CIND 820: Big Data Analytics Project:

The Effects of Resource Accessibility on Toronto Crime Rates

Alyzeh Jiwani
501106857
25 July 2022

Abstract

Research Question:

Ontario Government spending on adult correctional facilities is sizeable and rapidly increasing yearly, hitting $1,116,561,000 in the 2020/2021 period ($806,764,000 per year with an average increase per period of $42,913,250, adjusted in 2002/2003 dollars) (Government of Canada, Statistics Canada, 2022) . What factors are contributing to high crime rates, and where should our resources be focused in order to potentially see a reduction in these values in the future?

Context:

I intend to answer these questions by investigating the relationship between crime rates and socio-economic factors in Toronto by neighbourhood. I will also look at what resources each neighbourhood has and which ones are most significant in attempting to predict crime rates. I plan on looking at two scenarios in particular; how the socio economic factors and resources of a neighbourhood effect crime rates in the same year, and how they effect crime rates three years in the future. This should provide insight into what initiatives should be implemented in communities so as to reduce the probability of major and violent crimes occurring later on. If effective, given levels of significant factors at a certain point in time, future likelihoods of crimes may be accurately predicted. Moreover a better understanding of what factors need to be targeted by government resources, specific to individual neighbourhoods should be achieved.

Data Utilized:

I aim to use a vast cohort of datasets from the City of Toronto open data repositories. There are numerous datasets available providing insight into an extensive number of potential features that may be used in my analysis. These datasets have standardized IDs for individual neighbourhoods and districts, which facilitates analysis across the different data sources. Additionally, much of the data has been consistently collected periodically throughout the past decade, thus easing the process by which we can compare the effects of the same variables over a long period of time. The data comes in a wide variety of formats, including csv, xsls, geoJSON, and unstructured text files.

The two most notable datasets I plan on using are:

https://open.toronto.ca/dataset/neighbourhood-profiles/(2001, 2006, 2011, 2016 Census data)

https://open.toronto.ca/dataset/neighbourhood-crime-rates/ (dataset consisting of counts of each type of crime by neighbourhood, as well as population for 2021, similar datasets for previous years will also be used)

While these provide the core data that I require, there are numerous other relevant data sources in the repository that I intend to utilize as well.

Techniques and Tools:

The primary themes of my project are predictive analytics, data mining and knowledge discovery. I feel that these will be the most practical in resolving my proposed question. As mentioned earlier, I will be investigating many data sources from the repository to conduct my research. This is because the City of Toronto has individual datasets per socio-economic/demographic attribute per five year period, thus there is no one data source that contains all the information I require. Thus, I will compile the relevant information from these sources into a single relational database for ease of access. From there I will be able to export my data to my software of choice, where I will conduct an exploratory data analysis to evaluate the statistical significance of the features under consideration and reduce them to those that are the most applicable. I will create charts to visualize the effects the features have on the response variable, calculate and examine their correlation coefficients and variances. I will implement other feature selection and extraction algorithms if applicable to further cut down on the dimensionality of my data. I will then create my predictive model using the earlier dated data for the test and train sets and, once content with the quality of my model, use it to predict future crime rates of each Toronto neighbourhood based on the current values of statistically significant features.

Literary Review

As discussed in the abstract, I plan on investigating the factors that are contributing to high crime rates in Toronto, and not only where our resources should be focused, but which resources should we be using in order to see a reduction in these values. We explored our approach to answering these questions in our methodology section where we briefly evaluated the use of various methods such as tests of association, and time series cross validation in order to procure solutions to our problem.

Whilst my question focusses more on what individual traits do each neighbourhood have that contribute to their crime rates, similar studies have been done but with slightly different objectives. A very notable example of this is the Toronto Strong Neighbourhoods Strategy, which is an extension of the Neighbourhood improvement Areas program, data from which we are using in our own evaluations. This study, like ours, evaluated each neighbourhood across various socioeconomic and demographic traits, and accordingly either assigned it as an NIA or not (with the potential to reevaluate later). Neighbourhoods that are assigned as NIAs become part of the neighbourhood planning strategy with the objective that upon revelation it will no longer be an NIA. This strategy has been proven effective as there are a few neighbourhoods who are no longer designated as NIAs. This program uses classification to decide whether a neighbourhood is at risk or not, and then uses focus groups and community members to help assist in

deciding what changes should be made (i.e. improvements in adult education, employment resources, child care, etc). (City of Toronto, 2020). Likewise, I primarily use classification in my analysis in order to answer my research question. However, I have used multi class classification methods as I have been classifying data points into 5 levels of crime rate severity, whereas the Toronto Strong neighbourhoods strategy classifies its data into either NIA or not an NIA. Further, The Toronto Strong initiative uses crime rate as an attribute that can be seen as contributing towards a neighbourhoods NIA status, whereas in my research crime rate is the variable of interest. The Toronto Strong initiative is a Toronto Municipal initiative and thus, unlike with my own research, has greater access to important contextual data. Examples of this would be overdose rates, archived homeless shelter data, and neighbourhood specific demographic information. These are some of many potentially useful features that I would have liked to examine in my own study, however was unable to due to access issues. In this way the Toronto Strong initiative was better able to fit models that were likely to be far more effective than my own.

Another study focuses on youth crime in Toronto, specifically police incidents involving youth. It looks at clusters of this activity, where it takes place (commercial, public, private property, indoors, outdoors etc). In particular it looked at the association between between youth crime rates and neighbourhood characteristics such as criminal opportunities, subway traffic, social control, and residential mobility. In this way this

study evaluates traits of neighbourhoods that are very different from the traits that I am evaluating in my own investigation. Here there is a focus on the geographical and environmental traits of neighbourhoods that enable crime to take place, whereas my focus is primarily on the more social factors within these neighbourhoods and how they can contribute to an increased likelihood to commit crime later. The study uses multivariate regression models to evaluate the significance of various geographical and environmental factors which are then used to predict where youth crime hotspots are located on a map. Once again, this study had access to data that I did not, and thus was able to create far more effective models and draw more robust conclusions as it was able to utilize far greater amounts of context providing data.

Methodology

Our data consists primarily of numeric variables. Our response variable, C_Rate (crime rate) and Inflation are continuous quantitative variables, whilst most of the others are discrete variables. NIA (Neighbourhood Improvement Areas) is a binary variable, taking on the value of 1 if a neighbourhood had been flagged by the city of Toronto, or 0 if it has not.

Our data was obtained as multiple different data sets. Thus in order to prepare it for further analysis it was tidied up (irrelevant columns were removed, missing values were treated, format was made uniform to facilitate consolidation into a single dataset, etc). Any missing values I encountered were found in datasets where the location of a facility or resource was unavailable. This was due to the fact that resources with a missing value for location were online resources, thus as a result I removed those resources completely from the individual datasets. This is because I'm looking at how the availability of different resources in different neighbourhoods and their individual demographics affect their crime rates. If a resource is equally accessible regardless of location it is essentially a constant and doesn't contribute to our analysis in any way.

We have classified our data into stratified groups (neighbourhoods), our goal will be to see what traits or factors these groups contain that contribute to their crime rate (a trait). Thus, we are looking to assess the association of the traits within a pre classified group, and are not necessarily looking to compare groups to each other. We will entertain two scenarios:

1. Analyzing feature values and crime rate levels measured in the same year
2. Analyzing feature values taking three years prior to their corresponding neighbourhood crime rate level.

We will compare the various models we fit onto both situations to see which one is most useful in classifying a neighbourhoods crime rate level based on sociodemographic information and resources available.

The algorithms and models we will use in our analysis are:

1. Learning Vector Quantization Model
2. Decision Trees
3. Permutation Feature Importance
4. Multinomial Logistic Regression

As our dataset is small it is vital that our model has low complexity to prevent overfitting. Thus we will aim to reduce dimensionality as much as possible and ensure that our models use few parameters. We will also use ensemble methods where we use multiple classifiers to help fit our data to help counter this issue as well. Further, we will try to counter imbalanced class levels in our dataset by using the synthetic minority oversampling technique (SMOTE) to increase the size of our dataset and reduce the number of minority class levels. We will run our models on both our original data and our data after implementing SMOTE and compare model efficacy.

Exploratory Data Analysis, Feature Selection, and Initial Modelling

After compiling the numerous datasets into one cohesive data frame, we began conducting our exploratory data analysis and visualizing our data. We explored the association between our response variable and our features. Further, we experimented with our data by exploring the different degrees of association between the attributes and crime rate when our attribute data pints were taken in different years from our response data points.

Our analysis revealed many issues with our data. Firstly, there were many inconsistencies in the public open source data, which resulted in the inability to use significant amounts of important demographic information in the analysis. This was due to demographic information being recorded every five years per ward, whereas the majority of the data we were using was collected every year by neighbourhood. More so, variables used by city officials to collect socio-economic data would change every 10-15 years, resulting in us being unable to compare older data to newer data and thus reducing the time frame of focus in our study. This caused many issues as, as our dataset was very small, we were unable to make normality assumptions on our data, thus restricting the methods by which we could conduct our analysis. This issue was especially encountered when initially

attempting to fit our data using a multinomial linear regression model, and later proven to be true using the Kolmogorv-Smirnov test.

It was concluded that we would attempt to overcome issues with normality and dataset size by using other types of machine learning models. We would facilitate this by transforming our response variable from a continuous variable to a factor variable, thus enabling us to use classification methods that do not necessarily require data to be normal.

We would fit and evaluate model performance on two situations; one in which feature values and crime rate values occur in the same year and one in which there is a tree year gap between the two. Initially we would have liked to use a five year gap, but as a result of data accessibility issues a shorter gap period was more appropriate.

Filter and wrapping feature selection methods were used to reduce dimensionality of the data. We constructed a learning vector quantization model to assist in estimating variable importance for both of our scenarios.

- For our model with all datapoint coming from the same year, our most the model estimated our most important features to be Year, Inflation, Population, and Community Housing

- For our model with the three year gap, the most important features were estimated to be Inflation, Year, Community Housing and Employment Resources.
    -

CIND 820 - EDA Data Visualisation

Alyzeh Jiwani

07/06/2022

First we read in our dataframe

```
crime_data <- read.csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/pandas_df.csv', header = TRU
```

We look at the head of our data and summary stats

```
head(crime_data)
```

```
##   X Year N_ID      Pop    C_Rate Ad_Ed Child_Care Com_House Emp_Res Sub_Trt
## 1 0 2014   97 11197.33  69.46098     0          0         0       0       0
## 2 1 2014   27 25528.89 314.67620     0          0         4       1       0
## 3 2 2014   38 14298.67 102.57366     0          0         0       1       0
## 4 3 2014   31 13508.44 311.73917     0          0        48       0       0
## 5 4 2014   16 22787.56 156.51816     0         46         0       0       0
## 6 5 2014  118 25097.78 119.97521     0          0        15       0       0
##   Trans_House Recreation Inflation NIA
## 1           0          0      1.91   0
## 2           0          0      1.91   1
## 3           0          0      1.91   0
## 4           0          0      1.91   0
## 5           0          0      1.91   0
## 6           0          0      1.91   0
```

```
summary(crime_data)
```

```
##        X               Year          N_ID             Pop
##  Min.   :   0.0   Min.   :2014   Min.   :  1.00   Min.   :  584.4
##  1st Qu.: 279.8   1st Qu.:2016   1st Qu.: 35.75   1st Qu.:10894.0
##  Median : 559.5   Median :2018   Median : 70.50   Median :16091.0
##  Mean   : 559.5   Mean   :2018   Mean   : 70.50   Mean   :17963.8
##  3rd Qu.: 839.2   3rd Qu.:2019   3rd Qu.:105.25   3rd Qu.:23610.8
##  Max.   :1119.0   Max.   :2021   Max.   :140.00   Max.   :87808.0
##      C_Rate             Ad_Ed         Child_Care       Com_House
##  Min.   :   46.27   Min.   :0.000   Min.   : 0.000   Min.   :  0.00
##  1st Qu.:  131.51   1st Qu.:0.000   1st Qu.: 0.000   1st Qu.:  1.00
##  Median :  175.53   Median :0.000   Median : 0.000   Median :  5.00
##  Mean   :  489.26   Mean   :0.492   Mean   : 7.471   Mean   : 14.75
##  3rd Qu.:  252.31   3rd Qu.:1.000   3rd Qu.: 0.000   3rd Qu.: 16.00
##  Max.   :10315.88   Max.   :9.000   Max.   :62.000   Max.   :228.00
##     Emp_Res          Sub_Trt        Trans_House        Recreation
```

