University of Waterloo

Faculty of Engineering  
Department of Management Sciences

Exploring and Predicting Violent Crime in Chicago

University of Waterloo

200 University Ave W, Waterloo, ON N2L 3G1

Waterloo, Ontario, Canada

Prepared by  
  
Yingzi Zhang  
20515934  
4A Mechatronics Engineering

And

Xiang Li

20574900

4A Mechatronics Engineering

27 November 2018

1. Data Collection and Preprocessing Code

Class Variable

library**(**tidyr**)**

library**(**dplyr**)**

library**(**plyr**)**

# crime data from 2001

crime **<-** data.frame**(**crimes2001**$**X...ID,

crimes2001**$**Date,

crimes2001**$**Primary.Type,

crimes2001**$**Description,

crimes2001**$**Location.Description,

crimes2001**$**Community.Area**)**

names**(**crime**)** **<-** c**(**'id', 'date', 'type', 'description', 'location', 'community'**)**

crime **<-** crime**[!(**is.na**(**crime**$**community**)** **|** crime**$**community**==**'' **|** crime**$**community**==**'0'**)**, **]**

# dividing offense involving children into violent vs non-violent

crime**$**type **<-** ifelse**(**grepl**((**'CRIM SEX ABUSE BY FAM MEMBER'**)**, crime**$**description**)**

**|** grepl**((**'CHILD ABUSE'**)**, crime**$**description**)**

**|** grepl**((**'AGG SEX ASSLT OF CHILD FAM MBR'**)**, crime**$**description**)**

**|** grepl**((**'CHILD ABDUCTION'**)**, crime**$**description**)**

**|** grepl**((**'AGG CRIM SEX ABUSE FAM MEMBER'**)**, crime**$**description**)**

**|** grepl**((**'SEX ASSLT OF CHILD BY FAM MBR'**)**, crime**$**description**)**

**|** grepl**((**'CRIM SEX ABUSE BY FAM MEMBER'**)**, crime**$**description**)**,

gsub**(**'OFFENSE INVOLVING CHILDREN', 'VIOLENT OFFENSE INVOLVING CHILDREN', crime**$**type**)**,

gsub**(**'OFFENSE INVOLVING CHILDREN', 'NON-VIOLENT OFFENSE INVOLVING CHILDREN', crime**$**type**))**

# generating counts

commu\_crime **<-** data.frame**(**crime**$**type, crime**$**community**)**

names**(**commu\_crime**)** **<-** c**(**'type', 'community'**)**

count\_commu\_crime **<-** ddply**(**commu\_crime, .**(**commu\_crime**$**community, commu\_crime**$**type**)**, nrow**)**

names**(**count\_commu\_crime**)** **<-** c**(**"community", "type", "count"**)**

count **<-** spread**(**count\_commu\_crime, key **=** type, value **=** count**)**

count**[**is.na**(**count**)]** **<-** 0

# population

population **<-** data.frame**(**population\_chicago**$**GeogKey,

population\_chicago**$**Geog,

population\_chicago**$**`Total Population`**)**

names**(**population**)** **<-** c**(**'community', 'community name', 'population(2010)'**)**

# sum violent crimes and total crimes

names**(**sum\_crime**)** **<-** c**(**'community'**)**

sum\_crime**$**violent\_crime **<-** **NA**

sum\_crime**$**total\_crime **<-** **NA**

sum\_crime**$**violent\_crime **<-** rowSums**(**count**[**, c**(**'ASSAULT', 'BATTERY', 'CRIM SEXUAL ASSAULT',

'HOMICIDE', 'KIDNAPPING',

'VIOLENT OFFENSE INVOLVING CHILDREN',

'PUBLIC PEACE VIOLATION', 'RITUALISM',

'ROBBERY', 'SEX OFFENSE', 'WEAPONS VIOLATION'**)])**

sum\_crime**$**total\_crime **<-** rowSums**(**count**[**, **!(**colnames**(**count**)** **==** "community"**)])**

#merge sum\_crime and population

sum\_crime **<-** merge**(**x **=** sum\_crime, y **=** population, by.x **=** 'community', by.y **=** 'community', all **=** **TRUE)**

sum\_crime **<-** sum\_crime**[**, c**(**1, 4, 5, 2, 3**)]**

Average School Rating

# Creating new dataframe for school info

schooldf **<-** data.frame**(**School\_Profile\_Information**$**School\_ID,

School\_Profile\_Information**$**Short\_Name,

School\_Profile\_Information**$**Overall\_Rating,

School\_Profile\_Information**$**Rating\_Status,

School\_Locations**$**COMMAREA,

School\_Locations**$**WARD\_15**)**

community **<-** data.frame**(**census\_data\_by\_community\_area**$**communityAreaNumber,

census\_data\_by\_community\_area**$**Community**)**

names**(**schooldf**)** **<-** c**(**'id', 'name', 'rating', 'status', 'community', 'ward'**)**

