ...
##  $ Urban             : int  1 1 1 1 1 1 0 1 1 1 ...
##  $ Pop2010           : int  1188 3165 5614 8916 3592 6638 2128 1193 2632 2
300 ...
##  $ OHU2010           : int  591 1485 1740 3663 1588 1986 822 606 815 1066
...
##  $ LowIncomeTracts   : int  1 1 0 0 1 0 0 1 1 1 ...
##  $ PovertyRate       : num  23.4 35.2 1.8 17.3 24.3 13.7 10.7 41.9 33 20.4
```

```
...
##  $ MedianFamilyIncome: int  60938 26336 151944 60625 45208 56510 85703 254
41 39958 56797 ...
##  $ foodDesert        : int  1 1 1 1 1 1 1 1 1 1 ...

##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 1000      10                2                  8                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0          1000              10000        93816
```




```
# Look at relative frequency plot to see how many times these items have
#appeared as compared to others

itemFrequencyPlot(transactions, topN=20, type="relative", col=brewer.pal(8,
'Pastel2'),main="Relative Item Frequency Plot for Food Access Research Atlas
(FARA)")
```

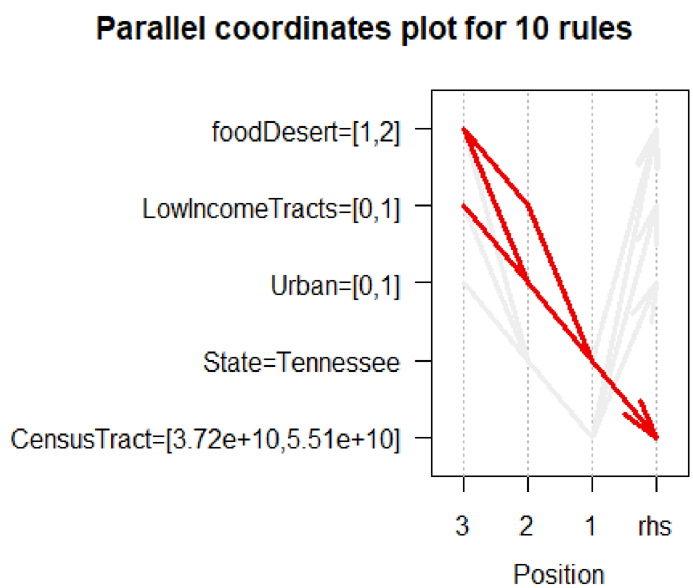


APRIORI

```
## Apriori
# Get the rules with low support and low confidence
rules <- apriori(transactions, parameter = list(supp = 0.02, conf = 0.7, minl
en=4))

# Show rules
inspect(rules[1:20])

## Sort by lift
SortedRules_conf <- sort(rules, by="confidence", decreasing=F)
inspect(SortedRules_conf[1:50])

## Take the top 10 rules sorted by lift
top10rules_conf <- head(SortedRules_conf, n = 10, by = "confidence")
inspect(top10rules_conf)

## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.7    0.1    1 none FALSE            TRUE       5    0.02      4
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 20
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[70 item(s), 1000 transaction(s)] done [0.00s].
## sorting and recoding items ... [41 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.00s].
## writing ... [7915 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

##      lhs                                      rhs
support confidence coverage lift count
## [1]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       Urban=[0,1]}                         => {LowIncomeTracts=[0,1]}
0.02          1     0.02    1    20
## [2]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       LowIncomeTracts=[0,1]}               => {Urban=[0,1]}
0.02          1     0.02    1    20
## [3]  {State=Tennessee,
##       Urban=[0,1],
```

```
##       LowIncomeTracts=[0,1]}            => {CensusTract=[3.72e+10,5.51e+10
]}    0.02          1     0.02    3    20
## [4]  {CensusTract=[3.72e+10,5.51e+10]
,
##       State=Tennessee,
##       Urban=[0,1]}                      => {foodDesert=[1,2]}
0.02          1     0.02    1    20
## [5]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       foodDesert=[1,2]}                 => {Urban=[0,1]}
0.02          1     0.02    1    20
## [6]  {State=Tennessee,
##       Urban=[0,1],
##       foodDesert=[1,2]}                 => {CensusTract=[3.72e+10,5.51e+10
]}    0.02          1     0.02    3    20
## [7]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       LowIncomeTracts=[0,1]}            => {foodDesert=[1,2]}
0.02          1     0.02    1    20
## [8]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       foodDesert=[1,2]}                 => {LowIncomeTracts=[0,1]}
0.02          1     0.02    1    20
## [9]  {State=Tennessee
,
##       LowIncomeTracts=[0,1],
##       foodDesert=[1,2]}                 => {CensusTract=[3.72e+10,5.51e+10
]}    0.02          1     0.02    3    20
## [10] {State=Tennessee,
##       Urban=[0,1],
##       LowIncomeTracts=[0,1]}            => {foodDesert=[1,2]}

##      lhs                                   rhs                         sup
port confidence coverage lift count
## [1]  {CensusTract=[1.81e+10,3.72e+10),
##       Urban=[0,1],
##       PovertyRate=medium}               => {MedianFamilyIncome=medium}   0
.070       0.70    0.100  2.1    70
## [2]  {CensusTract=[1.81e+10,3.72e+10),
##       LowIncomeTracts=[0,1],
##       PovertyRate=medium}               => {MedianFamilyIncome=medium}   0
.070       0.70    0.100  2.1    70
## [3]  {CensusTract=[1.81e+10,3.72e+10),
##       PovertyRate=medium,
##       foodDesert=[1,2]}                 => {MedianFamilyIncome=medium}   0
.070       0.70    0.100  2.1    70
## [4]  {CensusTract=[1.81e+10,3.72e+10),
##       Urban=[0,1],
##       LowIncomeTracts=[0,1],
```

