University of Waterloo

Faculty of Engineering  
Department of Management Sciences

Status Update on 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

13 November 2018

1. Summary

* R programming language is used
* Scope of project reduced so instead of random unspecified blocks of places, we just do by community areas
* No dimension of time considered anymore (makes analysis harder)
* 12 predictors: average school rating of community area, average SSL rating of community area, total park area each community area has, number of hospitals that area has, teenage mother birth rate, infant mortality rate, proportion of Hispanic, black, white, Asian, other races, and percent of children in poverty
* Preprocessing took a lot of time to do (a lot of data was geographic data and we had to use polygon algorithms and do intersections of polygons to get e.g. total park area for each community area)
* EDA: some x’s were definitely related to the class variable (e.g. race, teen mom rate, etc.)
* Prediction:
* Clustering:
* Association:

1. Data Collection Progress

Table 1 displays all of the final chosen datasets, which increased from the original project proposal due to the inclusion of additional predictors such as public health data, and because some predictors required additional datasets to be able to transform them into usable data for prediction and clustering analysis.

Table . Information on Final Chosen Datasets.

|  |  |  |
| --- | --- | --- |
| Dataset | Number of Rows | What Does a Row Represent? |
| Crimes from 2001 [1] | 6,706,459 | Reported Crime |
| Strategic Subject List [2] | 398,684 | Person Likely to be Involved in a Shooting |
| Chicago Public Schools - School Profile Information SY1718 [3] | 661 | School |
| Population and Poverty Data by Chicago Community Area [4] | 77 | Community Area |
| Parks - Chicago Park District Park Boundaries (current) [5] | N/A (Shapely File of 597 Parks) | N/A (Shapely File of 597 Parks) |
| Boundaries - Community Areas (current) [6] | N/A (Shapely File of 77 Community Areas) | N/A (Shapely File of 77 Community Areas) |
| Hospitals – Chicago [7] | N/A (Shapely File of 42 Hospitals) | N/A (Shapely File of 42 Hospitals) |
| Public Health Statistics - Births to mothers aged 15-19 years old in Chicago, by year, 1999-2009 [8] | 77 | Community Area |
| Public Health Statistics- Infant mortality in Chicago, 2005– 2009 [9] | 77 | Community Area |

The remainder of this section will describe how all the predictors and the class variable are obtained from the above datasets. Note that the code written in R to conduct these data transformations are available in the Appendix section.

Class Variable

The class variable is the percent of violent crime per 1000 people in the specified community area. This is obtained via the following formula:

*violentCrimeForCommunityArea \* 1000 / populationOfCommunityArea*

In order to obtain the total number of violent crime in a community area, the “Crimes from 2001” dataset is used. In this dataset, each crime is given a type and the community area the crime occurred in, and each crime’s type was used to filter for violent types only. Examples include: assault, battery, and homicide. Finally, the population of each community area is obtained from census data.

Average School Rating

The average school rating is the average rating of schools in a certain community area, and it is obtained by formula:

*Sum(schoolRatingForCommunityArea) / numSchoolsInCommunityArea*

The ratings for each school are from dataset, “Chicago Public Schools - School Profile Information SY1718” [3]. In this dataset, the general information about schools is given, such as names, location, ratings, and student count. The strings representing school levels in the “Overall\_Rating” column is translated to numerical scores from 1 to 5 according to the level.

Average SSL Rating

Recall that the SSL is defined as a numerical score with a range of 0 to 500, representing the likelihood of an offender to be involved in a shooting crime in the near future. 0 is extremely low risk and 500 is extremely high risk. The average SSL rating predictor is simply calculated as the average SSL rating of all strategic subject people in a certain community area. The SSL ratings is from dataset, “Strategic Subject List” [2], which takes samples from the list of arrest data from August 1, 2012 to July 31, 2016.

Total Park Area

To obtain the total park area for each community area, all park shape files [5] and all community area shape files [6] were obtained. Then, using the Raster library in R, the intersection area of each park with each community area is calculated. These intersection areas are then summated for each community area.