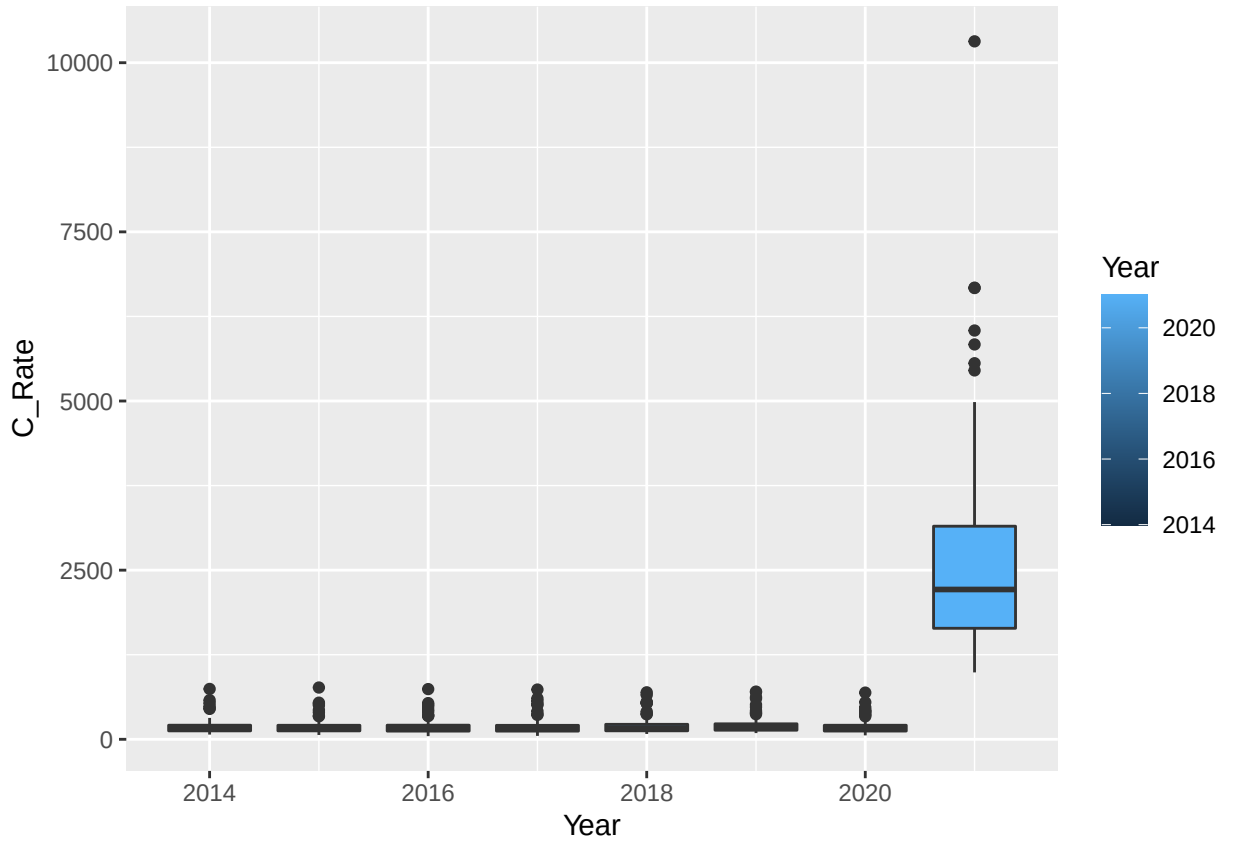
```
##  Min.   : 0.000   Min.   :0.0000   Min.   :0.00000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 0.0000
##  Median : 1.000   Median :0.0000   Median :0.00000   Median : 0.0000
##  Mean   : 1.178   Mean   :0.3214   Mean   :0.08214   Mean   : 0.9179
##  3rd Qu.: 2.000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.: 1.0000
##  Max.   :11.000   Max.   :8.0000   Max.   :1.00000   Max.   :11.0000
##    Inflation          NIA
##  Min.   :0.720   Min.   :0.0000
##  1st Qu.:1.355   1st Qu.:0.0000
##  Median :1.755   Median :0.0000
##  Mean   :1.801   Mean   :0.2214
##  3rd Qu.:2.030   3rd Qu.:0.0000
##  Max.   :3.400   Max.   :1.0000
```

We already checked and treated for missing values so we will move on with data visualisation.

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

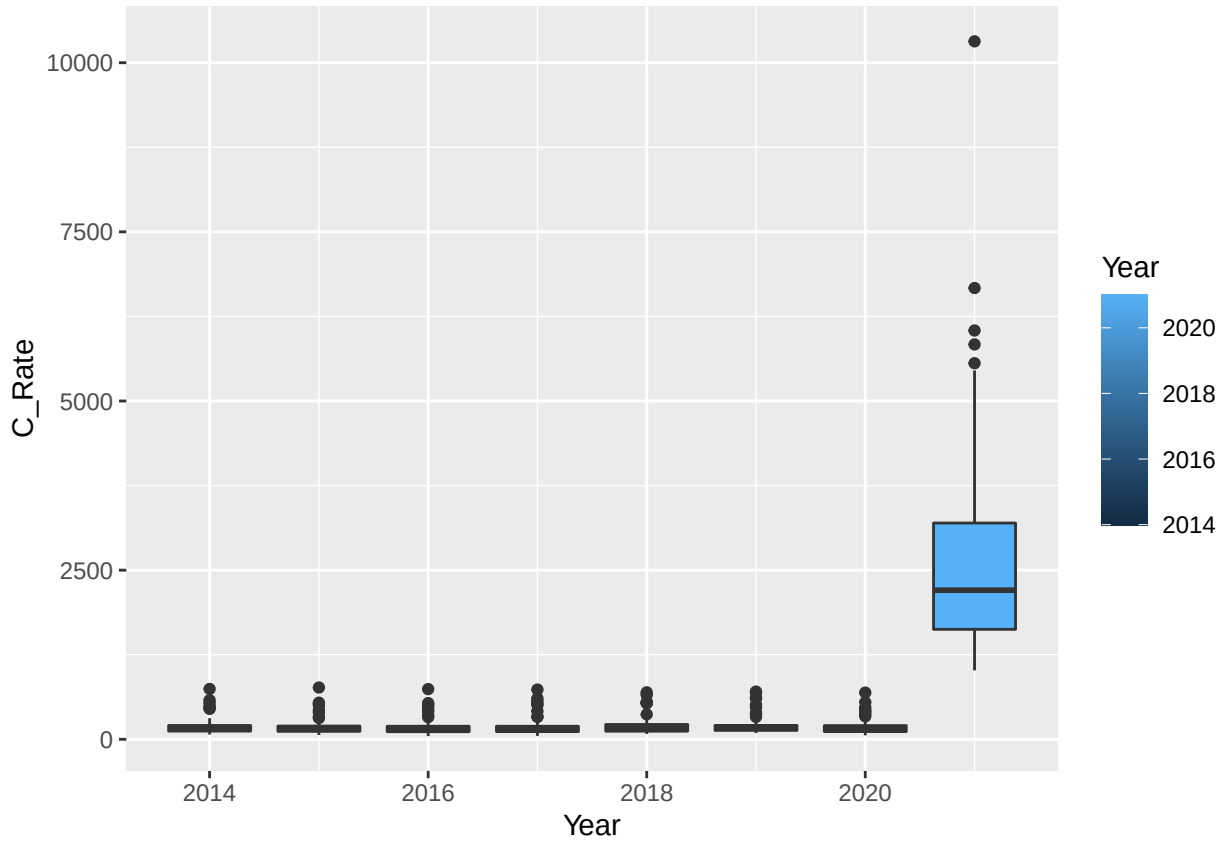
```
bp1 <- ggplot(crime_data, aes(x=Year, y = C_Rate, group = Year))+
  geom_boxplot(aes(fill = Year))
bp1
```



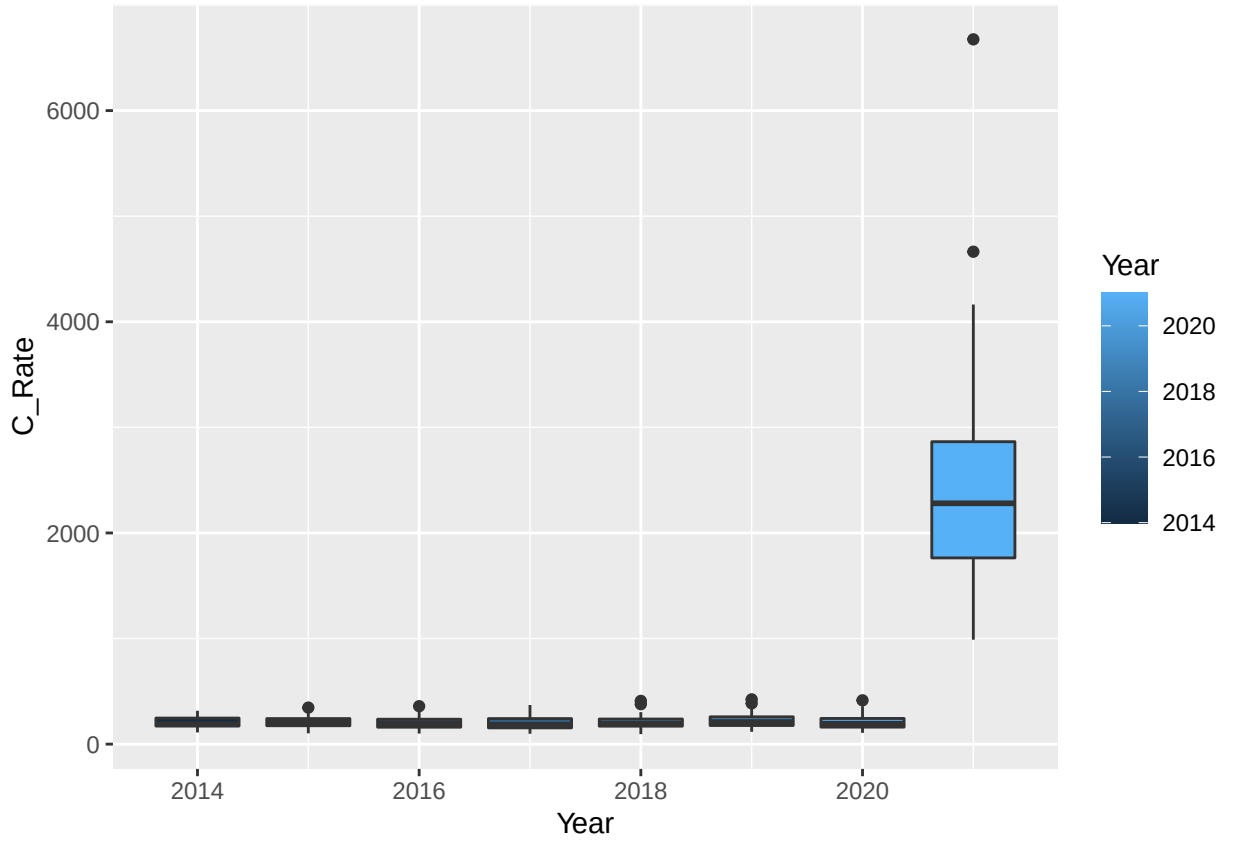
So looking at this we can see that the total crime rate inreased significantly in 2021.

Looking at how crime rate is affected more closely Instead of faceting by year and NID, we will facet by Year and NIA, this is because there are too many different neighbourhood IDs, and the NIA measure is an effective way for us to see whether being an at risk neighbourhood affects crime rate.

```
bp2 <- ggplot(crime_data[crime_data$NIA == 0,], aes(x= Year, y=C_Rate, group = Year))+
  geom_boxplot(aes(fill=Year))
bp2
```

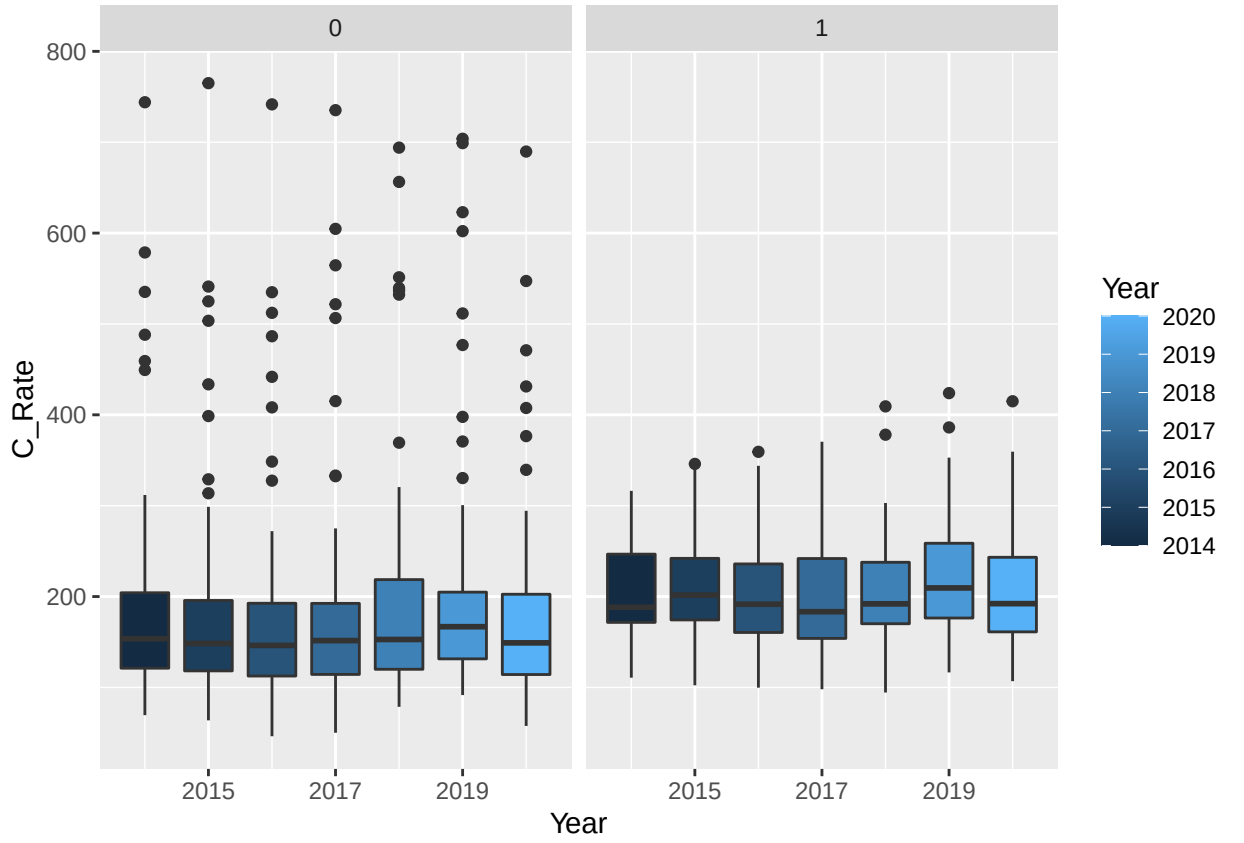



```
bp3 <- ggplot(crime_data[crime_data$NIA == 1,], aes(x= Year, y=C_Rate, group = Year))+
  geom_boxplot(aes(fill=Year))
bp3
```



The data from the year 2021 seems to be somewhat of an outlier, a likely outcome of covid and high inflation rates. Lets try to see what the spread of the data looks like without data from 2021.

```
bp4 <- ggplot(crime_data[crime_data$Year != 2021,], aes(x= Year, y=C_Rate, group = Year))+
  geom_boxplot(aes(fill=Year))+
  facet_grid(~NIA)
bp4
```



Intrestingly enough, the general spread and average rate of crime appears to be somewhat the same regardless of NIA assignment, however there are many outliers past the upper bounds in neighbourhoods that are not NIAs.This could be atributed to the fat that neighbourhoods that are considered NIAs may have higher police presence, or even the general similarity in crime rates may be attributed to spill over from other neighbourhoods.