```
##       PovertyRate=medium}                            => {MedianFamilyIncome=medium}    0
.070       0.70    0.100  2.1    70

##      lhs                                                rhs
support confidence coverage lift count
## [1]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       Urban=[0,1]}                                 => {LowIncomeTracts=[0,1]}
0.02          1      0.02    1    20
## [2]  {CensusTract=[3.72e+10,5.51e+10],
##       State=Tennessee,
##       LowIncomeTracts=[0,1]}                       => {Urban=[0,1]}
0.02          1      0.02    1    20
## [3]  {State=Tennessee,
##       Urban=[0,1],
##       LowIncomeTracts=[0,1]}                       =>

## Visualize the rules with a parallel coordinate plot
plot(top10rules_conf, method = "paracoord")
plot(top10rules_conf, method = "graph", interactive = T)
```



Section 2: Clustering

PRE-PROCESSING / DATA MUNGING

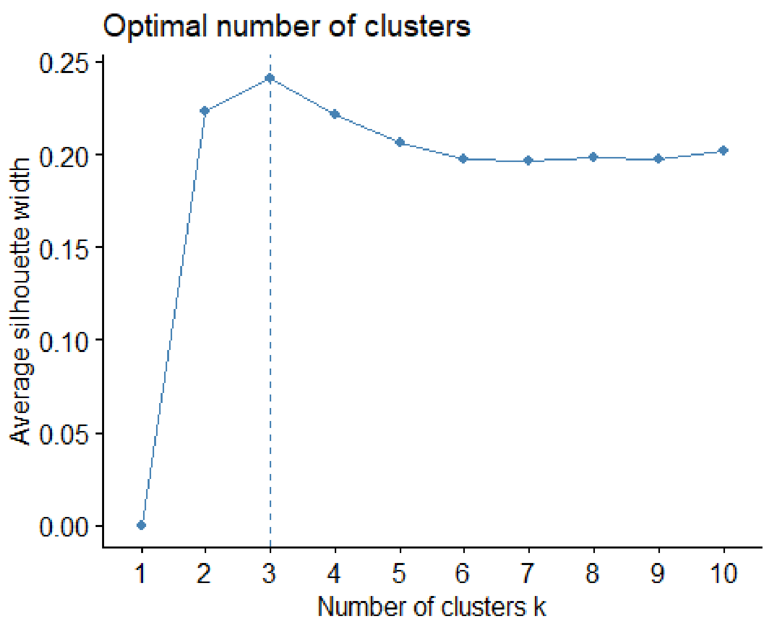
```
project_data -> raw.data.cluster

str(raw.data.cluster)

## 'data.frame':    1000 obs. of  10 variables:
##  $ CensusTract       : num  1.70e+10 4.20e+10 1.31e+10 4.51e+10 6.04e+09 .
..
```

```
## $ State            : chr "Illinois" "Pennsylvania" "Georgia" "South Car
olina" ...
## $ County           : chr "Cook County" "Allegheny County" "Forsyth Coun
ty" "York County" ...
## $ Urban            : int 1 1 1 1 1 1 0 1 1 1 ...
## $ Pop2010          : int 1188 3165 5614 8916 3592 6638 2128 1193 2632 2
300 ...
## $ OHU2010          : int 591 1485 1740 3663 1588 1986 822 606 815 1066
...
## $ LowIncomeTracts  : int 1 1 0 0 1 0 0 1 1 1 ...
## $ PovertyRate      : num 23.4 35.2 1.8 17.3 24.3 13.7 10.7 41.9 33 20.4
...
## $ MedianFamilyIncome: int 60938 26336 151944 60625 45208 56510 85703 254
41 39958 56797 ...
## $ foodDesert       : int 1 1 1 1 1 1 1 1 1 1 ...

##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 1000      10                2                  8                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0          1000              10000        93816

##
##   1   2
## 500 500

  # Convert Data to Numeric-Only For Clustering
    colnames(data.cluster[,c(1,2,3,4)])

## [1] "Group.1"     "CensusTract" "State"       "County"

    str(data.cluster[,c(-1,-2,-3,-4)])

## 'data.frame':    50 obs. of  7 variables:
## $ Urban            : num 0.667 0 0.833 0.556 0.901 ...
## $ Pop2010          : num 4060 4193 4378 3422 4608 ...
## $ OHU2010          : num 1604 1688 1573 1375 1563 ...
## $ LowIncomeTracts  : num 0.524 0 0.583 0.778 0.582 ...
## $ PovertyRate      : num 24.7 9.8 20.9 22.1 15.9 ...
## $ MedianFamilyIncome: num 50854 92000 56345 50307 79064 ...
## $ foodDesert       : num 1.48 1 1.46 1.78 1.32 ...

    num.data.cluster <- data.cluster[,c(-1,-2,-3,-4)]
    num.data.cluster <- as.data.frame(scale(num.data.cluster))
    str(num.data.cluster)

## 'data.frame':    50 obs. of  7 variables:
## $ Urban            : num -0.128 -2.877 0.559 -0.586 0.838 ...
## $ Pop2010          : num 0.214 0.408 0.678 -0.719 1.014 ...
## $ OHU2010          : num 0.422 0.768 0.293 -0.531 0.249 ...
## $ LowIncomeTracts  : num -0.3397 -2.5871 -0.0843 0.75 -0.0882 ...
## $ PovertyRate      : num 0.898 -1.585 0.268 0.478 -0.557 ...
```