Number of Hospitals

To obtain the number of hospitals for each community area from the hospital data [7] was simple, since each hospital data point included the community area it is located in.

Birth Rate by Teenage Mothers

The dataset of birth rate by teenage mothers has all the rates from 1999 to 2009 [8]. The average of all these birth rates for each community areas is used.

Infant Mortality Rate

Similar to teenage mother birth rate, infant mortality rate for each community area is calculated as the average infant mortality rate from all years where data was available: from 2005 to 2009 [9]. Two values for one community area from two years had null values, so the average of the non-null values was calculated.

Proportion of Hispanic People

Nothing was needed to be transformed for this predictor [4].

Proportion of Black People

Nothing was needed to be transformed for this predictor [4].

Proportion of White People

Nothing was needed to be transformed for this predictor [4].

Proportion of Asians

Nothing was needed to be transformed for this predictor [4].

Proportion of Other Races

Nothing was needed to be transformed for this predictor [4].

Percent of Children in Poverty

Note that actual poverty rate was unable to be obtained, but percentage of children in poverty was easily obtainable and thus is used instead.

The original dataset of children poverty rates in 2018 has children separated between ages 0 to 5 and ages 6 to 12, and each of these two groups had a poverty rate percentage [4]. A weighted average based on the total population of these two groups was used to obtain the average poverty rate of all children across these two age ranges. This was done in Excel rather than in R, so no code for this data preprocessing exists in the Appendix.

1. Exploratory Data Analysis Progress

Class Variable

Average School Rating

Average SSL Rating

Total Park Area

Number of Hospitals

Birth Rate by Teenage Mothers

Infant Mortality Rate

Proportion of Hispanic People

Proportion of Black People

Proportion of White People

Proportion of Asians

Proportion of Other Races

Percent of Children in Poverty

1. Supervised and Unsupervised Learning Progress

Numeric regression and Clustering analysis were successfully conducted as of current. However, association rule mining is not completed yet.

Numeric Regression Analysis

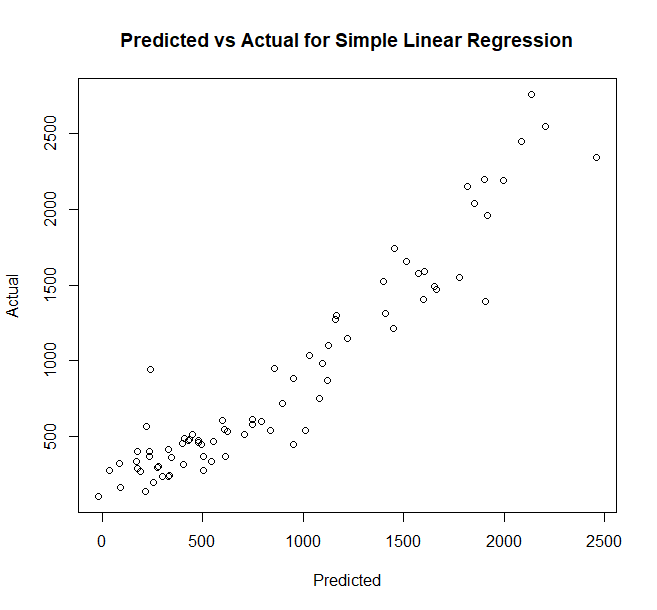
Three types of regressions were used: ordinary least squares (OLS) linear regression, elastic net regularized regression, and generalized additive linear model. Elastic nets is a mixture of lasso and ridge regression, both of which penalize the coefficients in regular OLS regression in order to prevent overfitting. Ridge regression is where the square of each coefficient for each explanatory variable is penalized, and lasso regression is where the absolute value of all the coefficients for each explanatory variable is penalized. Lasso regression can force coefficients of uncorrelated features to be zero (which can be seen as feature selection), though this may lead to information loss. Thus, elastic nets is seen as “best of both worlds”. However, as seen from the previous section of this report, many of the relationships between the explanatory variables and the class variable seem to be nonlinear in nature. Thus, the authors believed GAM might be prove to give fruitful results, since the GAM algorithm attempts to fit polynomial splines on the explanatory variables and conduct an additive linear regression on these transformed predictors.