names**(**community**)** **<-** c**(**'number', 'name'**)**

# Converting ratings to numbers

schooldf**[**, 'rating'**]** **=** toupper**(**schooldf**[**, 'rating'**])**

schooldf**$**rating\_num **<-** **NA**

schooldf**$**rating\_num**[**schooldf**$**rating**==**'INABILITY TO RATE'**]** **<-** **-**1

schooldf**$**rating\_num**[**is.na**(**schooldf**$**rating**)]** **<-** **-**1

schooldf**$**rating\_num**[**schooldf**$**rating**==**'LEVEL 3'**]** **<-** 1

schooldf**$**rating\_num**[**schooldf**$**rating**==**'LEVEL 2'**]** **<-** 2

schooldf**$**rating\_num**[**schooldf**$**rating**==**'LEVEL 2+'**]** **<-** 3

schooldf**$**rating\_num**[**schooldf**$**rating**==**'LEVEL 1'**]** **<-** 4

schooldf**$**rating\_num**[**schooldf**$**rating**==**'LEVEL 1+'**]** **<-** 5

# Calculating average rating by community area

avg **<-** aggregate**(**schooldf**$**rating\_num, list**(**schooldf**$**community**)**, mean**)**

names**(**avg**)** **<-** c**(**'community', 'avg\_rating'**)**

# Add community number

community**[**, 'name'**]** **=** toupper**(**community**[**, 'name'**])**

community**[**, 'name'**]** **<-** gsub**(**'[^[:alnum:][:space:]]', '', community**[**, 'name'**])**

# Join two data frames

avg\_commu **<-** merge**(**x **=** community, y **=** avg, by.x **=** 'name', by.y **=** 'community', all **=** **TRUE)**

avg\_commu\_ordered **<-** avg\_commu**[**, c**(**2, 1, 3**)]**

avg\_commu\_ordered **<-** avg\_commu\_ordered**[**with**(**avg\_commu\_ordered, order**(**number**))**, **]**

Average SSL Rating

# Create data frame for ssl and community

ssl **<-** data.frame**(**Strategic\_Subject\_List**$**`SSL SCORE`,

Strategic\_Subject\_List**$**`COMMUNITY AREA`**)**

community **<-** data.frame**(**census\_data\_by\_community\_area**$**communityAreaNumber,

census\_data\_by\_community\_area**$**Community**)**

names**(**ssl**)** **<-** c**(**'score', 'community'**)**

names**(**community**)** **<-** c**(**'number', 'name'**)**

# Eliminate rows with blank community

ssl **<-** ssl**[!(**is.na**(**ssl**$**community**)** **|** ssl**$**community**==**''**)**, **]**

# Standardize community names

ssl**[**, 'community'**]** **=** toupper**(**ssl**[**, 'community'**])**

ssl**[**, 'community'**]** **<-** gsub**(**'[^[:alnum:][:space:]]', '', ssl**[**, 'community'**])**

# Calculate the average

avg **<-** aggregate**(**ssl**$**score, list**(**ssl**$**community**)**, mean**)**

names**(**avg**)** **<-** c**(**'community', 'avg\_rating'**)**

# Add community number

community**[**, 'name'**]** **=** toupper**(**community**[**, 'name'**])**

community**[**, 'name'**]** **<-** gsub**(**'[^[:alnum:][:space:]]', '', community**[**, 'name'**])**

# Join two data frames

avg\_commu **<-** merge**(**x **=** community, y **=** avg, by.x **=** 'name', by.y **=** 'community', all **=** **TRUE)**

avg\_commu\_ordered **<-** avg\_commu**[**, c**(**2, 1, 3**)]**

avg\_commu\_ordered **<-** avg\_commu\_ordered**[**with**(**avg\_commu\_ordered, order**(**number**))**, **]**

Total Park Area

##########################################

#parks by community area

##########################################

library**(**rgdal**)**

library**(**sp**)**

library**(**dplyr**)**

library**(**sf**)**

library**(**tidyverse**)**

library**(**raster**)**

#import the shape files

chicagoparks **<-** readOGR**(**'4A/MSCI 446/R/dataToPreprocess/chicagoparksshapefile', 'geo\_export\_287c1e81-adfc-4076-bbd4-7ac4b1ca62c2'**)**

chicagocommunityareas **<-** readOGR**(**'4A/MSCI 446/R/dataToPreprocess/communityareashapefile', 'geo\_export\_f2c553e7-eb62-4773-9655-8037a1bdd109', stringsAsFactors **=** **FALSE)**

#RUN THIS CODE FOR TOTAL PARK AREA FOR EACH COMMUNITY AREA#

totalParkAreaForCommunityAreas **<-** rep**(**0, nrow**(**chicagocommunityareas**))**

**for(**i **in** 1**:**nrow**(**chicagocommunityareas**))** **{**

totalArea **<-** 0

**for** **(**j **in** 1**:**nrow**(**chicagoparks**))** **{**

#get intersection of community area & park

intersect **<-** intersect**(**chicagocommunityareas**[**i, **]**, chicagoparks**[**j, **])**