```
## $ MedianFamilyIncome: num -0.728 1.793 -0.391 -0.761 1 ...
## $ foodDesert        : num -0.133 -2.289 -0.214 1.232 -0.846 ...
```

MODELS

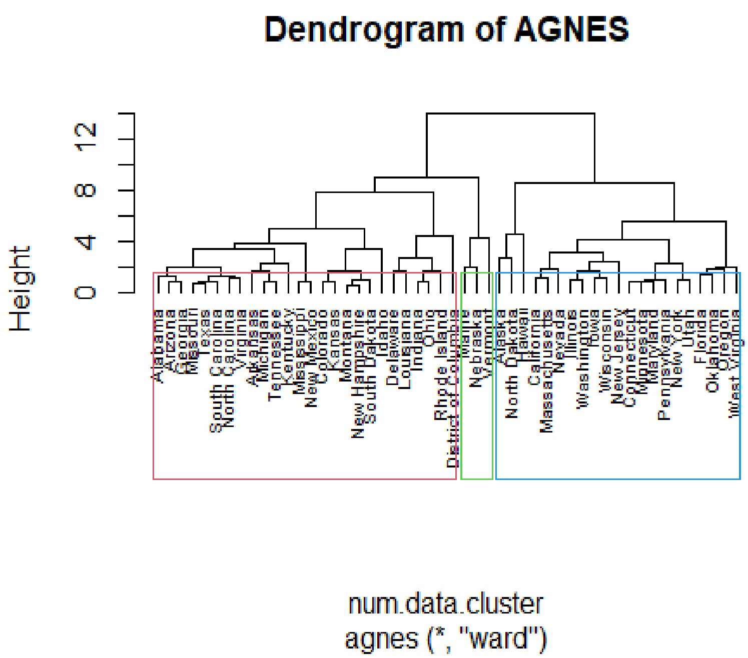
```
# Optimal Amount of Clusters | Average Silhouette Method
    fviz_nbclust(num.data.cluster, FUN = hcut, method = "silhouette")
      # plot shows 3 optimal clusters
```



Agnes Function | Dendrogram

```
    # Determine Optimal Agnes Method
    m.assess <- c("average", "single", "complete", "ward")
    names(m.assess) <- c( "average", "single", "complete", "ward")

    compute.coeff <- function(x)
      {
      agnes(num.data.cluster, method = x)$ac
      }

    hc.coeff.df <- as.data.frame(map_dbl(m.assess, compute.coeff))
    hc.coeff.df

##         map_dbl(m.assess, compute.coeff)
## average                        0.7337589
## single                         0.5997242
```

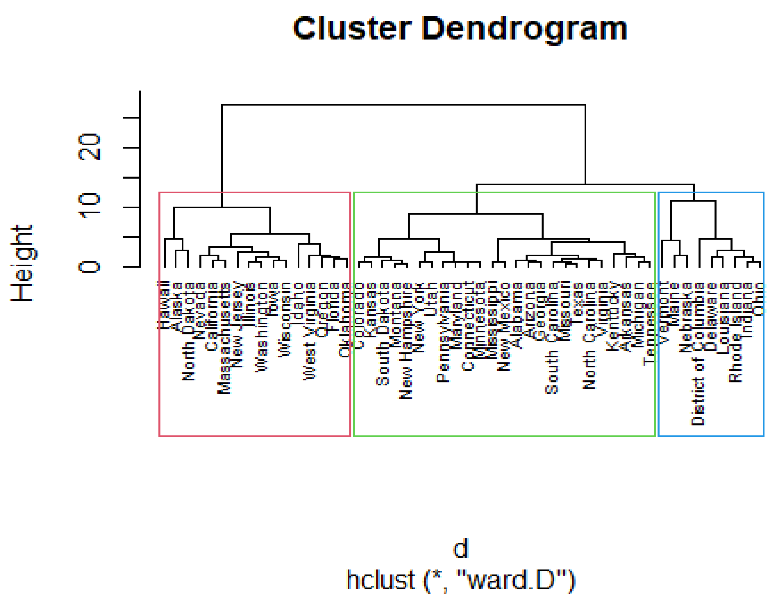
```
## complete                                   0.8469888
## ward                                       0.8895291

      # Method 'ward' conveys highest quality with 0.8895291

    H.C <- agnes(num.data.cluster, method = "ward")

    # Agglomerative coefficient (which measures the amount of clustering stru
cture found)
    # (values closer to 1 suggest strong clustering structure)
    H.C$ac

## [1] 0.8895291

    ## [1] 0.8895291

    pltree(H.C, cex = 0.6, hang = -1, main = "Dendrogram of AGNES")
    rect.hclust(H.C, k = 3, border = 2:5)
```



hclust Function / Dendrogram

```
    # Dissimilarity matrix
    d <- dist(num.data.cluster, method = "euclidean")
    #d
    # Hierarchical clustering using Complete Linkage
    hc1 <- hclust(d, method = "ward.D" )