Lets look at associations between different variables.

```
df <- crime_data[-1]
head(df)
```

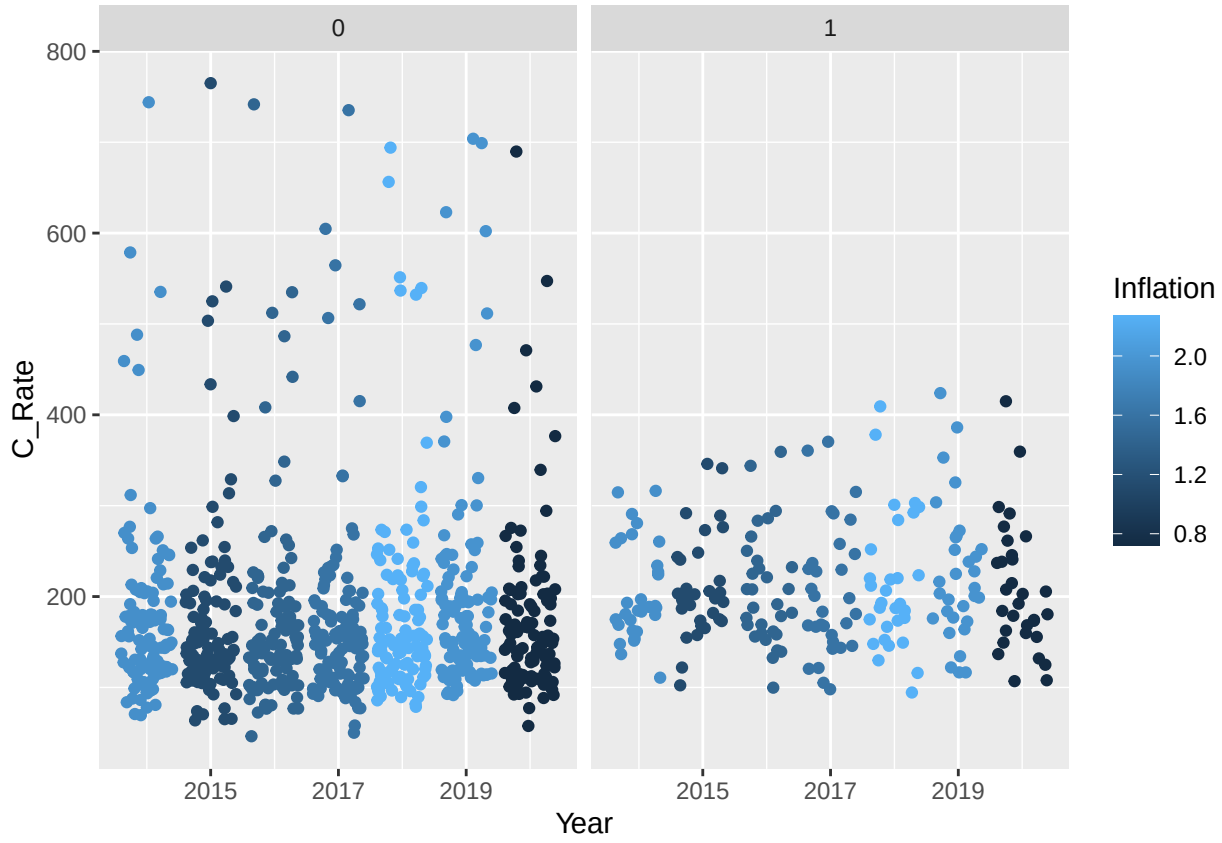
```
##   Year N_ID      Pop    C_Rate Ad_Ed Child_Care Com_House Emp_Res Sub_Trt
## 1 2014   97 11197.33  69.46098     0          0         0       0       0
## 2 2014   27 25528.89 314.67620     0          0         4       1       0
## 3 2014   38 14298.67 102.57366     0          0         0       1       0
## 4 2014   31 13508.44 311.73917     0          0        48       0       0
## 5 2014   16 22787.56 156.51816     0         46         0       0       0
## 6 2014  118 25097.78 119.97521     0          0        15       0       0
##   Trans_House Recreation Inflation NIA
## 1           0          0      1.91   0
## 2           0          0      1.91   1
## 3           0          0      1.91   0
## 4           0          0      1.91   0
## 5           0          0      1.91   0
## 6           0          0      1.91   0
```

```
cor(df)
```

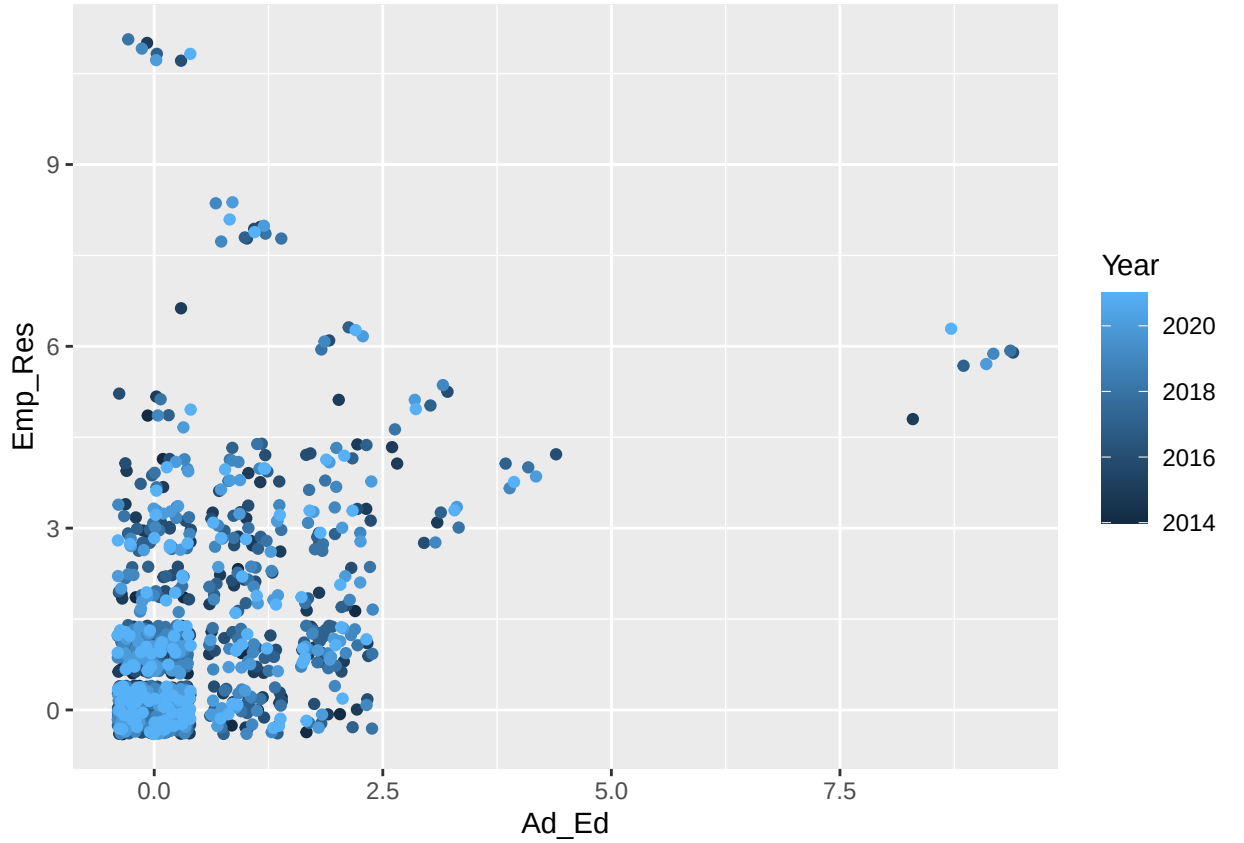
```
##                    Year        N_ID         Pop       C_Rate        Ad_Ed
## Year         1.000000000  0.00000000 -0.23255123  0.495081707  0.095126689
## N_ID         0.000000000  1.00000000  0.08695004  0.015884718  0.015812461
## Pop         -0.232551234  0.08695004  1.00000000 -0.442537501  0.155272494
## C_Rate       0.495081707  0.01588472 -0.44253750  1.000000000  0.101652592
## Ad_Ed        0.095126689  0.01581246  0.15527249  0.101652592  1.000000000
## Child_Care   0.000000000 -0.64582623 -0.04833429 -0.012728792 -0.062303869
## Com_House    0.001273552  0.02176567  0.21161234  0.062736180  0.207126774
## Emp_Res      0.121402768  0.04824455  0.39650712  0.172314897  0.406680553
## Sub_Trt      0.063679113  0.03589922  0.17160280  0.100091876  0.283627830
## Trans_House  0.055347051  0.06565584  0.17389845  0.069292831  0.202789855
## Recreation   0.129941423  0.14595138  0.19385851  0.134141255  0.457228711
## Inflation    0.381346034  0.00000000 -0.43658677  0.682234239 -0.003751164
## NIA          0.000000000  0.03001019 -0.00857220  0.007868282  0.092560904
##                Child_Care     Com_House      Emp_Res      Sub_Trt  Trans_House
## Year         0.000000e+00  0.0012735523  0.121402768  0.063679113  0.055347051
## N_ID        -6.458262e-01  0.0217656661  0.048244548  0.035899225  0.065655839
## Pop         -4.833429e-02  0.2116123402  0.396507117  0.171602801  0.173898450
## C_Rate      -1.272879e-02  0.0627361803  0.172314897  0.100091876  0.069292831
## Ad_Ed       -6.230387e-02  0.2071267739  0.406680553  0.283627830  0.202789855
## Child_Care   1.000000e+00 -0.0101195801 -0.086189838 -0.089196831 -0.026693598
## Com_House   -1.011958e-02  1.0000000000  0.269652185  0.018978699  0.100359609
## Emp_Res     -8.618984e-02  0.2696521853  1.000000000  0.367934276  0.240782833
## Sub_Trt     -8.919683e-02  0.0189786994  0.367934276  1.000000000  0.222004613
## Trans_House -2.669360e-02  0.1003596089  0.240782833  0.222004613  1.000000000
## Recreation  -1.406271e-01  0.3068642420  0.456822318  0.287532733  0.118692358
## Inflation   -6.879834e-21  0.0003821679 -0.002330719  0.002176484  0.004434589
## NIA          8.945859e-02  0.1228772991  0.126419337 -0.041360590  0.004922640
##                Recreation     Inflation           NIA
## Year         0.1299414228  3.813460e-01  0.000000e+00
## N_ID         0.1459513843  0.000000e+00  3.001019e-02
## Pop          0.1938585117 -4.365868e-01 -8.572200e-03
## C_Rate       0.1341412547  6.822342e-01  7.868282e-03
## Ad_Ed        0.4572287109 -3.751164e-03  9.256090e-02
## Child_Care  -0.1406270545 -6.879834e-21  8.945859e-02
## Com_House    0.3068642420  3.821679e-04  1.228773e-01
## Emp_Res      0.4568223181 -2.330719e-03  1.264193e-01
## Sub_Trt      0.2875327328  2.176484e-03 -4.136059e-02
## Trans_House  0.1186923575  4.434589e-03  4.922640e-03
## Recreation   1.0000000000 -3.373377e-04  1.969603e-01
## Inflation   -0.0003373377  1.000000e+00  4.761693e-21
## NIA          0.1969602904  4.761693e-21  1.000000e+00
```

None of our have a high correlation (correlation coeeficient >0.7) with each other, and thus we do not remove any as of yet

```
ggplot(df[df$Year != 2021,], aes(x = Year, y= C_Rate, colour = Inflation))+
  geom_jitter()+
  facet_grid(~NIA)
```