**if** **(!**is.null**(**intersect**))** **{**

#if intersection!=null, add to totalArea for current community area

totalArea **<-** totalArea **+** area**(**intersect**)**

**}**

**}**

totalParkAreaForCommunityAreas**[**i**]** **<-** totalArea

**}**

#create the dataframe consisting of three columns: communityArea, communityAreaNumber, and totalParkArea

library**(**readxl**)**

censusdata **<-** read\_excel**(**"4A/MSCI 446/R/dataToPreprocess/Census-Data-by-Chicago-Community-Area-2017 (2).xlsx"**)**

censusdata **<-** data.frame**(**censusdata**$**Community, censusdata**$**CommunityAreaNumber**)**

names**(**censusdata**)** **<-** c**(**'Community', 'communityAreaNumber'**)**

censusdata**$**Community **<-** toupper**(**censusdata**$**Community**)**

chicagocommunityareas@data**$**community**[**75**]** **<-** 'O\'HARE' #naming difference

communityAreaNumber **<-** rep**(**0, nrow**(**chicagocommunityareas**))**

#Need to match totalParkAreaForCommunityAreas to each communityAreaNumber

**for(**i **in** 1**:**nrow**(**chicagocommunityareas**))** **{**

communityAreaNumber**[**i**]** **<-** censusdata**[**which**(**censusdata**$**Community**==**chicagocommunityareas@data**$**community**[**i**])**,2**]**

**}**

totalParkAreaDF **<-** data.frame**(**chicagocommunityareas@data**$**community, communityAreaNumber, totalParkAreaForCommunityAreas**)**

names**(**totalParkAreaDF**)** **<-** c**(**'Community', 'communityAreaNumber', 'totalParkArea'**)**

#save the totalParkArea by Community Area Number

write.csv**(**totalParkAreaDF, 'totalParkAreaByCommunityArea.csv'**)**

Number of Hospitals, Teen Mom Birth Rate, Infant Mortality Rate

##########################################

#Public Safety Data

##########################################

library**(**rgdal**)**

library**(**sp**)**

library**(**dplyr**)**

library**(**sf**)**

library**(**tidyverse**)**

library**(**raster**)**

#Code for numHospitalsPerCommunityArea#################################

hospitals **<-** readOGR**(**'4A/MSCI 446/R/dataToPreprocess/Hospitals', 'Hospitals', stringsAsFactors **=** **FALSE)**

numHospitalsPerCommunityArea **<-** as.data.frame**(**table**(**hospitals@data**$**AREA\_NUMBE**))**

names**(**numHospitalsPerCommunityArea**)** **<-** c**(**'communityAreaNum', 'numHospitals'**)**

numHospitalsPerCommunityArea**$**communityAreaNum **<-** as.numeric**(**levels**(**numHospitalsPerCommunityArea**$**communityAreaNum**))**

**for** **(**i **in** 1**:**77**)** **{**

**if** **(**sum**(**numHospitalsPerCommunityArea**$**communityAreaNum **==** i**)** **==** 0**)** **{**

newDF **<-** data.frame**(**i,0**)**

names**(**newDF**)<-**c**(**'communityAreaNum', 'numHospitals'**)**

numHospitalsPerCommunityArea **<-** rbind**(**numHospitalsPerCommunityArea, newDF**)**

**}**

**}**

numHospitalsPerCommunityArea **<-** numHospitalsPerCommunityArea**[**order**(**numHospitalsPerCommunityArea**$**communityAreaNum**)**,**]**

var**(**numHospitalsPerCommunityArea**$**numHospitals**)**

#Code for teenMomRatePerCommunityArea#################################

teenMomsData **<-** read.csv**(**'4A/MSCI 446/R/dataToPreprocess/Public\_Health\_Statistics\_-\_Births\_to\_mothers\_aged\_15-19\_years\_old\_in\_Chicago\_\_by\_year\_\_1999-2009.csv'**)**

teenBirthRates **<-** data.frame**(**teenMomsData**$**Teen.Birth.Rate.1999,

teenMomsData**$**Teen.Birth.Rate..2000,

teenMomsData**$**Teen.Birth.Rate.2001,

teenMomsData**$**Teen.Birth.Rate.2002,

teenMomsData**$**Teen.Birth.Rate.2003,

teenMomsData**$**Teen.Birth.Rate.2004,

teenMomsData**$**Teen.Birth.Rate.2005,

teenMomsData**$**Teen.Birth.Rate.2006,

teenMomsData**$**Teen.Birth.Rate.2007,

teenMomsData**$**Teen.Birth.Rate.2008,

teenMomsData**$**Teen.Birth.Rate.2009**)**

teenBirthRates **<-** teenBirthRates**[**1**:**nrow**(**teenBirthRates**)-**1,**]**

teenBirthRatesTransposed **<-** t**(**teenBirthRates**)**

rownames**(**teenBirthRatesTransposed**)** **<-** **NULL**

colnames**(**teenBirthRatesTransposed**)** **=** seq**(**1**:**77**)**

#find the mean teenMomBirthRate for years 1999-2009. Use this as each community area's "teenMomBirthRate"

teenMomRatePerCommunityAreaVec**=**c**()**

**for(**i **in** 1**:**ncol**(**teenBirthRatesTransposed**)){**

teenMomRatePerCommunityAreaVec**[**i**]** **=** mean**(**teenBirthRatesTransposed**[**,i**]**, na.rm **=** **FALSE)**