    # Plot the obtained dendrogram
```

```r
plot(hc1, cex = 0.6, hang = -1)
rect.hclust(hc1, k = 3, border = 2:5)
```




```r
#fviz_nbclust(num.data.cluster, kmeans, method = "wss")
```

K MEANS

```r
k.means <- kmeans(num.data.cluster, 3)
k.means
```

```
## K-means clustering with 3 clusters of sizes 4, 15, 31
##
## Cluster means:
##        Urban     Pop2010     OHU2010 LowIncomeTracts PovertyRate
## 1 -0.9086002  0.8152088  0.7177478     -2.26061986  -1.7716848
## 2 -0.2280598 -0.9927497 -1.0534376      0.78546056   0.9185914
## 3  0.2275903  0.3751745  0.4171152     -0.08836867  -0.2158752
##   MedianFamilyIncome foodDesert
## 1          2.2236455 -1.2306823
## 2         -0.8477377  0.4461845
## 3          0.1232736 -0.0570980
##
## Clustering vector:
##              Alabama               Alaska              Arizona
##                    3                    1                    3
##             Arkansas           California             Colorado
##                    2                    3                    3
##          Connecticut             Delaware District of Columbia
##                    3                    2                    2
##              Florida              Georgia               Hawaii
##                    3                    3                    1
```

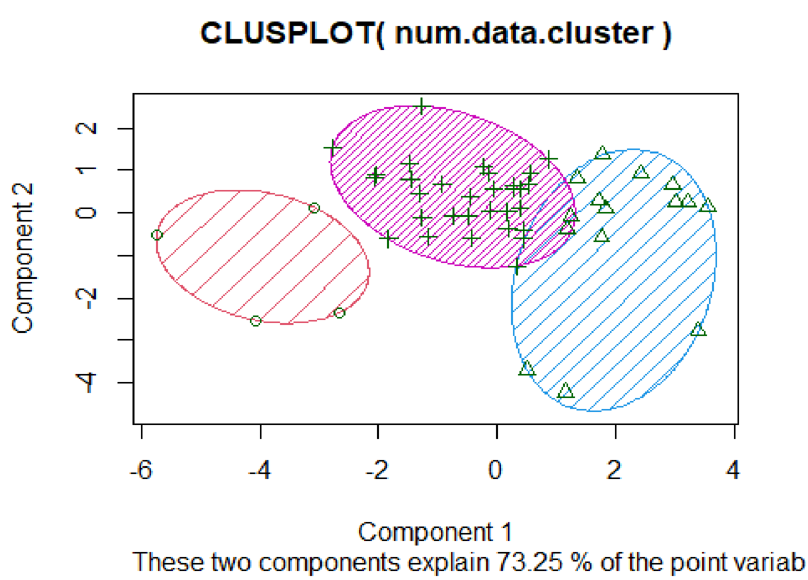
```
##              Idaho          Illinois         Indiana 
##                  3                 3               2 
##               Iowa            Kansas        Kentucky 
##                  3                 3               2 
##          Louisiana             Maine        Maryland 
##                  2                 2               3 
##      Massachusetts          Michigan       Minnesota 
##                  3                 3               3 
##        Mississippi          Missouri         Montana 
##                  2                 3               3 
##           Nebraska            Nevada   New Hampshire 
##                  2                 3               3 
##         New Jersey        New Mexico        New York 
##                  1                 2               3 
##     North Carolina      North Dakota            Ohio 
##                  2                 1               2 
##           Oklahoma            Oregon    Pennsylvania 
##                  3                 3               3 
##       Rhode Island    South Carolina    South Dakota 
##                  2                 3               3 
##          Tennessee             Texas            Utah 
##                  2                 3               3 
##            Vermont          Virginia      Washington 
##                  2                 3               3 
##      West Virginia         Wisconsin 
##                  3                 3 
## 
## Within cluster sum of squares by cluster:
## [1] 20.25762 90.21556 84.16343
##  (between_SS / total_SS =  43.3 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
k.means$centers
```

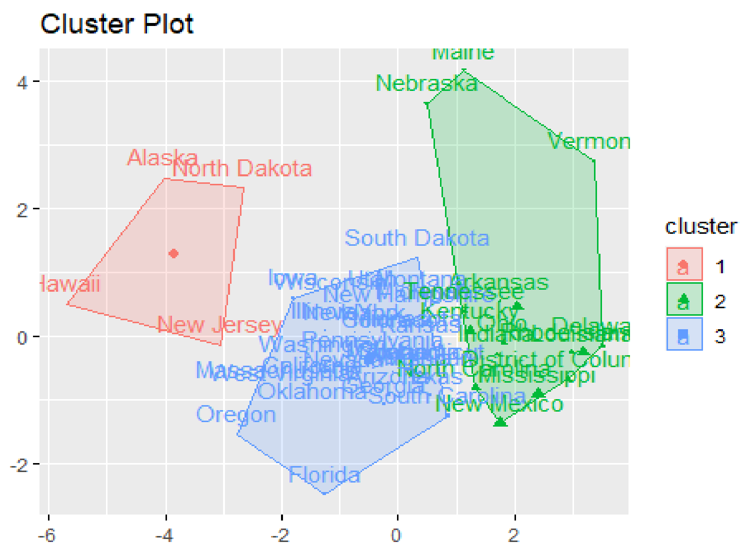
```
##        Urban    Pop2010    OHU2010 LowIncomeTracts PovertyRate
## 1 -0.9086002  0.8152088  0.7177478     -2.26061986  -1.7716848
## 2 -0.2280598 -0.9927497 -1.0534376      0.78546056   0.9185914
## 3  0.2275903  0.3751745  0.4171152     -0.08836867  -0.2158752
##   MedianFamilyIncome foodDesert
## 1          2.2236455 -1.2306823
## 2         -0.8477377  0.4461845
## 3          0.1232736 -0.0570980
```

```
assignment_clusters <- data.frame(num.data.cluster, k.means$cluster)
assignment_clusters
```

```
head(assignment_clusters)

##                 Urban    Pop2010    OHU2010 LowIncomeTracts PovertyRate
## Alabama    -0.1280735  0.2137247  0.4222219     -0.33966663   0.8983645
## Alaska     -2.8767226  0.4084355  0.7684432     -2.58707088  -1.5848068
## Arizona     0.5590888  0.6782671  0.2926119     -0.08427978   0.2676231
## Arkansas   -0.5861817 -0.7186934 -0.5309601      0.74998392   0.4784008
## California  0.8384845  1.0141702  0.2494695     -0.08820881  -0.5570090
## Colorado    0.5952552 -0.0234643  0.1716279     -0.10309776  -0.5221703
##            MedianFamilyIncome foodDesert k.means.cluster
## Alabama            -0.7276807 -0.1329249               3
## Alaska              1.7928665 -2.2886065               1
## Arizona            -0.3912764 -0.2137629               3
## Arkansas           -0.7611959  1.2323401               2
## California          1.0004205 -0.8459580               3
## Colorado            0.3378109  1.0470272               3

clusplot(num.data.cluster, k.means$cluster, color=T, shade=T,
         Labels=2, lines=0) # plot clusters
```




```
fviz_cluster(k.means, data = num.data.cluster,
             main = 'Cluster Plot',
             xlab = '',
             ylab = '', pointsize = num.data.cluster$PovertyRate)
```