10-fold cross validation was used to obtain the alpha and lambda parameters for the elastic nets regression. Note that alpha is mixing parameter with a range from 0 to 1. A value of 0 represents 100% ridge regression, and a value of 1 represents 100% lasso regression. Also note that lambda is the amount of penalization with increasing coefficients for the explanatory variables in the final model.

Leave-one-out cross validation was used to obtain three performance metrics for each regression type, which include: root mean squared error, (RMSE), coefficient of determination (R2), and mean absolute error (MAE).

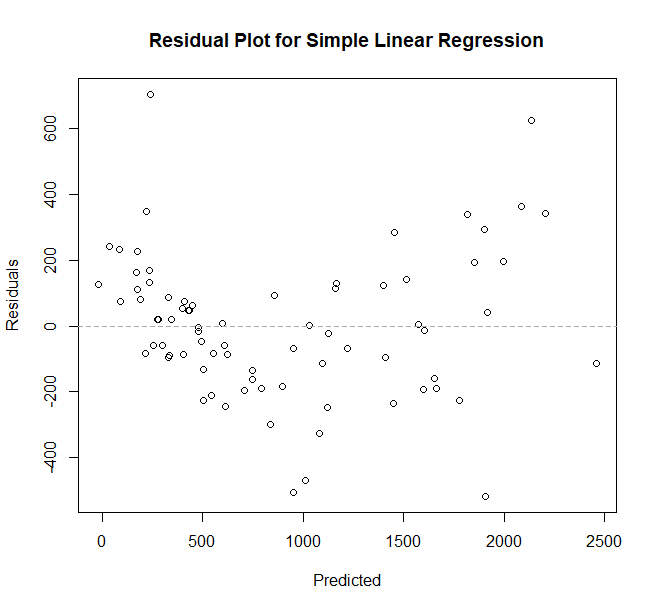
OLS Linear Regression

Figure 1 shows the predicted class variables versus the actual class variables for OLS linear regression. Ideally, the points should be uniformly distributed around the line . As shown, the predictions have this trend, which means the predictions seem to be roughly accurate. However, the general trend has somewhat of a small nonlinear concave curve.



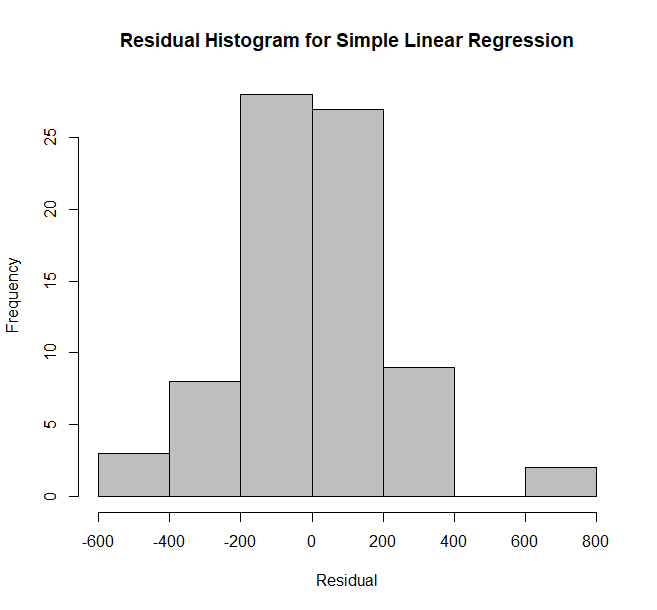
*Figure 1. Predicted Class Values vs Actual Values for OLS Linear Regression*

This concavity is more pronounced in the residual plot shown in Figure 2. Evidently, the scatter plots show that the data’s uniformity around the line is not independent of the value of x (the predicted values), which suggests linear regression may not be the best algorithm.



*Figure 2. Residual Plot for OLS Linear Regression.*

Figure 3 shows the residual histogram for OLS linear regression. While it shows a roughly uniform distribution centred around 0, it does not show the concavity pattern from the previous figure, which is one weakness of a histogram.