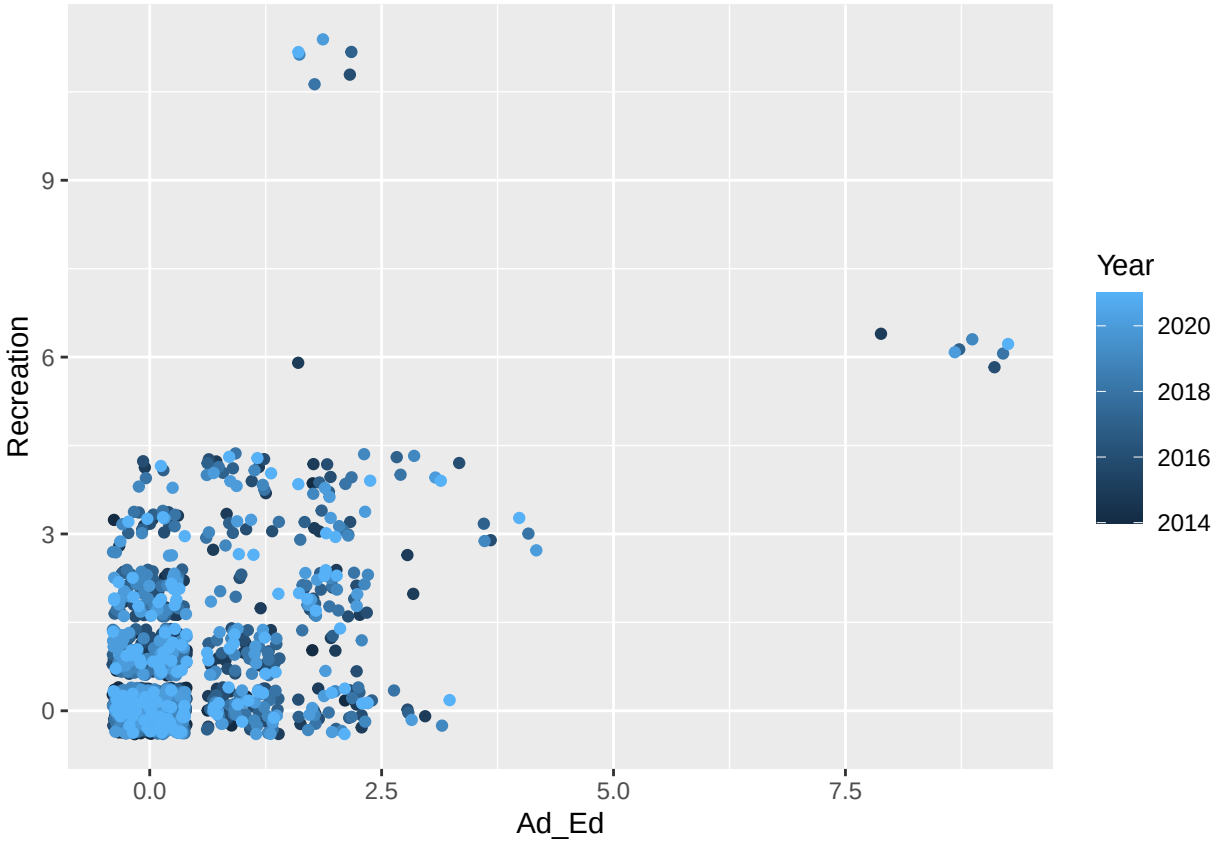

```
ggplot(df, aes(x= Ad_Ed, y = Emp_Res, colour = Year))+
  geom_jitter()
```



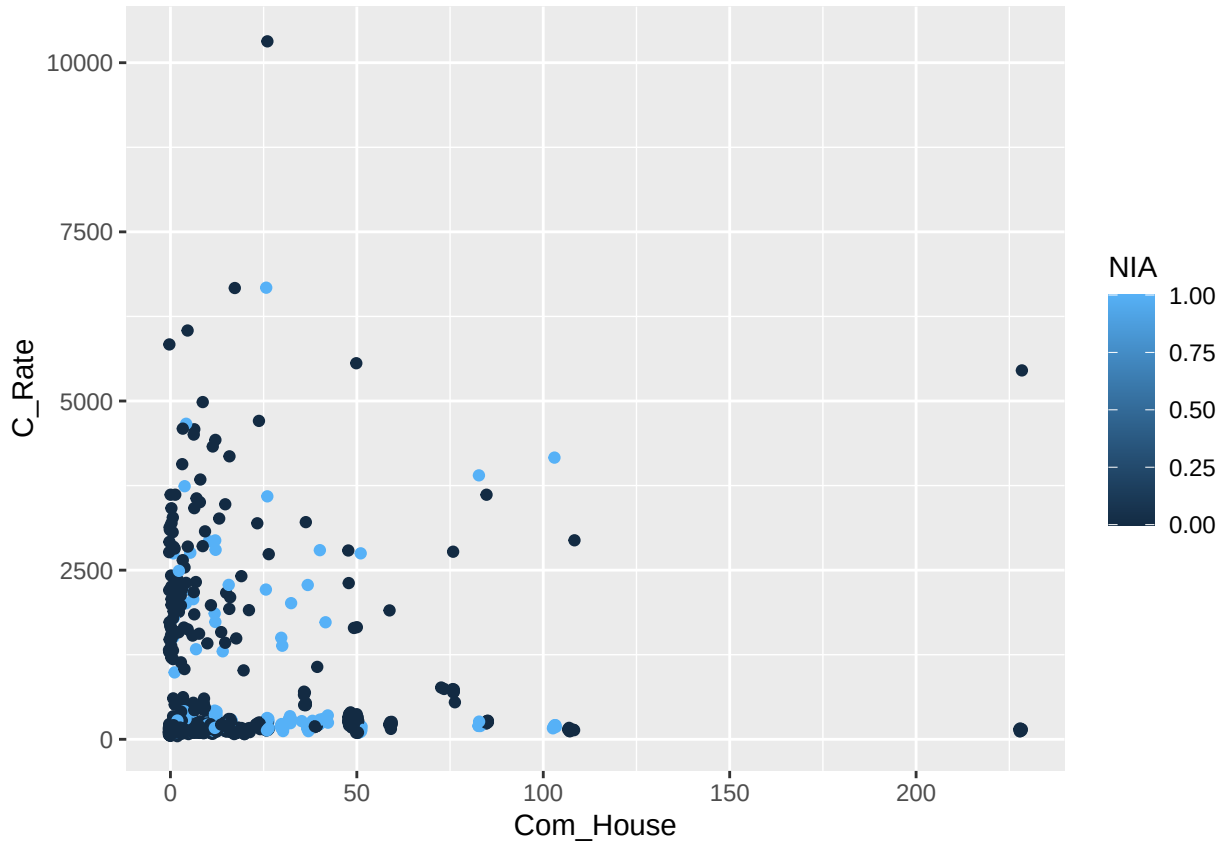
mild positive association between adult education facilities and employment resources. this is likely because the two services are probable to be placed in the same places as one would assume that accessing adult education would lead to being able to start looking for a job

likewise the mild association between adult education and recreation could be inferred as being a result of how adult education services are likely to be placed in areas where other recreation facilities exist.

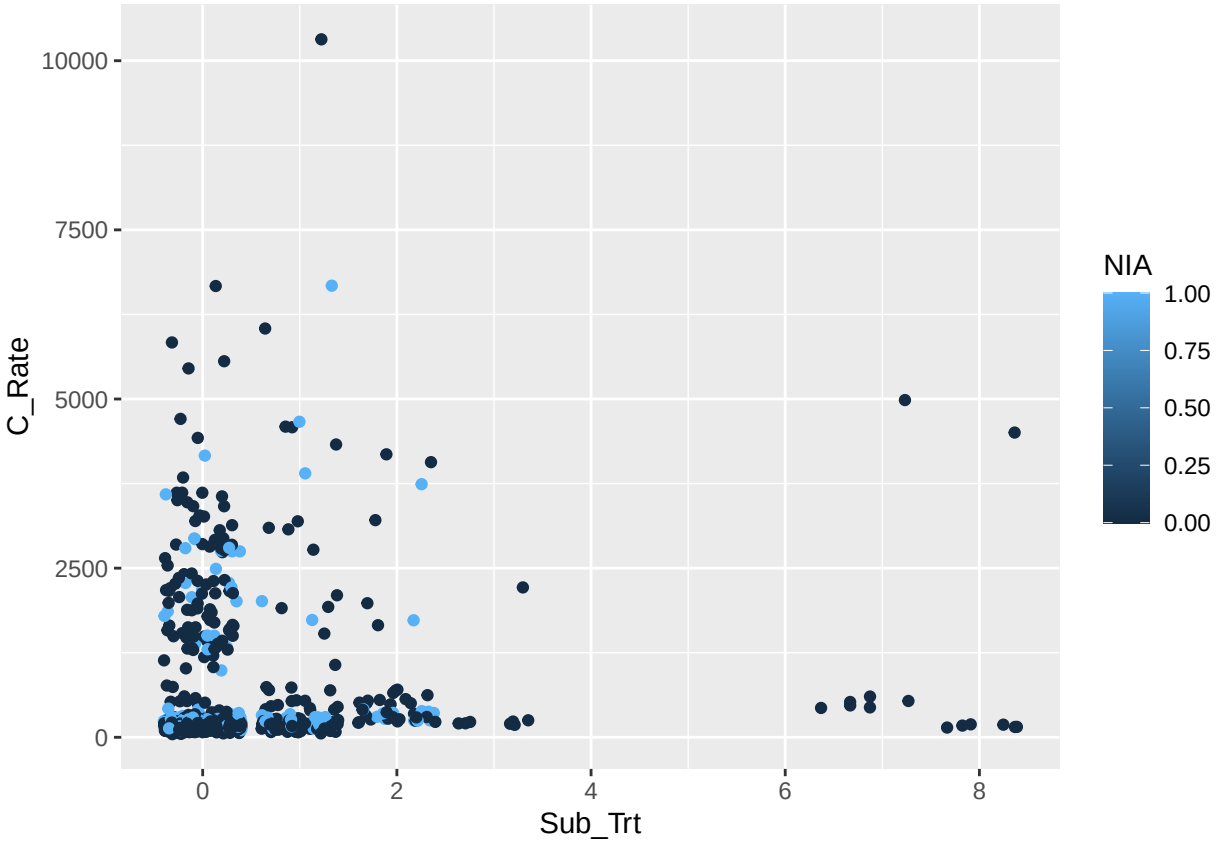
```
ggplot(df, aes(x= Ad_Ed, y = Recreation, colour = Year))+
  geom_jitter()
```




```
ggplot(df, aes(x = Com_House, y = C_Rate, colour = NIA))+
  geom_jitter()
```




```
ggplot(df, aes(x= Sub_Trt, y = C_Rate, colour = NIA))+
  geom_jitter()
```



From visualising our data and getting a better picture of what is going on, there are a few conclusions I have made. 1) Further changes to my data are needed. I feel that I may need to transform the other count data into rates to more accurately access how much they contribute to the response variable 2) I also need to incorporate more demographic data per neighbourhood. This is a little trickier to incorporate into my data as this data is only collected every 5 years, and thus doesnt give an accurate picture of what is going on in each neighbourhood each year. 3) This issue could be solved by expanding the time span of my study, however another obstacle that comes up when we do this is that the data collected by the city of toronto is no longer consistent. Different variables and different measures of each neighbourhood are taken and this makes it hard to accurately assess what exactly causes high and low crime rates.

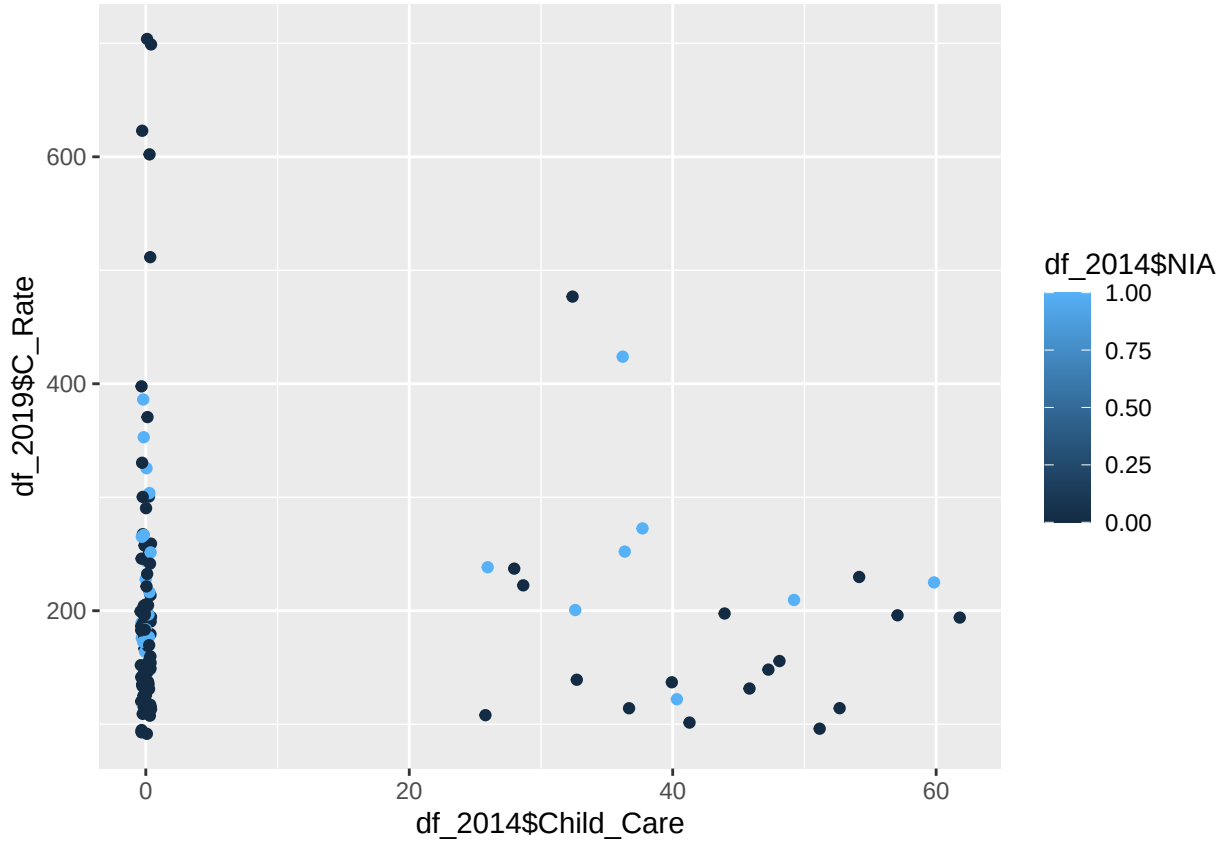
That being said, it may be easier to find relationships between crime rate and neighbourhood resources when measurements are taken years apart.

```
df_2014 <- df[df$Year == 2014,]
df_2015 <- df[df$Year == 2015,]
df_2016 <- df[df$Year == 2016,]
df_2017 <- df[df$Year == 2017,]
df_2018 <- df[df$Year == 2018,]
df_2019 <- df[df$Year == 2019,]
df_2020 <- df[df$Year == 2020,]
df_2021 <- df[df$Year == 2021,]
```

```
ggplot(df_2014, aes(x = df_2014$Child_Care, y = df_2019$C_Rate, colour = df_2014$NIA))+
  geom_jitter()
```

```
## Warning: Use of ‘df_2014$Child_Care‘ is discouraged. Use ‘Child_Care‘ instead.
```

```
## Warning: Use of ‘df_2014$NIA‘ is discouraged. Use ‘NIA‘ instead.
```