**}**

teenMomRatePerCommunityArea **<-** data.frame**(**teenMomsData**[**1**:**nrow**(**teenMomsData**)-**1,1**]**, teenMomRatePerCommunityAreaVec**)**

names**(**teenMomRatePerCommunityArea**)** **<-** c**(**'communityAreaNum', 'teenMomRate'**)**

#Code for infantMortalityRatePerCommunityArea#################################

infantMortalityData **<-** read.csv**(**'4A/MSCI 446/R/dataToPreprocess/Public\_Health\_Statistics-\_Infant\_mortality\_in\_Chicago\_\_2005\_\_2009.csv'**)**

infantMortalityData **<-** infantMortalityData**[**1**:**nrow**(**infantMortalityData**)-**1,**]**

infantMortalityRatePerCommunityArea **<-** data.frame**(**infantMortalityData**$**ﮮCommunity.Area, infantMortalityData**$**Average.Infant.Mortality.Rate.2005...2009**)**

names**(**infantMortalityRatePerCommunityArea**)** **<-** c**(**'communityAreaNum', 'infantMortalityRate'**)**

remove**(**infantMortalityData**)**

#write all three to csv

publicHealthData **<-** data.frame**(**infantMortalityRatePerCommunityArea**$**communityAreaNum,

numHospitalsPerCommunityArea**$**numHospitals,

teenMomRatePerCommunityArea**$**teenMomRate,

infantMortalityRatePerCommunityArea**$**infantMortalityRate**)**

names**(**publicHealthData**)** **<-** c**(**'communityAreaNum',

'numHospitals',

'teenMomRate',

'infantMortalityRate'**)**

write.csv**(**publicHealthData, 'publicHealth.csv'**)**

Proportion of Different Races, and Percent of Children in Poverty

##########################################

#poverty & race by community area

##########################################

#did most of the conversion in excel, and using R to just create a csv of it.

library**(**readxl**)**

censusdata **<-** read\_excel**(**"4A/MSCI 446/R/dataToPreprocess/Census-Data-by-Chicago-Community-Area-2017 (2).xlsx"**)**

censusdata **<-** data.frame**(**censusdata**$**Community,

censusdata**$**CommunityAreaNumber,

censusdata**$**Hispanic,

censusdata**$**Black,

censusdata**$**White,

censusdata**$**Asian,

censusdata**$**Other,

censusdata**$**PercentChildrenInPoverty**)**

names**(**censusdata**)** **<-** c**(**'Community', 'communityAreaNumber', 'Hispanic', 'Black', 'White', 'Asian', 'Other', 'PercentChildrenInPoverty'**)**

write.csv**(**censusdata, 'censusdataByCommunityArea.csv'**)**

Combining all Datasets into One

##########################################

#Combining all the data

##########################################

avgSchoolRating **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/avg\_school\_rating\_by\_community.csv"**)**

avgSSLscore **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/avg\_ssl\_score\_by\_community.csv"**)**

censusData **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/censusdataByCommunityArea.csv"**)**

typesOfCrimes **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/crime\_count\_in\_community.csv"**)**

predictedVarDF **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/total\_crime\_by\_community.csv"**)**

totalParkArea **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/totalParkAreaByCommunityArea.csv"**)**

publicHealth **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/publicHealth.csv"**)**

#because totalParkArea dataframe is not sorted by ascending community area number:

totalParkArea **<-** totalParkArea**[**order**(**totalParkArea**$**communityAreaNumber**)**,**]**

predTable **<-** data.frame**(**totalParkArea**$**Community,

totalParkArea**$**communityAreaNumber,

predictedVarDF**$**violent\_crime **\*** 1000 **/** **(**predictedVarDF**$**population.2010.**)**,