*Figure 3. Residual Histogram for OLS Linear Regression.*

Table 2 shows the performance metrics of this algorithm.

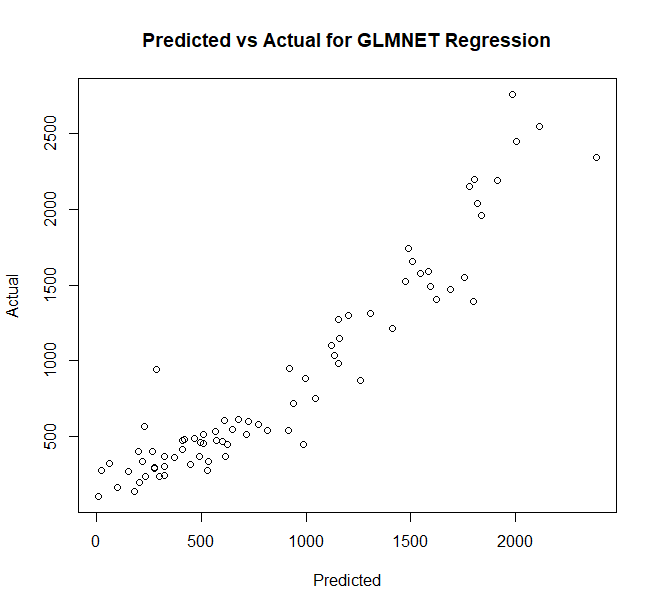
*Table 2. Performance Metrics for OLS Linear Regression*

|  |  |  |
| --- | --- | --- |
| RMSE | R2 | MAE |
| 285.243497339416 | 0.822397725231063 | 206.537526304179 |

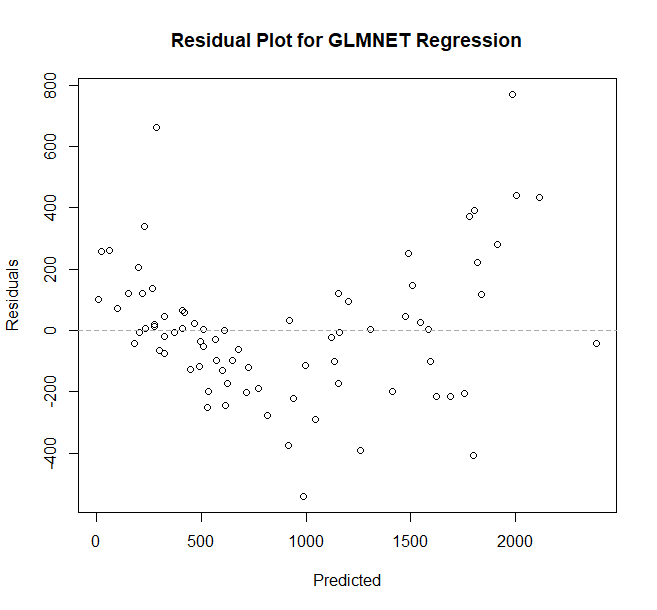
Elastic Net Regression

Using 10-fold cross validation, the best alpha and lambda values were determined to be 0 (100% ridge regression) and 1.

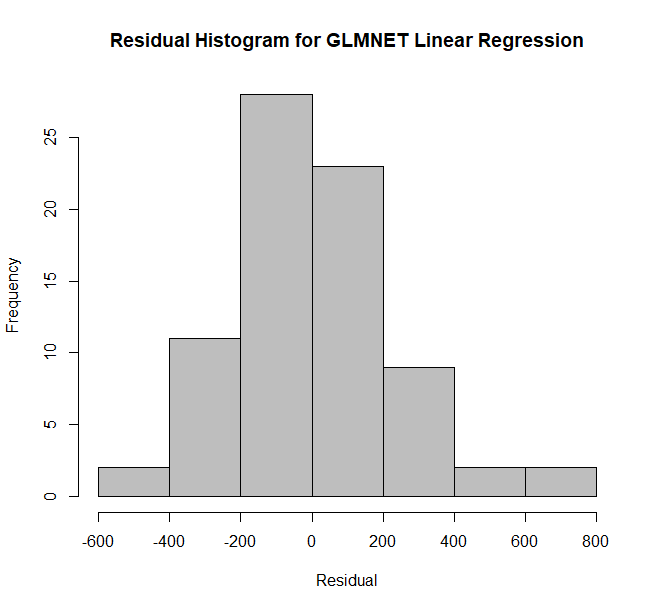
Figure 4, Figure 5, and Figure 6 show similar types of plots as ones shown for OLS regression in the last subsection. It seems that there is not a large visual difference between elastic net and OLS, though the histogram distribution slightly changed.