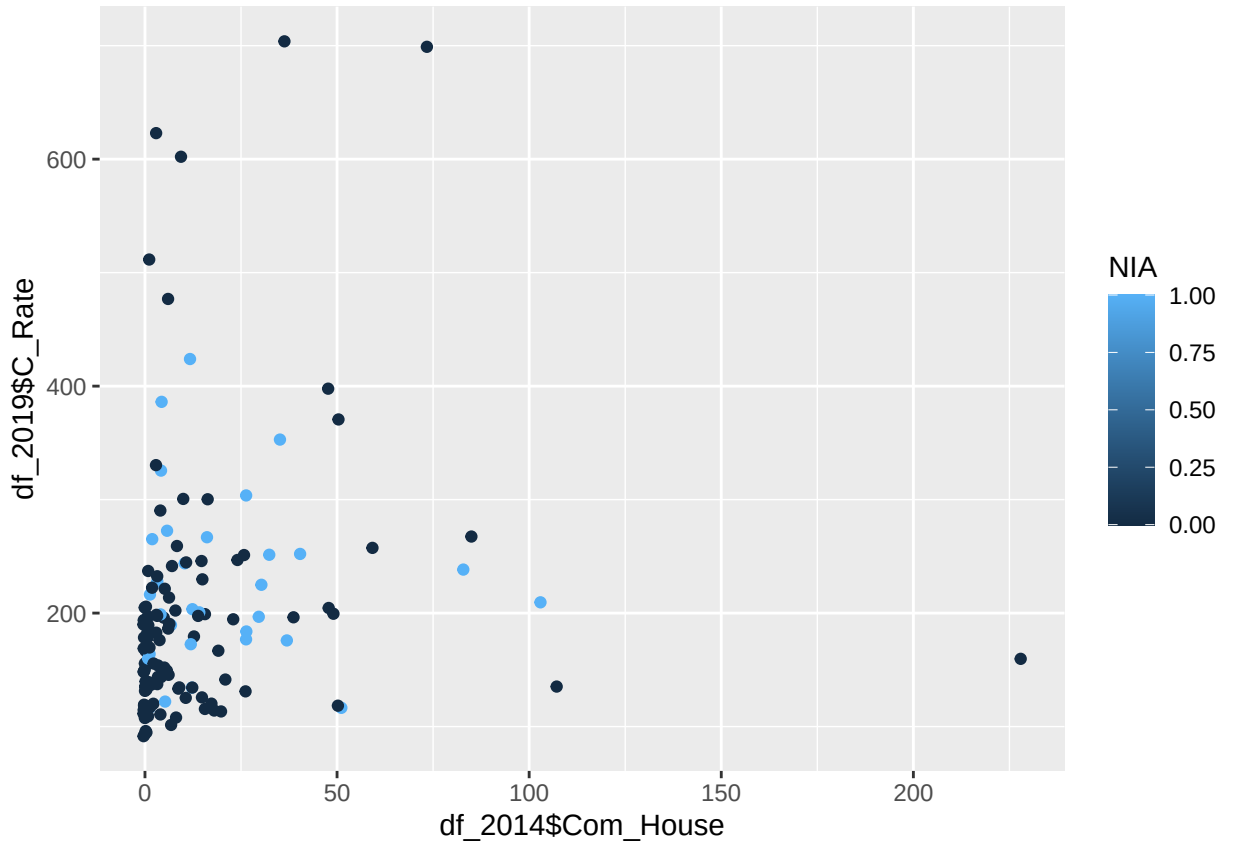
Here we can see that there is somewhat of a negative association. Where 2014 Child Care resources were low, 5 years later those same neighbourhoods had the highest crime rates. Likewise when there were ample child care resources in a neighbourhood, we an see that none of those data points were extremely high

```
cor(df_2014$Child_Care, df_2019$C_Rate)
```

```
## [1] -0.04248743
```

```
ggplot(df_2014, aes(x = df_2014$Com_House, y =df_2019$C_Rate, colour = NIA))+
  geom_jitter()
```

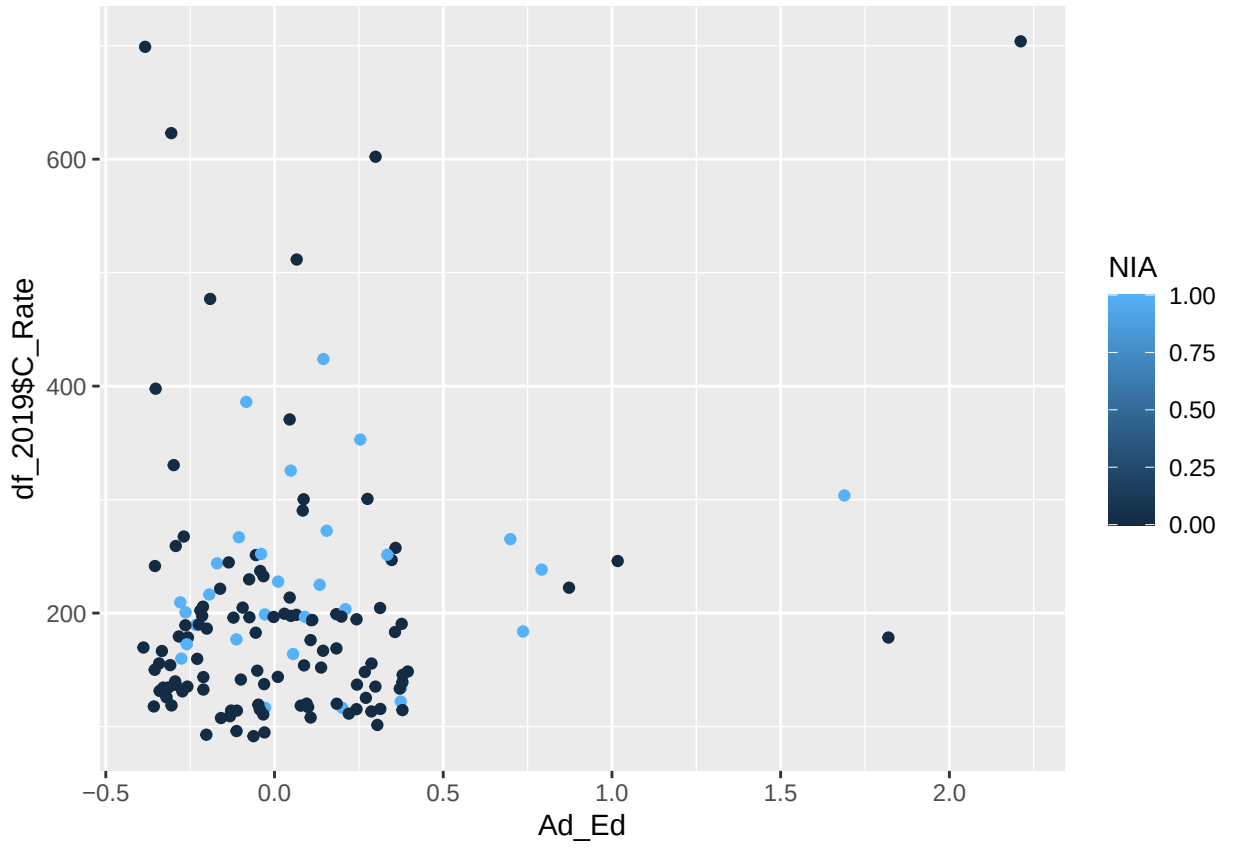
```
## Warning: Use of ‘df_2014$Com_House‘ is discouraged. Use ‘Com_House‘ instead.
```




```
cor(df_2014$Com_House, df_2019$C_Rate)
```

```
## [1] 0.164452
```

```
ggplot(df_2014, aes(x = Ad_Ed, y = df_2019$C_Rate, colour = NIA))+
  geom_jitter()
```




```
cor(df_2014$Ad_Ed, df_2019$C_Rate)
```

```
## [1] 0.251447
```

Looking at these results I can conclude the following: 1) The data paints a clearer picture of association when the response variable crime rate is looked some time (in this case five years) after the counts of the variables are collected. This makes sense as resources often need time to be implemented effectively and make an impact on their communities. 2) I would still try to obtain some more data on the individual neighbourhoods although, as mentioned earlier, this will be tricky due to consistency issues.

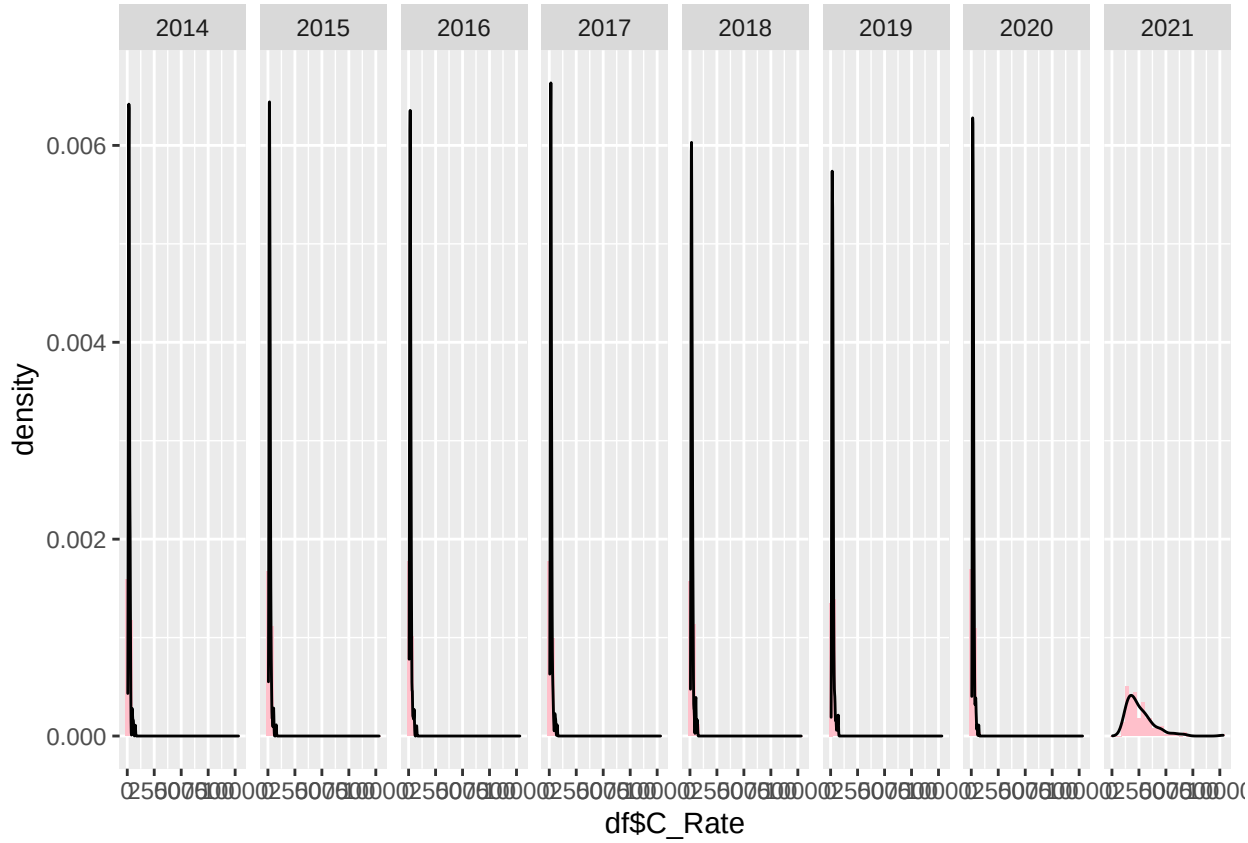
Initial Results and Code

As discussed in our literature review, we will first try to assess the association of resources avaialble within each neighbourhood and its crime rate. We initially wanted to look at this in 5 year increments (resources available in year X vs crime rate in year X+5) however, due to data availability we will reduce this to three year increments.

We will look at the distribution of our target variable

```
ggplot(df, aes(df$C_Rate))+
  geom_histogram(aes(y = ..density..),fill = 'pink')+
  geom_density()+
  facet_grid(~Year)
```

```
## Warning: Use of ‘df$C_Rate‘ is discouraged. Use ‘C_Rate‘ instead.

## Warning: Use of ‘df$C_Rate‘ is discouraged. Use ‘C_Rate‘ instead.

## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



We will try to apply a multinomial linear regression

```
model_1 <- glm(df_2020$C_Rate ~ df_2017$Child_Care + df_2017$Com_House + df_2017$Ad_Ed + df_2017$Emp_Res
summary(model_1)
```

```
##
## Call:
## glm(formula = df_2020$C_Rate ~ df_2017$Child_Care + df_2017$Com_House +
##     df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreation + df_2017$Sub_Trt,
##     data = df_2020)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -197.914   -38.745    -7.428    34.612   235.547
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        131.6565     8.3855  15.700  < 2e-16 ***
## df_2017$Child_Care   0.4137     0.3533   1.171 0.243726
```

```
## df_2017$Com_House   -0.1789     0.2324  -0.770 0.442749
## df_2017$Ad_Ed       23.0878     6.2240   3.709 0.000304 ***
## df_2017$Emp_Res      8.4815     3.8168   2.222 0.027964 *
## df_2017$Recreation  20.9337     4.7251   4.430 1.95e-05 ***
## df_2017$Sub_Trt     11.8012     6.1653   1.914 0.057751 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 4693.547)
##
##     Null deviance: 1144003  on 139  degrees of freedom
## Residual deviance:  624242  on 133  degrees of freedom
## AIC: 1589.7
##
## Number of Fisher Scoring iterations: 2
```

So here we can see that recreation and adult education are highly significant, whilst employment resources are slightly significant

what if we played arounbd with the gap of time between the responce and each attribute?