```r
    # Helpful Data Tables
    # Create Separate DF
    main.cluster.df <- data.frame(raw.data.cluster, k.means$cluster)

## Warning in data.frame(raw.data.cluster, k.means$cluster): row names were f
ound
## from a short variable and have been discarded

    main.cluster.df <- main.cluster.df[,c(-1,-3,-4)]
    head(main.cluster.df)

##            State Pop2010 OHU2010 LowIncomeTracts PovertyRate MedianFamilyI
ncome
## 1       Illinois    1188     591               1        23.4
60938
## 2   Pennsylvania    3165    1485               1        35.2
26336
## 3        Georgia    5614    1740               0         1.8             1
51944
## 4 South Carolina    8916    3663               0        17.3
60625
## 5     California    3592    1588               1        24.3
45208
## 6          Texas    6638    1986               0        13.7
56510
##   foodDesert k.means.cluster
## 1          1               3
## 2          1               1
## 3          1               3
## 4          1               2
## 5          1               3
## 6          1               3
```

```
    # Discretize Poverty Rate
    pv.bins <- 3

    min.pv <- min(main.cluster.df$PovertyRate)
    min.pv

## [1] 0

    max.pv <- max(main.cluster.df$PovertyRate)
    max.pv

## [1] 82.8

    mid.pv <- (max.pv - min.pv) / pv.bins
    mid.pv

## [1] 27.6

    mid.pv * 3

## [1] 82.8

    main.cluster.df$DiscPovertyRate <- cut(main.cluster.df$PovertyRate,
                                           breaks = c(min.pv, mid.pv, max.pv,
Inf),
                                           labels = c('Min', 'Mid', 'Max'))

    str(main.cluster.df)

## 'data.frame':    1000 obs. of  9 variables:
##  $ State             : Factor w/ 50 levels "Alabama","Alaska",..: 14 39 11
41 5 44 28 39 11 10 ...
##  $ Pop2010           : int  1188 3165 5614 8916 3592 6638 2128 1193 2632 2
300 ...
##  $ OHU2010           : int  591 1485 1740 3663 1588 1986 822 606 815 1066
...
##  $ LowIncomeTracts   : int  1 1 0 0 1 0 0 1 1 1 ...
##  $ PovertyRate       : num  23.4 35.2 1.8 17.3 24.3 13.7 10.7 41.9 33 20.4
...
##  $ MedianFamilyIncome: int  60938 26336 151944 60625 45208 56510 85703 254
41 39958 56797 ...
##  $ foodDesert        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ k.means.cluster   : int  3 1 3 2 3 3 3 2 2 3 ...
##  $ DiscPovertyRate   : Factor w/ 3 levels "Min","Mid","Max": 1 2 1 1 1 1 1
2 2 1 ...

    head(main.cluster.df)

##            State Pop2010 OHU2010 LowIncomeTracts PovertyRate MedianFamilyI
ncome
## 1       Illinois    1188     591               1        23.4
60938
```

```
## 2   Pennsylvania    3165    1485               1        35.2
26336
## 3        Georgia    5614    1740               0         1.8             1
51944
## 4 South Carolina    8916    3663               0        17.3
60625
## 5     California    3592    1588               1        24.3
45208
## 6          Texas    6638    1986               0        13.7
56510
##   foodDesert k.means.cluster DiscPovertyRate
## 1          1               3             Min
## 2          1               1             Mid
## 3          1               3             Min
## 4          1               2             Min
## 5          1               3             Min
## 6          1               3             Min
```

```
cluster.1.df <- main.cluster.df[main.cluster.df$k.means.cluster == 1,]
Mode(cluster.1.df$DiscPovertyRate)
```

```
## [1] Min
## attr(,"freq")
## [1] 59
## Levels: Min Mid Max
```

```
cluster.1.df
```

```
cluster.2.df <- main.cluster.df[main.cluster.df$k.means.cluster == 2,]
Mode(cluster.2.df$DiscPovertyRate)
```

```
## [1] Min
## attr(,"freq")
## [1] 216
## Levels: Min Mid Max
```

```
mean(cluster.2.df$PovertyRate)
```

```
## [1] 20.80967
```

```
cluster.2.df
```

```
cluster.3.df <- main.cluster.df[main.cluster.df$k.means.cluster == 3,]
cluster.3.df <- na.omit(cluster.3.df)
Mode(cluster.3.df$DiscPovertyRate)
```

```
## [1] Min
## attr(,"freq")
## [1] 472
## Levels: Min Mid Max
```

```
cluster.3.df
```

```
    main.cluster.df <- aggregate(main.cluster.df, by = list(main.cluster.df$S
tate), FUN = mean)