*Figure 4. Predicted Class Values vs Actual Values for Elastic Net Regression.*



*Figure 5. Residual Plot for Elastic Net Regression.*



*Figure 6. Residual Histogram for Elastic Net Regression.*

Ultimately, Table 3 shows that the performance metrics only improved very slightly from OLS. The changed digits are bolded for ease of reading.

*Table 3. Performance Metrics for Elastic Net Linear Regression*

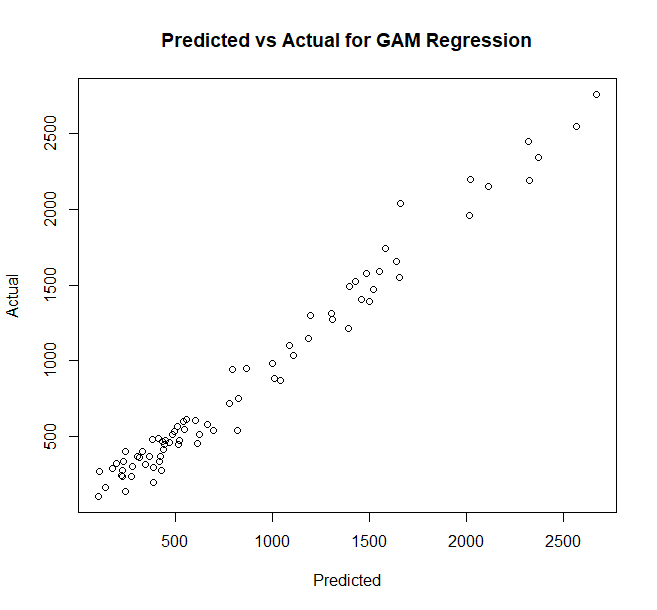
|  |  |  |
| --- | --- | --- |
| RMSE | R2 | MAE |
| 284.**112757633739** | 0.82**3577225374136** | 206.53**3355885329** |

GAM Regression

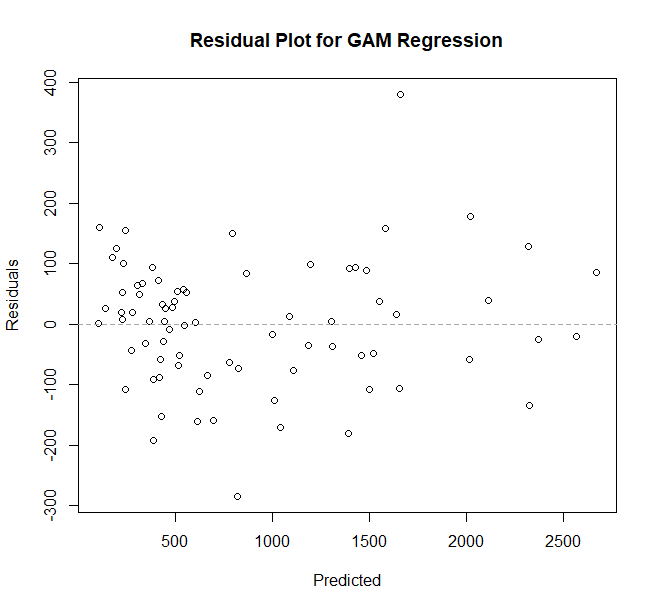
Using the scatter plots of each explanatory variable versus the class variable in the “Exploratory Data Analysis” section of this report, the relevant explanatory variables that may have nonlinear relationships with the class variable were identified. These include: total park area, birth rate for teenage mothers, infant mortality rate, percentage of Hispanics, percentage of white people, percentage of Asian people, and percentage of children in poverty. Thus, the resulting formula used is the following:

Where the function is a function that fits a spline between the input and the outcome.