```
model_2 <- glm(df_2020$C_Rate ~ df_2014$Child_Care + df_2017$Com_House + df_2017$Ad_Ed + df_2017$Emp_Res
```

```
summary(model_2)
```

```
##
## Call:
## glm(formula = df_2020$C_Rate ~ df_2014$Child_Care + df_2017$Com_House +
##     df_2017$Ad_Ed + df_2017$Emp_Res + df_2018$Recreation + df_2014$Sub_Trt,
##     data = df_2020)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -210.277   -39.758    -9.227    34.069   231.796
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        132.0542     8.5080  15.521  < 2e-16 ***
## df_2014$Child_Care   0.3942     0.3575   1.102 0.272253
## df_2017$Com_House   -0.2208     0.2346  -0.941 0.348431
## df_2017$Ad_Ed       23.9338     6.3374   3.777 0.000239 ***
## df_2017$Emp_Res     10.3727     3.7303   2.781 0.006212 **
## df_2018$Recreation  21.6211     4.7854   4.518 1.36e-05 ***
## df_2014$Sub_Trt     11.4028    17.5408   0.650 0.516765
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 4807.571)
##
##     Null deviance: 1144003  on 139  degrees of freedom
## Residual deviance:  639407  on 133  degrees of freedom
## AIC: 1593
##
## Number of Fisher Scoring iterations: 2
```

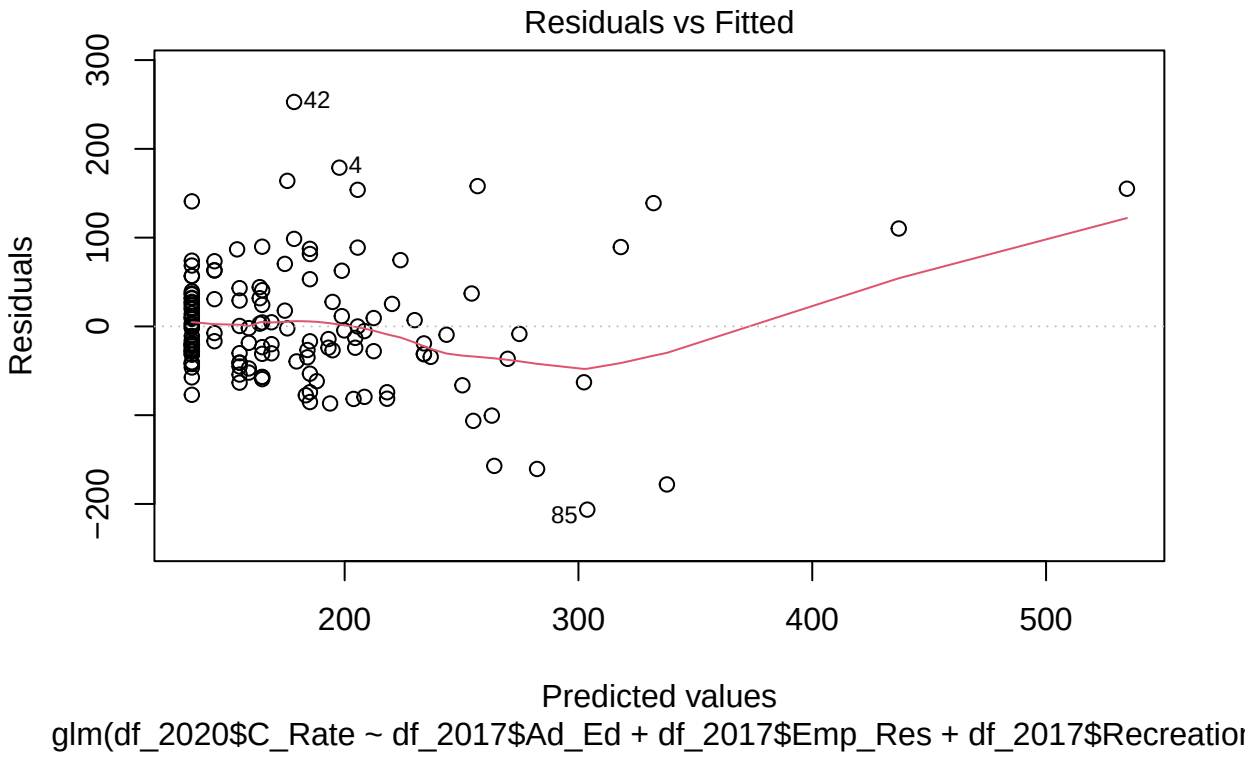
employment resources plays a more sgnificant role in this model when time between recreation facilties and time at which the response is measured is reduced.

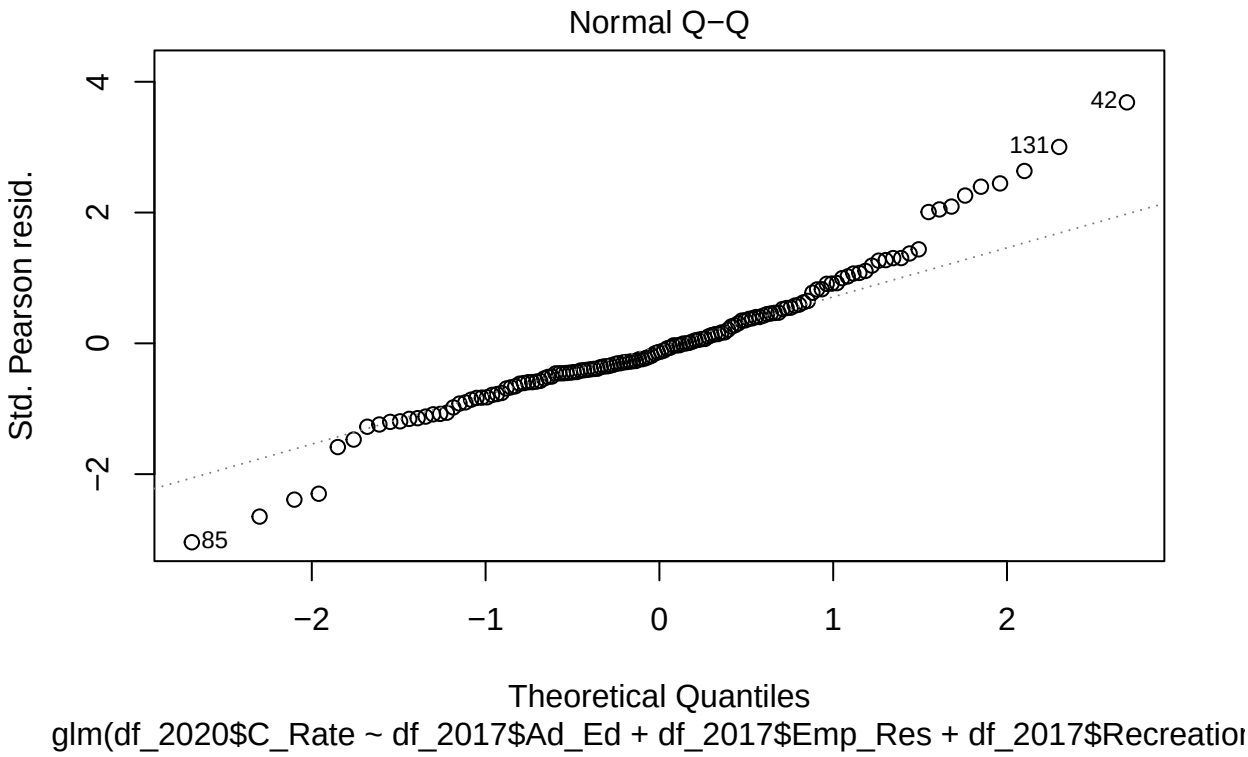
```
model_3 <- glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreation)
summary(model_3)
```

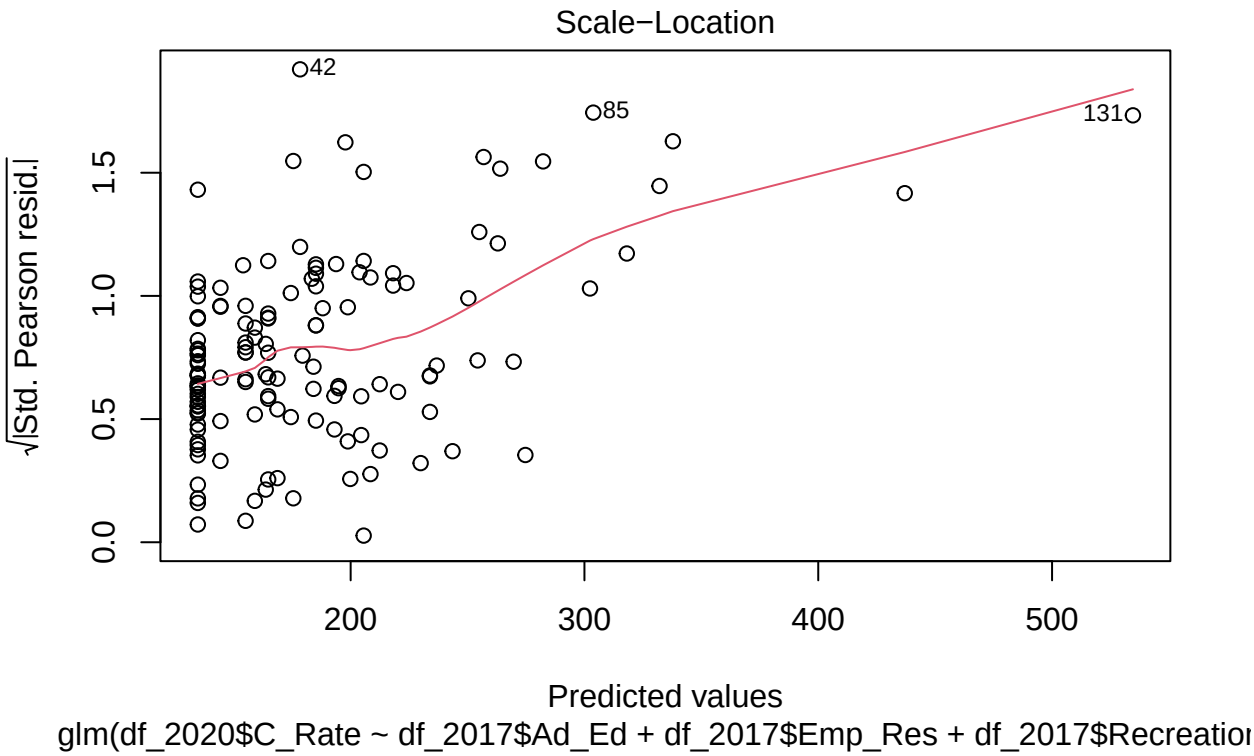
```
##
## Call:
## glm(formula = df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res +
##     df_2017$Recreation)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -206.378   -37.218    -8.911    31.889   252.839
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          134.645      7.652  17.596  < 2e-16 ***
## df_2017$Ad_Ed         24.376      6.229   3.913 0.000143 ***
## df_2017$Emp_Res        9.680      3.677   2.632 0.009463 **
## df_2017$Recreation    20.415      4.599   4.439 1.85e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 4790.882)
##
##     Null deviance: 1144003  on 139  degrees of freedom
## Residual deviance:  651560  on 136  degrees of freedom
## AIC: 1589.7
##
## Number of Fisher Scoring iterations: 2
```

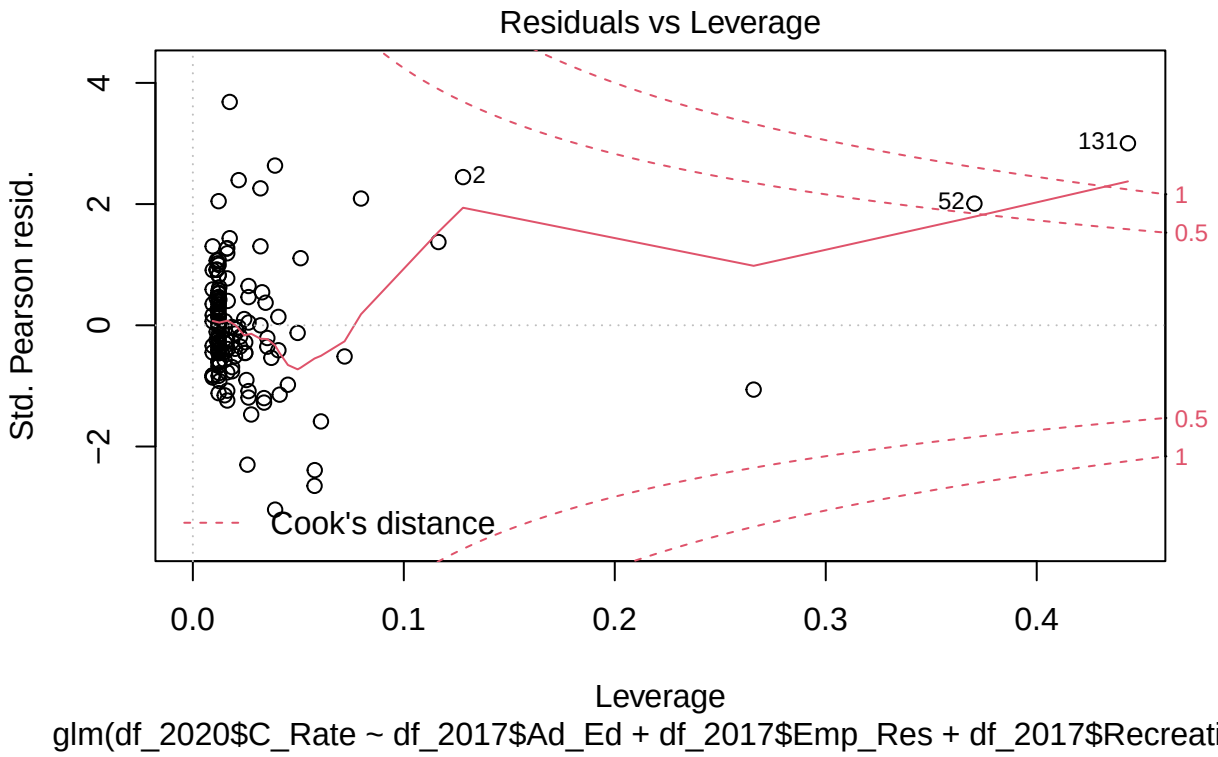
We want to check the normality of the residuals to ensure that our results are valid