avgSchoolRating**$**avg\_rating,

avgSSLscore**$**avg\_rating,

totalParkArea**$**totalParkArea,

publicHealth**$**numHospitals,

publicHealth**$**teenMomRate,

publicHealth**$**infantMortalityRate,

censusData**[**,4**:**ncol**(**censusData**)])**

names**(**predTable**)** **<-** c**(**

"community",

"communityAreaNum",

"percentViolentCrimePer1000Population",

"avgSchoolRating",

"avgSSLRating",

"totalParkArea",

"numHospitals",

"teenMomRate",

"infantMortalityRate",

"hispanic",

"black",

"white",

"asian",

"other",

"percentChildrenInPov"**)**

**)**

#write to csv

write.csv**(**predTable, 'predTable.csv'**)**

1. Explanatory Data Analysis Code

# gather useful columns

explanatory **<-** data.frame**(**predTable**$**communityAreaNum,

predTable**$**percentViolentCrimePer1000Population,

predTable**$**avgSchoolRating,

predTable**$**avgSSLRating,

predTable**$**totalParkArea,

predTable**$**numHospitals,

predTable**$**teenMomRate,

predTable**$**infantMortalityRate,

100**\***predTable**$**hispanic,

100**\***predTable**$**black,

100**\***predTable**$**white,

100**\***predTable**$**asian,

100**\***predTable**$**other,

100**\***predTable**$**percentChildrenInPov**)**

names**(**explanatory**)** **<-** c**(**'community', 'number\_of\_violent\_crimes\_per\_1000\_population',

'Average\_School\_Rating', 'Normalized\_Average\_SSL', 'Total\_Park\_Area\_(m2)',

'Number\_of\_Hospitals', 'Number\_of\_Teen\_Moms\_/\_1000\_Female\_Teenagers', 'Number\_of\_Infant\_Mortality\_/\_1000\_Live\_Births',

'Percent\_of\_Hispanic\_(%)', 'Percent\_of\_Black\_(%)', 'Percent\_of\_White\_(%)',

'Percent\_of\_Asian\_(%)', 'Percent\_of\_Other\_Race\_(%)', 'Percent\_of\_Children\_in\_Poverty\_(%)'**)**

# normalize SSL (266.0711 - 304.1068)

explanatory**$**Normalized\_Average\_SSL **<-** **(**explanatory**$**Normalized\_Average\_SSL**-**min**(**explanatory**$**Normalized\_Average\_SSL**))/(**max**(**explanatory**$**Normalized\_Average\_SSL**)** **-** min**(**explanatory**$**Normalized\_Average\_SSL**))**

# num\_hospital to binary

explanatory**$**Whether\_Community\_Has\_3\_or\_More\_Hospitals **<-** **NA**

explanatory**$**Whether\_Community\_Has\_3\_or\_More\_Hospitals **<-** explanatory**$**Number\_of\_Hospitals **>=** 3

col\_names **<-** colnames**(**explanatory**)**

# scatter plot

**for(**i **in** 3**:**14**)** **{**

plot**(**explanatory**[**,i**]**, explanatory**$**number\_of\_violent\_crimes\_per\_1000\_population,

main**=**paste**(**'Violent Crime Rate V.S.', gsub**(** '\\s\*\\([^\\)]+\\)', '', gsub**(**'\_', ' ', col\_names**[**i**])))**,

xlab**=**gsub**(**'\_', ' ', col\_names**[**i**])**, ylab**=**'Number of Violent Crimes / 1000 Population', pch**=**18**)**

**}**

plot**(**explanatory**[**,15**]**, explanatory**$**number\_of\_violent\_crimes\_per\_1000\_population,

main**=**paste**(**'Violent Crime Rate V.S. ', gsub**(** '\\s\*\\([^\\)]+\\)', '', gsub**(**'\_', ' ', col\_names**[**15**])))**,

xaxt**=**'n', xlim**=**c**(-**1,2**)**, xlab**=**gsub**(**'\_', ' ', col\_names**[**15**])**, ylab**=**'Number of Violent Crimes / 1000 Population', pch**=**18**)**

axis**(**1, at**=**0**:**1, labels**=**c**(**'FALSE', 'TRUE'**))**

# histograms

hist**(**explanatory**$**number\_of\_violent\_crimes\_per\_1000\_population,

col**=**'grey',

main**=**'Number of Violent Crimes / 1000 Population (Histogram)',

xlab**=**'Number of Violent Crimes / 1000 Population', ylab**=**'Number of Communities'**)**

**for(**i **in** 3**:**14**)** **{**

hist**(**explanatory**[**,i**]**,

col**=**'grey',

main**=**paste**(**gsub**(** '\\s\*\\([^\\)]+\\)', '', gsub**(**'\_', ' ', col\_names**[**i**]))**, '(Histogram)'**)**,

xlab**=**gsub**(**'\_', ' ', col\_names**[**i**])**, ylab**=**'Number of Communities'**)**