    main.cluster.df
```

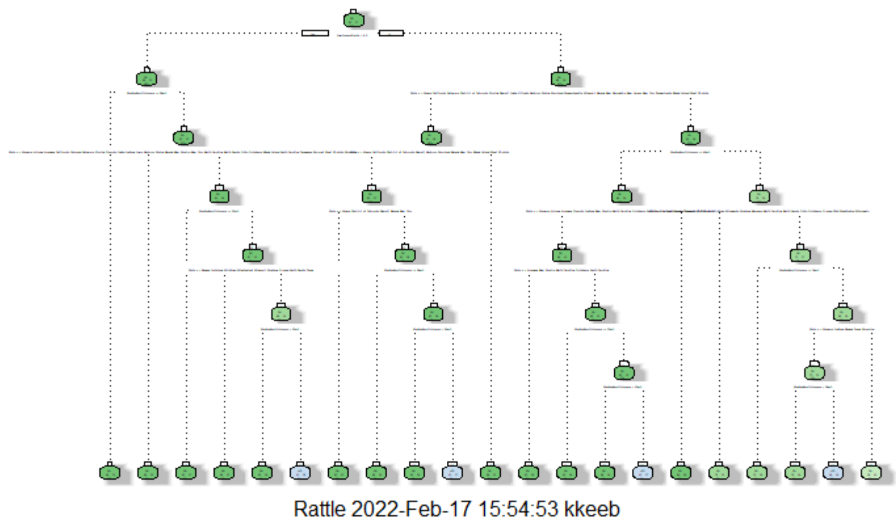
Section 3: Classification Models

DECISION TREE PREP

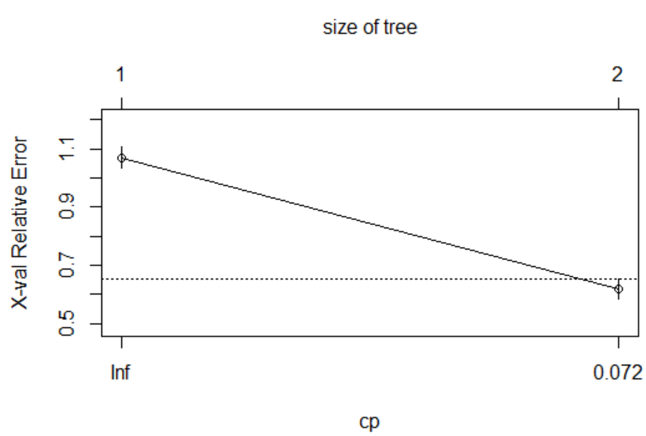
```
## Must normalize some columns and make their numbers between 0 and 1 for
##Classification Models

set.seed(341)

#randomize the dataset
fooddesert[sample(nrow(fooddesert)),]-> fooddesert


#make train 80% of data and test 20%
nrow(fooddesert)*.8-> index
fooddesert[1:index,]->train
fooddesert[(index+1): nrow(fooddesert),]->test

# check percentages
prop.table(table(train$foodDesert))

##
##      0      1
## 0.5075 0.4925

prop.table(table(test$foodDesert))

##
##    0    1
## 0.47 0.53
```

DECISION TREE MODELS

```
#decision tree 1
train_tree1 <- rpart(foodDesert ~ ., data = train,
                     method="class", control=rpart.control(cp=0, maxdepth=7))
predicted1= predict(train_tree1, test, type="class")
fancyRpartPlot(train_tree1)
table(FoodDesert=predicted1, true=data_test$foodDesert)
```

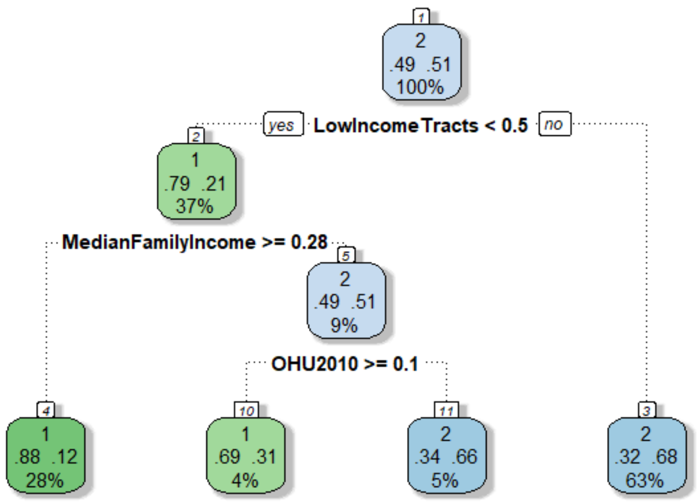



```
## Decision tree #2
train_tree1 <- rpart(foodDesert ~
Urban+OHU2010+LowIncomeTracts+PovertyRate+MedianFamilyIncome,
                     data = train, method="class",
                     control=rpart.control(cp=0.013, maxdepth=4))

#verify CP and size of tree
rsq.rpart(train_tree1)
plotcp(train_tree1)
```



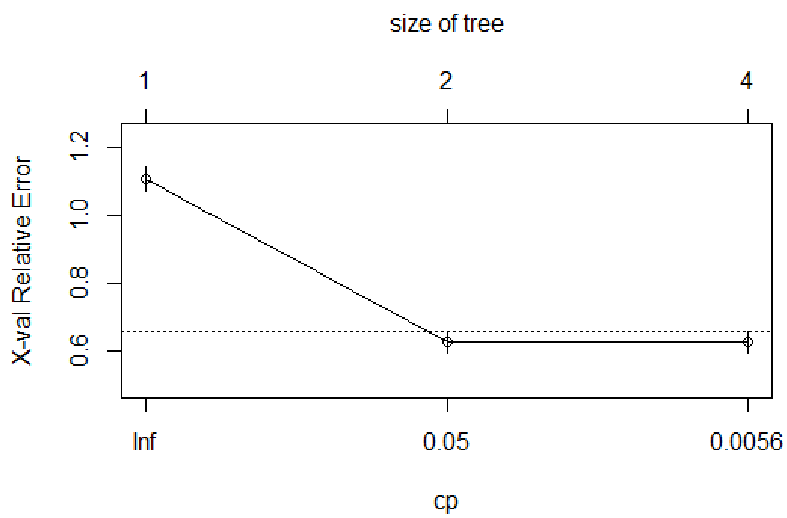

```
printcp(train_tree1)
## Classification tree:
## rpart(formula = foodDesert ~ Urban + OHU2010 + LowIncomeTracts +
##     PovertyRate + MedianFamilyIncome, data = train, method = "class",
##     control = rpart.control(cp = 0.013, maxdepth = 4))
##
## Variables actually used in tree construction:
## [1] MedianFamilyIncome
```