```
plot(model_3)
```

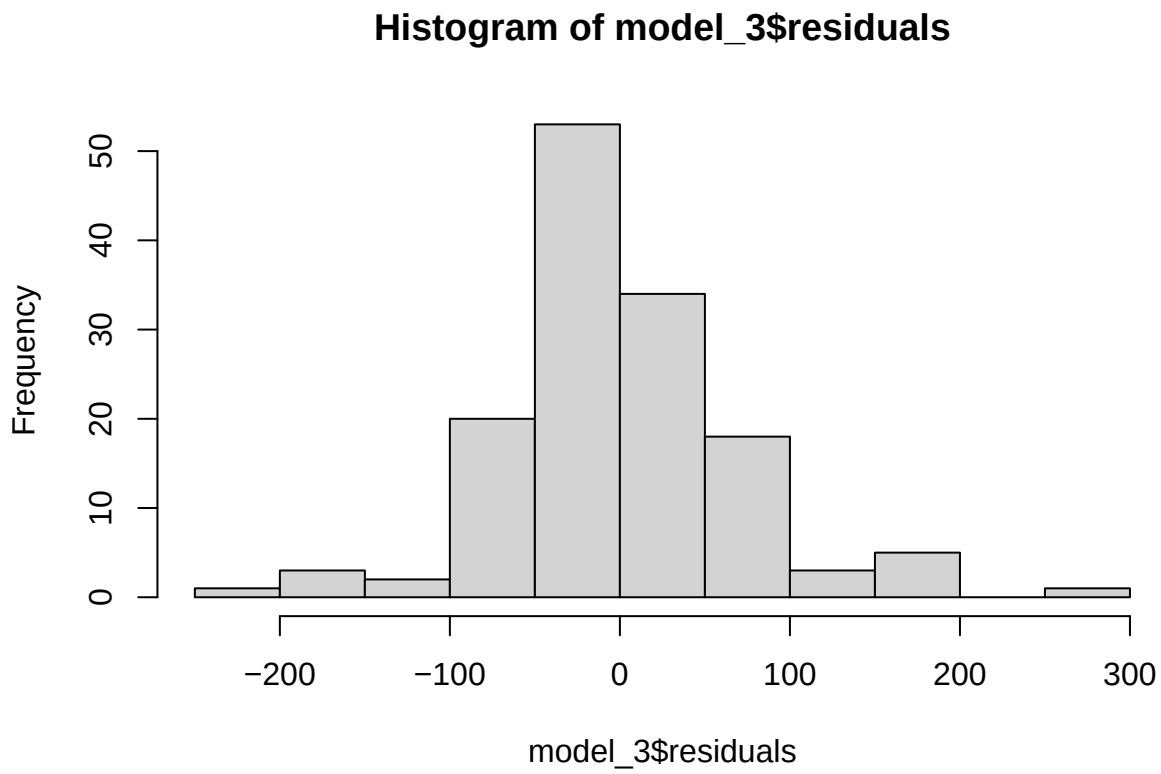

Residuals vs Fitted
Residuals
Predicted values
glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreatio
42
4
85
300
200
100
0
−200
200
300
400
500


Normal Q−Q
Std. Pearson resid.
Theoretical Quantiles
glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreatio
42
131
85
−2
0
2
4
−2
−1
0
1
2

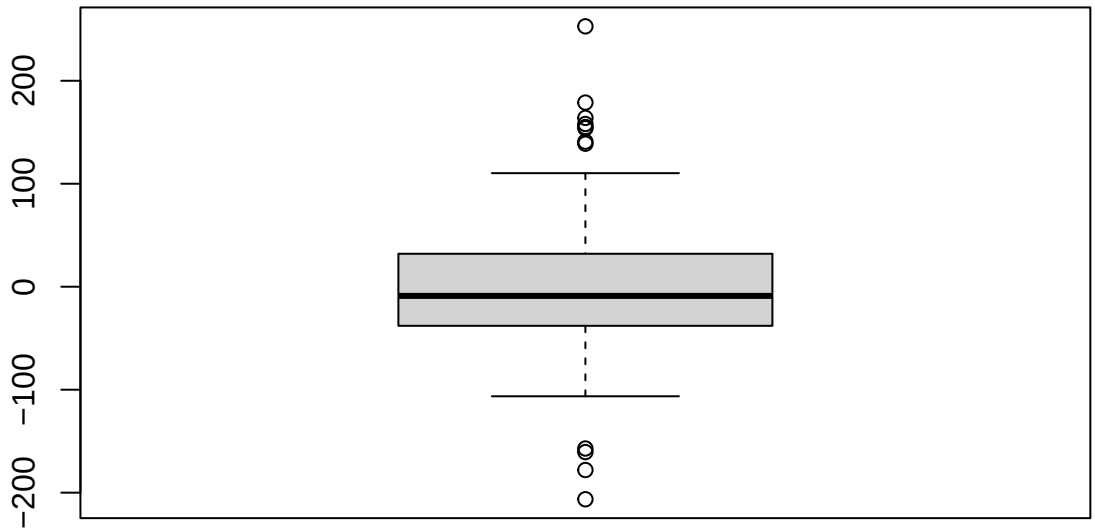

Scale−Location
42
85
131
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
200
300
400
500
Predicted values
glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreatio




```
hist(model_3$residuals)
```




```
boxplot(model_3$residuals)
```



we will preform the Kolmogorov-Smirnov test on the residuals

```
library(dgof)
```

```
## Warning: package 'dgof' was built under R version 4.1.2
```

```
##
## Attaching package: 'dgof'
```

```
## The following object is masked from 'package:stats':
##
##     ks.test
```

```
set.seed(1)
norm_dist <- rnorm(50)
ks.test(norm_dist, model_3$residuals)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  norm_dist and model_3$residuals
## D = 0.52857, p-value = 5.518e-10
## alternative hypothesis: two-sided
```

our p-value is very small –> so we should reject our null hypothesis that the two distributions are equal as there is sufficient evidence to suggest that the distribution of our residuals is not normal

unfortunately this means that we cannot use a multinomial linear regression.

Our next step is to use the Schapiro Wilk Test to test the normality of our dependent variable (crime rate)

```
library(dplyr)
library(ggpubr)
```

```
shapiro.test(df$C_Rate)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$C_Rate
## W = 0.4458, p-value < 2.2e-16
```

our p-value is very small, giving us sufficient evidence to suggest that our dependent variable Y is not normally distributed.

First we will investigate

We will now employ feature selection to reduce the dimensionality of our data.

We will try to transform our depndent variable from a continuous numerical variable to a categorical variable.

```
qs <- quantile(df$C_Rate,c(0,0.1,0.35,0.5,0.65,0.9,1.0))
qs
```

```
##          0%         10%         35%         50%         65%         90%
##    46.27219   105.57477   147.84343   175.53186   208.80148  1541.15435
##        100%
## 10315.87822
```

```
library(gtools)
```

```
## Warning: package 'gtools' was built under R version 4.1.2
```

```
df_new <- df
```

Now we turn C_Rate into a factor variable with the levels 'Very Low' (10th quantile), 'Low'(between 10th and 35th quantile),'Medium'(between 35th and 65th quantile),'High' (between 65th and 90th quantile) and 'Very High' (above the 90th quantile)

```
df_new$C_Rate <- quantcut(df_new$C_Rate, c(0,0.1,0.35,0.5,0.65,0.9,1.0))
```

```
levels(df_new$C_Rate)<- c('Very Low', 'Low', 'Medium', 'Medium', 'High', 'Very High')
```

Here we can explore two models. We can create one model where features and crime rate are from the same year, and one model where there is a three year gap between features and crime rate.

df_new is the data frame we will use to create the model where features and crime rate are both values from the same year. Here we can implement a 70/30 train/test split as we will have crime rate data available for every data instance.

The second model, df_2, where there is a 3 year gap between features and crime rate will have a 62.5/37.5 train/test split. This is because we have BOTH feature and crime rate data up until the feature year/ crime rate year combination of 2018/2021. For feature values in the year 2019 onwards there is no crime rate data available as that data hasnt been collected yet.

```
df_2 <- df_new
```

```
df_2[df_2$Year==2014,]$C_Rate <-df_new[df_new$Year==2017,]$C_Rate
df_2[df_2$Year==2015,]$C_Rate <-df_new[df_new$Year==2018,]$C_Rate
df_2[df_2$Year==2016,]$C_Rate <-df_new[df_new$Year==2019,]$C_Rate
df_2[df_2$Year==2017,]$C_Rate <-df_new[df_new$Year==2020,]$C_Rate
df_2[df_2$Year==2018,]$C_Rate <-df_new[df_new$Year==2021,]$C_Rate
```

```
df_2_train <- df_2[df_2$Year %in% c(2014,2015,2016,2017,2018),]
df_2_test <- df_2[df_2$Year %in% c(2019,2020,2021), !names(df_2) %in% c('C_Rate')]
```

```
dim(df_2)
```

```
## [1] 1120   13
```

```
dim(df_2_test)
```

```
## [1] 420  12
```

```
dim(df_2_train)
```

```
## [1] 700  13
```

We already implemented filter feature selection methods earlier see if we could reduce the dimensionality of our data. We will now apply wrapping methods to see if we can reduce dimensionality further.

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(class)
library(mlbench)
library(base)
```

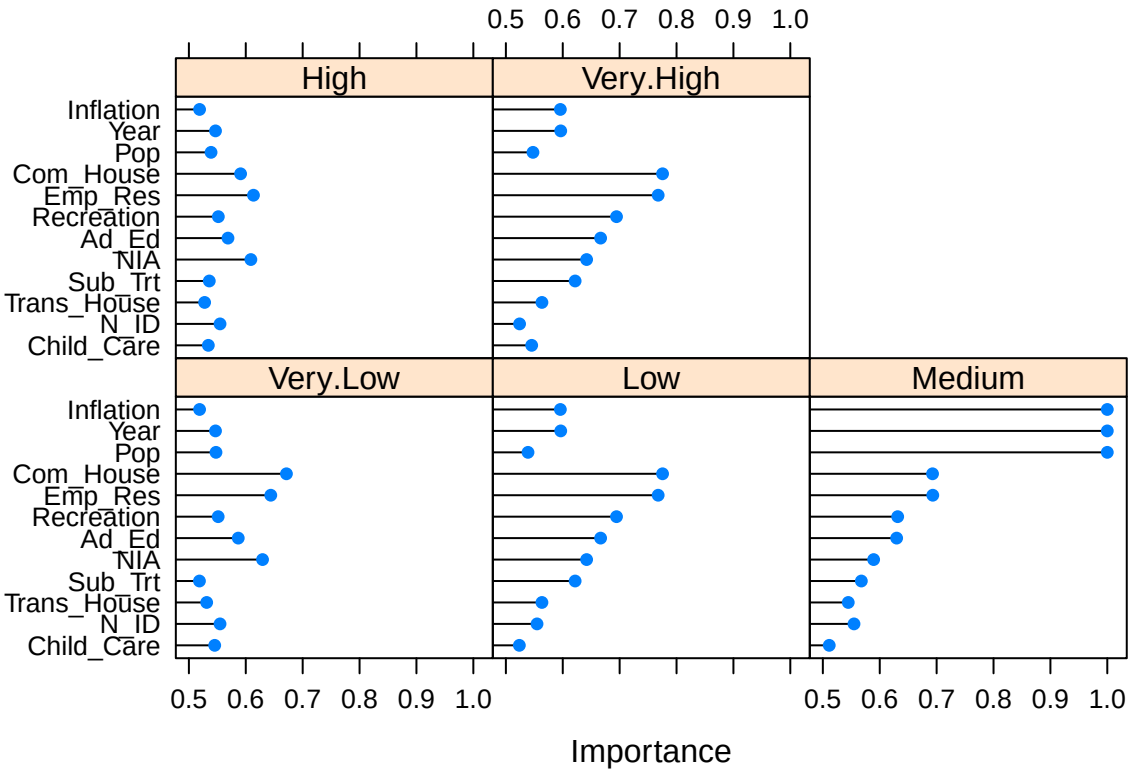
For df_new:

We will construct a Learning Vector QUantization model to estimate variable importance.LVQ is a classification algorithm and can be used for both binary and multiclass problems.

```
set.seed(7)
control <- trainControl(method = 'repeatedcv', number = 10, repeats = 3)
model <- train(C_Rate~., data = df_new, method = 'lvq', preProcess = 'scale', trControl = control)
importance <- varImp(model, scale = FALSE)
print(importance)
```