**}**

# box plots

boxplot**(**explanatory**$**number\_of\_violent\_crimes\_per\_1000\_population, data**=**explanatory,

col**=**'grey',

main**=**"Number of Violent Crimes / 1000 Population (Box Plot)",

xlab**=**"Number of Violent Crimes / 1000 Population"**)**

**for(**i **in** 3**:**14**)** **{**

boxplot**(**explanatory**[**,i**]**, data**=**explanatory,

col**=**'grey',

main**=**paste**(**gsub**(** '\\s\*\\([^\\)]+\\)', '', gsub**(**'\_', ' ', col\_names**[**i**]))**, '(Box Plot)'**)**,

xlab**=**gsub**(**'\_', ' ', col\_names**[**i**]))**

**}**

1. Numeric Regression Code

##########################################

#Numeric Regression

##########################################

library**(**caret**)**

library**(**dplyr**)**

#Import data: includes explanatory variables AND class variable but also other columns (e.g. community area name)

data **<-** read.csv**(**"4A/MSCI 446/R/explanatoryvariables/predTable.csv"**)**

#remove extraneous columns (e.g. community area name)

dataForPred **<-** dplyr**::**select**(**data, **-**X, **-**community, **-**communityAreaNum**)**

dataForPred **<-** dataForPred**[**,1**:**13**]**

dataForPred**$**numHospitals **<-** ifelse**(**dataForPred**$**numHospitals **>=** 3, 1, 0**)**

colnames**(**dataForPred**)[**5**]** **<-** "has3OrMoreHospitals"

#Use 10-fold cross-validation for getting alpha/lambda values for glmnet

tControlObj **<-** caret**::**trainControl**(**

method **=** "cv", number **=** 10,

verboseIter **=** **TRUE**,

summaryFunction **=** defaultSummary

**)**

k **<-** 10

#10-fold cross validation for performance metrics (RMSE, Rsquared, MAE)

splitPlan **<-** kWayCrossValidation**(**nrow**(**dataForPred**)**, k, **NULL**, **NULL)**

#initialization of the dataframe that will store the performance metrics

metricsDF **<-** as.data.frame**(**matrix**(**nrow **=** 3, ncol **=** 4**))**

names**(**metricsDF**)** **<-** c**(**"Model", "RMSE", "Rsquared", "MAE"**)**

OLS Linear Regression

##########################################

#train using linear regression#

modelLM **<-** train**(**

x **=** dataForPred**[**,2**:**13**]**,

y **=** dataForPred**[**,1**]**,

method **=** "lm",

trControl **=** tControlObj

**)**

#Predicted vs Actual Plot

plot**(**modelLM**$**finalModel**$**fitted.values, dataForPred**[**,1**]**, main**=**'Predicted vs Actual for Simple Linear Regression', xlab**=**'Predicted', ylab**=**'Actual'**)**

#Residual Plot

plot**(**modelLM**$**finalModel**$**fitted.values, modelLM**$**finalModel**$**residuals, main**=**'Residual Plot for Simple Linear Regression', xlab**=**'Predicted', ylab**=**'Residuals'**)**

abline**(**h **=** 0, col **=** "darkgrey", lty **=** 2**)**

#Residual Histogram Plot

hist**(**modelLM**$**finalModel**$**residuals,

col**=**'grey',

main**=**'Residual Histogram for Simple Linear Regression',

xlab**=**'Residual', ylab**=**'Frequency'**)**

#Cross Validation to get OLS Performance Metrics

lmPredValues **<-** data.frame**(**"predicted" **=** rep**(**0, nrow**(**dataForPred**)))**

**for(**i **in** 1**:**k**)** **{**

split **<-** splitPlan**[[**i**]]**

model **<-** lm**(**percentViolentCrimePer1000Population **~** ., data **=** dataForPred**[**split**$**train,**])**

lmPredValues**$**predicted**[**split**$**app**]** **<-** predict**(**model, newdata **=** dataForPred**[**split**$**app,**])**

**}**

metricsDF**[**1,**]** **<-** c**(**"Linear Regression CV", postResample**(**lmPredValues**$**predicted, dataForPred**[**,1**]))**

metricsDF**[**4,**]** **<-** c**(**"Linear Regression", postResample**(**predict**(**modelLM, dataForPred**[**2**:**13**])**, dataForPred**[**,1**]))**

Elastic Net Regression

##########################################

#train using glmnet#

modelGLMNET **<-** train**(**

x **=** dataForPred**[**,2**:**13**]**,

y **=** dataForPred**[**,1**]**,

method **=** "glmnet",

metric **=** "RMSE",

tuneGrid **=** expand.grid**(**alpha **=** 0**:**10**/**10**)**, #lambda = seq(0.0001, 1, length = 20)

trControl **=** tControlObj

**)**

#obtain the predicted values

predictionGLMNET **<-** predict**(**modelGLMNET, dataForPred**[**, 2**:**13**])**

#plots RMSE over different alpha and lambda values.

plot**(**modelGLMNET, main**=**'Alpha and Lambda Values for GLMNET'**)**

#Predicted vs Actual Plot

plot**(**predictionGLMNET, dataForPred**[**,1**]**, main**=**'Predicted vs Actual for GLMNET Regression', xlab**=**'Predicted', ylab**=**'Actual'**)**