```
##
## Root node error: 397/800 = 0.49625
##
## n= 800
##
##        CP nsplit rel error  xerror     xstd
## 1 0.40302      0   1.00000 1.07053 0.035553
## 2 0.01300      1   0.59698 0.61965 0.032877

predicted1= predict(train_tree1, test, type="class")
fancyRpartPlot(train_tree1)
```



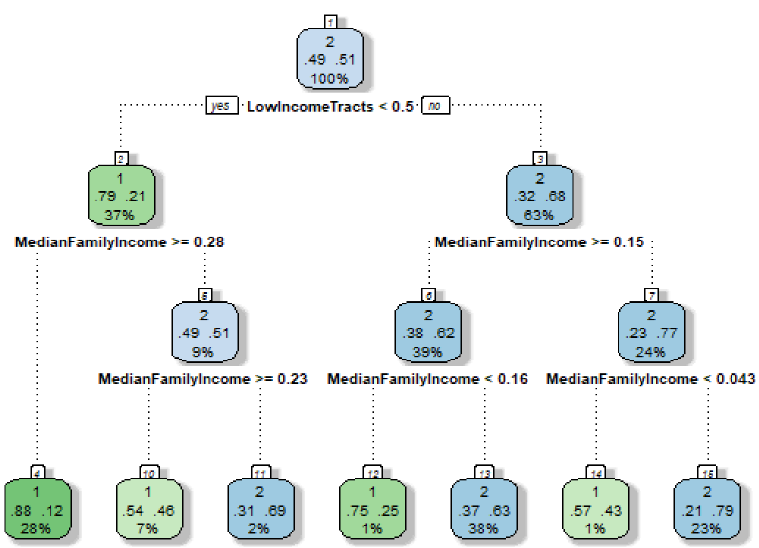

```
confusionMatrix(predicted1, as.factor(test$foodDesert))
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 63 11
##          2 40 86
##
##                Accuracy : 0.745
##                  95% CI : (0.6787, 0.8039)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : 2.232e-11
##
##                   Kappa : 0.4939
##
## Mcnemar's Test P-Value : 8.826e-05
##
##             Sensitivity : 0.6117
##             Specificity : 0.8866
##          Pos Pred Value : 0.8514
##          Neg Pred Value : 0.6825
##              Prevalence : 0.5150
```

```
##         Detection Rate : 0.3150
##   Detection Prevalence : 0.3700
##      Balanced Accuracy : 0.7491
##
##       'Positive' Class : 1

## Decision Tree #3
train_tree1 <- rpart(foodDesert ~ LowIncomeTracts+MedianFamilyIncome,
                     data = train, method="class",
                     control=rpart.control(cp=0.005, maxdepth=3))
rsq.rpart(train_tree1)
plotcp(train_tree1)
```




```
printcp(train_tree1)
## Classification tree:
## rpart(formula = foodDesert ~ LowIncomeTracts + MedianFamilyIncome,
##     data = train, method = "class", control = rpart.control(cp = 0.005,
##         maxdepth = 3))
##
## Variables actually used in tree construction:
## [1] LowIncomeTracts    MedianFamilyIncome
##
## Root node error: 397/800 = 0.49625
##
## n= 800
##
##          CP nsplit rel error xerror     xstd
## 1 0.4030227      0   1.00000 1.1058 0.035453
## 2 0.0062972      1   0.59698 0.6272 0.032987
## 3 0.0050000      3   0.58438 0.6272 0.032987

predicted1= predict(train_tree1, test, type="class")
fancyRpartPlot(train_tree1)
confusionMatrix(predicted1, as.factor(test$foodDesert))
```

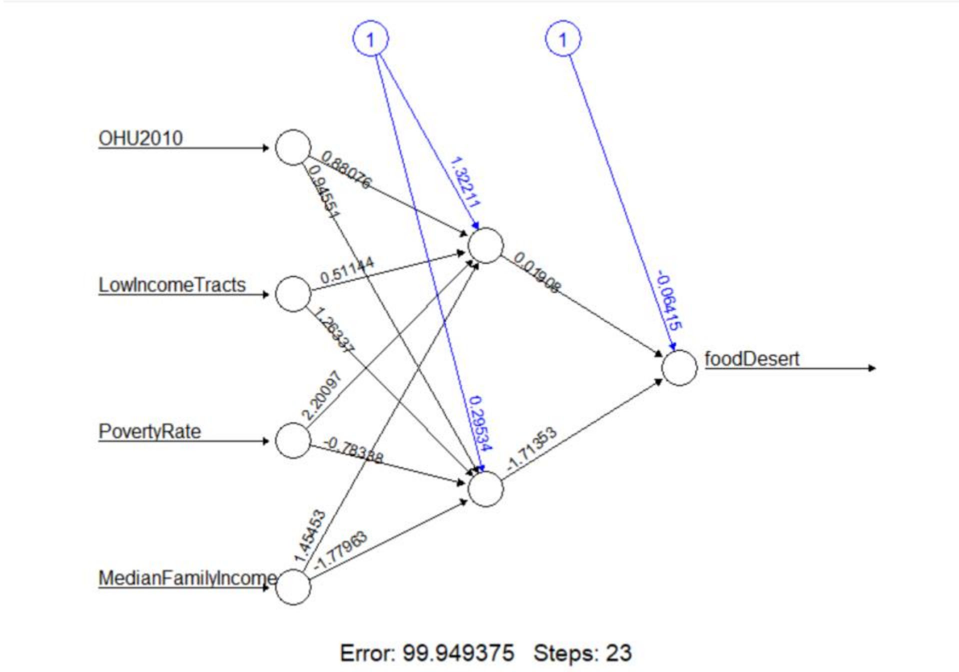
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  1  2
##          1 61  5
##          2 42 92
##
##                Accuracy : 0.765
##                  95% CI : (0.7, 0.8219)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : 3.060e-13
##
##                   Kappa : 0.5347
##
##  Mcnemar's Test P-Value : 1.512e-07
##
##             Sensitivity : 0.5922
##             Specificity : 0.9485
##          Pos Pred Value : 0.9242
##          Neg Pred Value : 0.6866
##              Prevalence : 0.5150
##          Detection Rate : 0.3050
##    Detection Prevalence : 0.3300
##       Balanced Accuracy : 0.7703
##
##        'Positive' Class : 1
printcp(train_tree1)
```