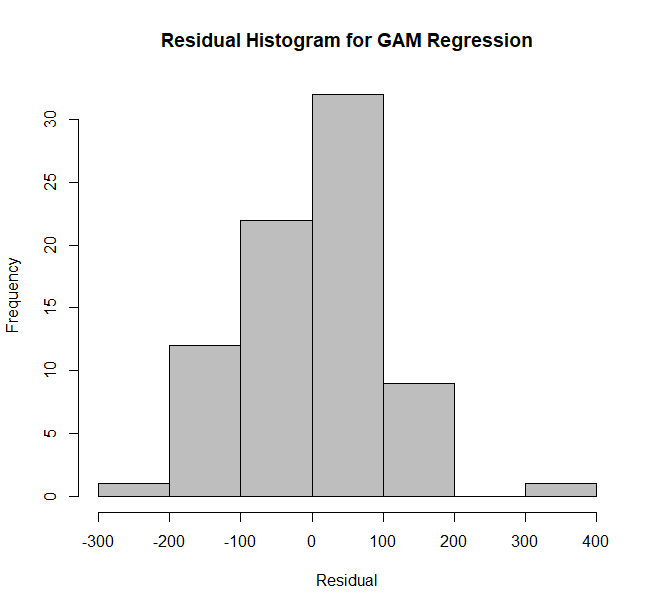
Figure 7 seems to indicate that the GAM model built on all 77 community area data points reduces the nonlinear concavity. In addition, Figure 8 and Figure 9 shows that the range of the residual decreased to . These all seem to indicate that the GAM model fits the data better.



*Figure 7. Predicted Class Values vs Actual Values for GAM Regression.*



*Figure 8. Residual Plot for GAM Regression.*



*Figure 9. Residual Histogram for GAM Regression.*

However, the performance metrics obtained from leave-one-out cross validation shown in Table 4 indicates that the GAM model performance on unseen data is actually worse than the previous two regression algorithms. With leave-one-out cross validation, 76 GAM models are each built from randomly selected 76 community area data points and tested on the single remaining community area.

*Table 4. Performance Metrics for GAM Regression.*

|  |  |  |
| --- | --- | --- |
| RMSE | R2 | MAE |
| **308.050825959162** | 0.**796551460938803** | **188.925268679306** |

Table 5 displays the performance metric of the GAM model built with all 77 community area data points. It is clear that by comparing Table 5 and Table 4, leaving even one data point out of the model significantly changes the performance metrics. Thus, the GAM algorithm heavily overfits.

*Table 5. Performance Metrics for GAM Regression (Built With All Data Points)*

|  |  |  |
| --- | --- | --- |
| RMSE | R2 | MAE |
| **103.30301646473** | 0.**976500401871001** | **79.55816922112** |

It seems that out of all three regression algorithms, elastic net regression performs the best, though it is only marginally better than OLS regression. Future recommendations include using other nonlinear regression algorithms other than GAM, as its fitted splines may use polynomials with higher than necessary orders.

Clustering Analysis

Association Rule Mining Analysis

Association rule mining is still in progress as of current. Current plans are to preprocess and create a CSV file of the explanatory variables in R. This CSV file will contain data that is interpretable with Python’s Apriori’s library. The reason this Python will be used is because the authors of this report are not familiar with association rule mining libraries in R, and because the authors believe they should gain more experience in Python.

1. Appendix

Data Collection and Preprocessing Code

Class Variable

Average School Rating

Average SSL Rating

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'**)**

Explanatory Data Analysis Code

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 **<-** 77

#leave-1-out 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'**)**

#Leave-1-out 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'**)**

#Leave-1-Out 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**)**

#Leave-1-Out 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**])**

Clustering Code