#Residual Plot

plot**(**predictionGLMNET, **(**dataForPred**[**,1**]-**predictionGLMNET**)**, main**=**'Residual Plot for GLMNET Regression', xlab**=**'Predicted', ylab**=**'Residuals'**)**

abline**(**h **=** 0, col **=** "darkgrey", lty **=** 2**)**

#Histogram Plot

hist**((**dataForPred**[**,1**]-**predictionGLMNET**)**,

col**=**'grey',

main**=**'Residual Histogram for GLMNET Linear Regression',

xlab**=**'Residual', ylab**=**'Frequency'**)**

#Cross Validation to get performance metrics

glmnetPredValues **<-** data.frame**(**"predicted" **=** rep**(**0, nrow**(**dataForPred**)))**

**for(**i **in** 1**:**k**)** **{**

split **<-** splitPlan**[[**i**]]**

model **<-** glmnet**(**as.matrix**(**dataForPred**[**split**$**train,2**:**13**])**, dataForPred**[**split**$**train,1**]**, alpha **=** modelGLMNET**$**bestTune**$**alpha, lambda **=** modelGLMNET**$**bestTune**$**lambda**)**

glmnetPredValues**$**predicted**[**split**$**app**]** **<-** predict**(**model, s **=** modelGLMNET**$**bestTune**$**lambda, newx **=** as.matrix**(**dataForPred**[**split**$**app,2**:**13**]))**

**}**

metricsDF**[**2,**]** **<-** c**(**"Elastic Net CV", postResample**(**glmnetPredValues**$**predicted, dataForPred**[**,1**]))**

metricsDF**[**5,**]** **<-** c**(**"Elastic Net", postResample**(**predictionGLMNET, dataForPred**[**,1**]))**

GAM Regression

##########################################

#GAM model#

#since caret can only do standard GAM model of y = s(x1) + s(x2) + etc. we will not be using caret

library**(**mgcv**)**

library**(**vtreat**)**

#GAM formula, based off scatter plots of each explanatory variable vs class variable from EDA

GAMformula **<-** percentViolentCrimePer1000Population **~**

avgSchoolRating **+**

avgSSLRating **+**

s**(**totalParkArea**)** **+**

has3OrMoreHospitals **+**

s**(**teenMomRate**)** **+**

s**(**infantMortalityRate**)** **+**

s**(**hispanic**)** **+**

black **+**

s**(**white**)** **+**

s**(**asian**)** **+**

other **+**

s**(**percentChildrenInPov**)**

# Cross Validation To get Performance Metrics

gamPredValues **<-** data.frame**(**"predicted" **=** rep**(**0, nrow**(**dataForPred**)))**

**for(**i **in** 1**:**k**)** **{**

split **<-** splitPlan**[[**i**]]**

model **<-** gam**(**GAMformula, data **=** dataForPred**[**split**$**train,**]**, family **=** gaussian**)**

gamPredValues**$**predicted**[**split**$**app**]** **<-** predict**(**model, newdata **=** dataForPred**[**split**$**app,**])**

**}**

metricsDF**[**3,**]** **<-** c**(**"GAM cv", postResample**(**gamPredValues**$**predicted, dataForPred**[**,1**]))**

#Building the final model

gamModel **<-** gam**(**GAMformula, data **=** dataForPred, family **=** gaussian**)**

finalPredictions **<-** predict**(**gamModel, dataForPred**[**, 2**:**13**])**

metricsDF**[**6,**]** **<-** c**(**"GAM", postResample**(**finalPredictions, dataForPred**[**,1**]))**

#Predicted vs Actual Plot

plot**(**finalPredictions, dataForPred**[**,1**]**, main**=**'Predicted vs Actual for GAM Regression', xlab**=**'Predicted', ylab**=**'Actual'**)**

#Residual Plot

plot**(**finalPredictions, **(**dataForPred**[**,1**]-**finalPredictions**)**, main**=**'Residual Plot for GAM Regression', xlab**=**'Predicted', ylab**=**'Residuals'**)**

abline**(**h **=** 0, col **=** "darkgrey", lty **=** 2**)**

#Residual Histogram

hist**((**dataForPred**[**,1**]-**finalPredictions**)**,

col**=**'grey',

main**=**'Residual Histogram for GAM Regression',

xlab**=**'Residual', ylab**=**'Frequency'**)**

postResample**(**predict**(**gamModel, dataForPred**[**, 2**:**13**])**, dataForPred**[**,1**])**

1. Clustering Code

# Clustering

# Normalize explanatory variables

normalized **<-** explanatory

**for** **(**col **in** 1**:**ncol**(**normalized**))** **{**

normalized**[**,col**]** **<-** **(**normalized**[**,col**]-**min**(**normalized**[**,col**]))/(**max**(**normalized**[**,col**])** **-** min**(**normalized**[**,col**]))**

**}**

# Clustering Euclidean Distance

euclidean **<-** matrix**(**, nrow **=** 25, ncol **=** 2**)**

**for(**round **in** 1**:**2**)** **{**

**for(**n **in** 1**:**25**)** **{**

cl **<-** kmeans**(**normalized**[**, 3**:**14**]**, n**)**

euclidean**[**n, round**]** **<-** cl**$**tot.withinss

**}**

**}**

# plot the euclidean distance vs number of centers

plot**(**euclidean**[**,2**]**, type**=**'o', col**=**1, pch**=**18, lty**=**1,

main**=**'Total Within-Cluster Sum of Squares v.s. Number of Centers',

xlab**=**'Number of Centers', ylab**=**'Total Within-Cluster Sum of Squares'**)**