Rattle 2022-Mar-15 19:27:04 kkeeb

NEURAL NETWORKS

```
#sed.seed to get the same results each time
set.seed(24)

#create a matrix with the columns you want to include in NN
model.matrix(~OHU2010+LowIncomeTracts+PovertyRate+MedianFamilyIncome+foodDese
rt,
             data=train)->train_matrix
model.matrix(~OHU2010+LowIncomeTracts+PovertyRate+MedianFamilyIncome+foodDese
rt,
             data=test)->test_matrix

#create formulas in order to run it through the neural network
#the first is with training data.
#we take out the first and last column otherwise we get an integer column
#and another fooddesert column
col_list <- paste(c(colnames(train_matrix[,-c(1,6)])),collapse="+")
col_list <- paste(c("foodDesert~",col_list),collapse="")
f <- formula(col_list)

#create a formula for the test matrix
col_list <- paste(c(colnames(test_matrix[,-c(1,6)])),collapse="+")
col_list <- paste(c("foodDesert~",col_list),collapse="")
m <- formula(col_list)

#design and run the neural network
neuralnet(f,data=train_matrix,hidden=1,
          threshold = 0.01,
          learningrate.limit = NULL,
          learningrate.factor =
            list(minus = 0.5, plus = 1.2),
          algorithm = "rprop+") ->nn1
plot(nn1)
```




```
#plug in the test matrix to the NN and name it
output <- compute(nn1, test_matrix[,-c(1,6)],rep=1)
summary(output)
##            Length Class  Mode
## neurons      2    -none- list
## net.result 200    -none- numeric

#create a subset of just the outcomes that we need, which is
#the food desert predictions
output$net.result->pred


#The outcome is many numbers between 0 and 1, so we tell the data to
#change any number above .5 into a 1, and any number below a .5, into a 0
ifelse(output$net.result>.5, 1, 0)->pred

confusionMatrix(as.factor(pred), as.factor(test$foodDesert))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 80 35
##          1 14 71
##
##                Accuracy : 0.755
##                  95% CI : (0.6894, 0.8129)
##     No Information Rate : 0.53
```

```
##     P-Value [Acc > NIR] : 4.656e-11
##
##                   Kappa : 0.5144
##
##  Mcnemar's Test P-Value : 0.004275
##
##             Sensitivity : 0.8511
##             Specificity : 0.6698
##          Pos Pred Value : 0.6957
##          Neg Pred Value : 0.8353
##              Prevalence : 0.4700
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.5750
##       Balanced Accuracy : 0.7604
##
##        'Positive' Class : 0
##
```

```
## Must make sure that the data we are testing is the same variable type pred
#is a number and test$fooddesert is a factor, so we need to include the
#as.factor function for pred in order for confusion matrix to run correctly
#can change the hidden layers, increase the number of neurons, add more data,
#or change the learning algorithm parameters, to try and increase accuracy
```

KNN

```
set.seed(341)

## create a smaller dataset
myvars<-c("Urban","OHU2010", "LowIncomeTracts", "PovertyRate", "MedianFamilyI
ncome", "foodDesert")
train2<- train[myvars]
test2<- test[myvars]
as.factor(train2$foodDesert)->train2$foodDesert
as.factor(test2$foodDesert)->test2$foodDesert

#Used many numbers in tuneLength to help improve accuracy- 5,15,25,27,28,29
train(foodDesert~., data=train2, method="knn", tuneLength=28)->yes
predict(yes, test2)->guess
#made both the variables factors
confusionMatrix(guess, test2$foodDesert)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 83 38
##          1 11 68
##
##                Accuracy : 0.755
##                  95% CI : (0.6894, 0.8129)
```

```
##     No Information Rate : 0.53
##     P-Value [Acc > NIR] : 4.656e-11
##
##                   Kappa : 0.5161
##
##  Mcnemar's Test P-Value : 0.0002038
##
##             Sensitivity : 0.8830
##             Specificity : 0.6415
##          Pos Pred Value : 0.6860
##          Neg Pred Value : 0.8608
##              Prevalence : 0.4700
##          Detection Rate : 0.4150
##    Detection Prevalence : 0.6050
##       Balanced Accuracy : 0.7622
##
##        'Positive' Class : 0
##
```

RANDOM FOREST

```
set.seed(341)
library(randomForest)

## foodDesert must be a factor for random forest to run as a classification
as.factor(train2$foodDesert)->train2$foodDesert
as.factor(test2$foodDesert)->test2$foodDesert
rfm <- randomForest(foodDesert~., data=train2, ntree=300, importance=T)
rfm
##
## Call:
##  randomForest(formula = foodDesert ~ ., data = train2, ntree = 300,      i
mportance = T)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 27.38%
## Confusion matrix:
##     0   1 class.error
## 0 333  73   0.1798030
## 1 146 248   0.3705584

predRF <- predict(rfm, test2, type=c("class"))
confusionMatrix(predRF, test2$foodDesert)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 79 36
```

```
##          1 15 70
##
##                Accuracy : 0.745
##                  95% CI : (0.6787, 0.8039)
##     No Information Rate : 0.53
##     P-Value [Acc > NIR] : 3.353e-10
##
##                   Kappa : 0.4945
##
##  Mcnemar's Test P-Value : 0.005101
##
##             Sensitivity : 0.8404
##             Specificity : 0.6604
##          Pos Pred Value : 0.6870
##          Neg Pred Value : 0.8235
##              Prevalence : 0.4700
##          Detection Rate : 0.3950
##    Detection Prevalence : 0.5750
##       Balanced Accuracy : 0.7504
##
##        'Positive' Class : 0
##
```