# plot trials with different sets of starting points

plot**(**euclidean**[**,1**]**, type**=**'o', col**=**4, pch**=**18, lty**=**2,

main**=**'Results Generated by Different Randomly Chosen Start Point',

xlab**=**'Number of Centers', ylab**=**'Total Within-Cluster Sum of Squares'**)**

lines**(**euclidean**[**,2**]**, type**=**"o", pch**=**18, lty**=**1, col**=**1**)**

legend**(**18, 50, c**(**"First Trial","Second Trial"**)**, cex**=**0.8,

col**=**c**(**1,4**)**, pch**=**18**:**18, lty**=**1**:**2**)**

# perform k-means again with 50 sets of starting points and take average

**for(**n **in** 1**:**25**)** **{**

cl **<-** kmeans**(**normalized**[**, 3**:**14**]**, n, nstart**=**50**)**

euclidean**[**n, round**]** **<-** cl**$**tot.withinss

**}**

plot**(**euclidean**[**,2**]**, type**=**'o', col**=**1, pch**=**18, lty**=**1,

main**=**'Sum of Squares v.s. Number of Centers',

xlab**=**'Number of Centers', ylab**=**'Total Within-Cluster Sum of Squares'**)**

# Clustering plots with 3 centres, with 50 sets of randomly chosen starting points

cl **<-** kmeans**(**normalized**[**, 3**:**14**]**, 3, nstart**=**50**)**

plot**(**normalized**[**, 3**:**14**]**, col **=** cl**$**cluster, pch**=**20**)**

points**(**cl**$**centers, col **=** cl**$**cluster, pch **=** 8, cex **=** 2**)**

1. Association Rule Mining Code

Binning

library**(**OneR**)**

# optimal number of bins:

# https://stats.stackexchange.com/questions/798/calculating-optimal-number-of-bins-in-a-histogram

col\_names **<-** colnames**(**explanatory**)**

**for(**i **in** 2**:**14**)** **{**

sorted **<-** sort**(**explanatory**[**, i**])**

hist**(**sorted, breaks**=**"FD", xlab**=**col\_names**[**i**])**

**}**

# number bins obtained from histogram

num\_bins **<-** c**(**6,6,8,12,8,6,12,5,2,5,10,10,8**)**

strs **<-** c**(**'crimes', 'school', 'SSL', 'park', 'hospital', 'teenMoms', 'infactMortality', 'hispanic', 'black', 'white', 'asian', 'other', 'childPoverty'**)**

#do fixed width binning

binnedFix **<-** explanatory

**for(**i **in** 2**:**14**)** **{**

binnedFix**[**,i**]** **<-** paste**(**strs**[**i**-**1**]**, bin**(**binnedFix**[**, i**]**, nbins **=** num\_bins**[**i**-**1**]**, labels **=** **NULL**, method **=** 'length', na.omit **=** **TRUE))**

**}**

#do adaptive width binning

binnedAdaptive **<-** explanatory

**for(**i **in** 2**:**14**)** **{**

binnedAdaptive**[**,i**]** **<-** paste**(**strs**[**i**-**1**]**, bin**(**binnedAdaptive**[**, i**]**, nbins **=** num\_bins**[**i**-**1**]**, labels **=** **NULL**, method **=** 'content', na.omit **=** **TRUE))**

**}**

#for each type of binning, do association rule mining

Apriori Algorithm (Python)

# Import necessary python libraries

import pandas as pd

import csv

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori, association\_rules

# Read in CSV file into an array of arrays

# Make sure that your data is structured like the data given in tutorial

dataset **=** **[]**

with open**(**'binnedAdaptive.csv'**)** as f**:**

reader **=** csv.reader**(**f**)**

**for** row **in** reader**:**

dataset.append**(**row**)**

# for row in dataset:

# print(row)

# Transform your data for the apriori algorithm

oht **=** TransactionEncoder**()**

oht\_ary **=** oht.fit**(**dataset**)**.transform**(**dataset**)**

df **=** pd.DataFrame**(**oht\_ary, columns**=**oht.columns\_**)**

frequent\_itemsets **=** apriori**(**df, min\_support**=**0.15, use\_colnames**=**True**)**

print**(**frequent\_itemsets**)**

frequent\_itemsets.to\_csv**(**'/Users/xiang\_li/Desktop/support015.csv'**)**

rules**=**association\_rules**(**frequent\_itemsets, metric**=**"confidence", min\_threshold**=**0.8**)**

d**=**rules**[[**'antecedents', 'consequents', 'support', 'confidence'**]]**

d.to\_csv**(**'/Users/xiang\_li/Desktop/support015\_conf08.csv'**)**