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##              Very.Low    Low Medium   High Very.High
## Year           0.5468 0.5964 1.0000 0.5468    0.5964
## Inflation      0.5188 0.5958 1.0000 0.5188    0.5958
## Pop            0.5476 0.5389 1.0000 0.5389    0.5476
## Com_House      0.6715 0.7754 0.6928 0.5909    0.7754
## Emp_Res        0.6440 0.7677 0.6932 0.6136    0.7677
## Recreation     0.5515 0.6945 0.6315 0.5517    0.6945
## Ad_Ed          0.5868 0.6666 0.6297 0.5687    0.6666
## NIA            0.6295 0.6420 0.5893 0.6089    0.6420
## Sub_Trt        0.5185 0.6218 0.5677 0.5359    0.6218
## Trans_House    0.5312 0.5634 0.5446 0.5277    0.5634
## N_ID           0.5548 0.5548 0.5548 0.5548    0.5241
## Child_Care     0.5454 0.5237 0.5112 0.5342    0.5454
```

```
plot(importance)
```

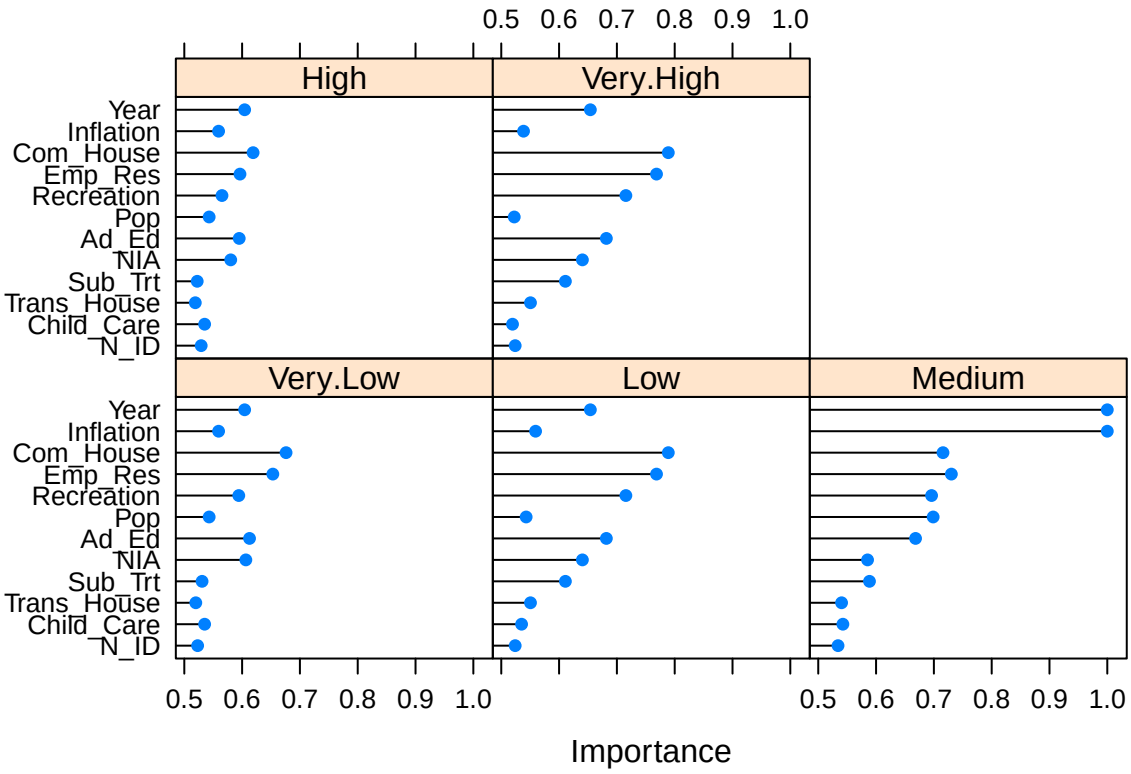


For df_2:

```
set.seed(7)
control <- trainControl(method = 'repeatedcv', number = 10, repeats = 3)
model <- train(C_Rate~., data = df_2_train, method = 'lvq', preProcess = 'scale', trControl = control)
importance <- varImp(model, scale = FALSE)
print(importance)
```

```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##              Very.Low    Low Medium   High Very.High
## Inflation      0.5593 0.5593 1.0000 0.5593    0.5385
## Year           0.6044 0.6541 1.0000 0.6044    0.6541
## Com_House      0.6760 0.7889 0.7158 0.6190    0.7889
## Emp_Res        0.6531 0.7686 0.7303 0.5963    0.7686
## Recreation     0.5943 0.7155 0.6960 0.5652    0.7155
## Pop            0.5430 0.5430 0.6988 0.5430    0.5225
## Ad_Ed          0.6128 0.6818 0.6683 0.5950    0.6818
## NIA            0.6065 0.6404 0.5853 0.5803    0.6404
## Sub_Trt        0.5307 0.6109 0.5886 0.5224    0.6109
## Trans_House    0.5199 0.5506 0.5403 0.5190    0.5506
## Child_Care     0.5351 0.5351 0.5426 0.5351    0.5195
## N_ID           0.5231 0.5240 0.5342 0.5292    0.5240
```

```
plot(importance)
```



Further Feature Selection and Model Fitting

Decision Trees and Permutation Feature importance

We then decided to further reduce dimensionality of our data by conducting more feature selection using decision trees/random forest. We aimed to keep our models simple by ensuring our decision trees were small to accommodate for our small datasets.

We started our trees using the features we had obtained from using the LVQ and got further insight into the importance of our features using permutation feature importance. Our results helped us reverse/forward engineer our decision tree models until the highest level of accuracy possible was obtained.

Multinomial Logistic Model Fitting

From our analysis using decision trees and permutation feature importance in python, we were able remove some features that are unlikely to contribute to our to our models. We then proceeded to try to fit a multinomial regression model to our datasets. Our value for accuracy for the data with no time gap was 42.37 for our training data and 41.32 for our test data. We repeated the process for our data with the three year time gap. For our time

gap data our test set consists of data points from the year 2019 onwards, and thus we do not have any actual value of data for our test set. As such, we were only able to calculate accuracy for our training data which, whilst still fairly low, was significantly better than that of our no gap data at 54.29.

Creation of Synthetic Data and Further Model Fitting

As our accuracy scores were fairly low, we considered creating synthetic data to our data set. This would not only help with issues of class imbalance, but it would also increase the number of datapoints in our set. We implemented the synthetic minority sampling technique (SMOTE) to assist us, and proceeded to retry fitting our models to the data.

Decision Trees and Further Feature Selection

As expected, upon rerunning and refitting our models our accuracy scores increased notably when datasets including synthesized data were utilized. We obtained slightly different results for feature selection using the new datasets as well, and used these to reduce dimensionality of our datasets again before refitting our multinomial regression models.

For our data with no time our accuracy increased from 42.37 to 49.96 for our training data, and from 41.32 to 49.5 for our test data.
Likewise for our data with the time gap our accuracy was 54.69, compared to 54.29 before adding synthesized data.

We concluded that the models using the synthesized data were the most effective, especially when a three year time gap between feature values and response values were put in place.

Feature Selection Using Decision Tree/Random Forest

First we will do this for the regular crime data (no three year gap between features and crime rate) and then we will do it for the three year gap data.

In [1]:

```
import pandas as pd
import time
```

In [2]:

```
Data_1 = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data.csv')
```

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [4]:

```
Data_1 = Data_1.drop('Unnamed: 0',axis = 1)
X_train, X_test, y_train, y_test = train_test_split(Data_1.drop('C_Rate',axis=1), Data_1
['C_Rate'], test_size=.3,random_state=22)
```

In [5]:

```
X_train.shape, X_test.shape
```

Out[5]:

```
((784, 12), (336, 12))
```

In [6]:

```
start = time.time_process()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 start = time.time_process()

AttributeError: module 'time' has no attribute 'time_process'
```

In [ ]:

```
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, mi
n_samples_leaf=5)
```

In [ ]:

```
model.fit(X_train, y_train)
```

In [ ]:

```
import io
from io import StringIO
from sklearn.tree import export_graphviz
# from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
import graphviz
from sklearn.metrics import classification_report, accuracy_score
from pydot import graph_from_dot_data
```

In [ ]:

```
xvar = Data_1.drop('C_Rate', axis = 1)
feature_cols = xvar.columns
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True, special_chara
cters = True, feature_names = feature_cols, class_names = ['Very Low','Low','Medium','Hi
gh','Very High'])
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Population appears to be the most important feature here

In [ ]:

```
predictions = model.predict(X_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model.predict(X_train)))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

In [ ]:

```
#runtime for model
t1 = time.process_time() - start
print(t1)
```

In [ ]:

```
print(classification_report(y_test,predictions))
```

We will now use permutation feature importance to further evaluate the importance of our features

In [ ]:

```
model.score(X_test,y_test)
```

In [ ]:

```
from sklearn.inspection import permutation_importance
```

running permutation importance on our test set:

In [ ]:

```
r = permutation_importance(model, X_test, y_test, n_repeats = 30, random_state = 0)
```

In [ ]:

```
for i in r.importances_mean.argsort()[::-1]:
    if r.importances_mean[i] - 2 * r.importances_std[i] > 0:
        print(f"{list(Data_1.columns.values)[i]:<8}"
              f"{r.importances_mean[i]:.3f}"
              f" +/- {r.importances_std[i]:.3f}")
```

running it on our train set:

In [ ]:

```
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [ ]:

```
for i in r_1.importances_mean.argsort()[::-1]:
    if r_1.importances_mean[i] - 2 * r_1.importances_std[i] > 0:
        print(f"{list(Data_1.columns.values)[i]:<8}"
              f"{r_1.importances_mean[i]:.3f}"
              f" +/- {r_1.importances_std[i]:.3f}")
```

Lets see how our model score changes keeping only the most "important" features. We will start by only keeping pop com_house and child_care

In [ ]:

```
train_1 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recr
eation','Inflation','NIA'], axis = 1)
test_1 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recrea
tion','Inflation','NIA'], axis = 1)
```

In [ ]:

```
model_1 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,
min_samples_leaf=5)
```

In [ ]:

```
model_1.fit(train_1, y_train)
```

In [ ]:

```
xvar_1 = Data_1.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recrea
tion','Inflation','NIA','C_Rate'], axis = 1)
feature_cols_1 = xvar_1.columns
dot_data_1 = StringIO()
export_graphviz(model_1, out_file = dot_data_1, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_1, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_1,)= graph_from_dot_data(dot_data_1.getvalue())
Image(graph_1.create_png())
```

In [ ]:

```
predictions = model_1.predict(test_1)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_1.predict(train_1)
))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_1.predict(test_1)))
```

In [ ]:

```
#runtime for model
t2 = time.process_time() - t1
print(t2)
```

we will add in inflation

In [ ]:

```
train_2 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recr
eation','NIA'], axis = 1)
test_2 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recrea
tion','NIA'], axis = 1)

model_2 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,
min_samples_leaf=5)

model_2.fit(train_2, y_train)

xvar_2 = Data_1.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recrea
tion','NIA','C_Rate'], axis = 1)
feature_cols_2 = xvar_2.columns
dot_data_2 = StringIO()
export_graphviz(model_2, out_file = dot_data_2, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_2, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

In [ ]:

```
predictions = model_2.predict(test_2)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_2.predict(train_2)
))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_2.predict(test_2)))
```

In [ ]:

```
#runtime for model
t3 = time.process_time() - t2
print(t3)
```

Accuracy didnt really change at all. So inflation may not be contributing to our model. We remove inflation and add Ad_Ed

In [ ]:

```
train_3 = X_train.drop(['Year', 'N_ID', 'Inflation','Emp_Res','Sub_Trt', 'Trans_House','
Recreation','NIA'], axis = 1)
test_3 = X_test.drop(['Year', 'N_ID', 'Inflation','Emp_Res','Sub_Trt', 'Trans_House','Re
creation','NIA'], axis = 1)

model_3 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,
min_samples_leaf=5)

model_3.fit(train_3, y_train)

xvar_3 = Data_1.drop(['Year', 'N_ID', 'Inflation','Emp_Res','Sub_Trt', 'Trans_House','Re
creation','NIA','C_Rate'], axis = 1)
feature_cols_3 = xvar_3.columns
dot_data_3 = StringIO()
export_graphviz(model_3, out_file = dot_data_3, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_3, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_3,)= graph_from_dot_data(dot_data_3.getvalue())
Image(graph_3.create_png())
```

In [ ]:

```
predictions = model_3.predict(test_3)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_3.predict(train_3)
))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_3.predict(test_3)))
```

In [ ]:

```
#runtime for model
t3 = time.process_time() - t2
print(t3)
```

According to our results, Com_House, Child_Care and Pop appear to be the most importnat features for this model

Feature Selection Using Decision Tree/Random Forest

We will now conduct feature selection using decision trees on the data where there is a three year gap between feature values and crime rate.

In [1]:

```
import pandas as pd
import time
```

In [2]:

```
Train_Data = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Year_Gap_Train.csv')
Test_Data = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Year_Gap_Test.csv')
```

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [4]:

```
Train_Data = Train_Data.drop('Unnamed: 0',axis = 1)
Test_Data = Test_Data.drop('Unnamed: 0', axis = 1)
X_train = Train_Data.loc[:,Train_Data.columns!='C_Rate']
X_test = Test_Data
y_train = Train_Data['C_Rate']
```

In [5]:

```
X_train.shape, X_test.shape, y_train.shape
```

Out[5]:

```
((700, 12), (420, 12), (700,))
```

In [6]:

```
start = time.process_time()
```

In [7]:

```
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, min_samples_leaf=5)
```

In [8]:

```
model.fit(X_train, y_train)
```

Out[8]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [9]:

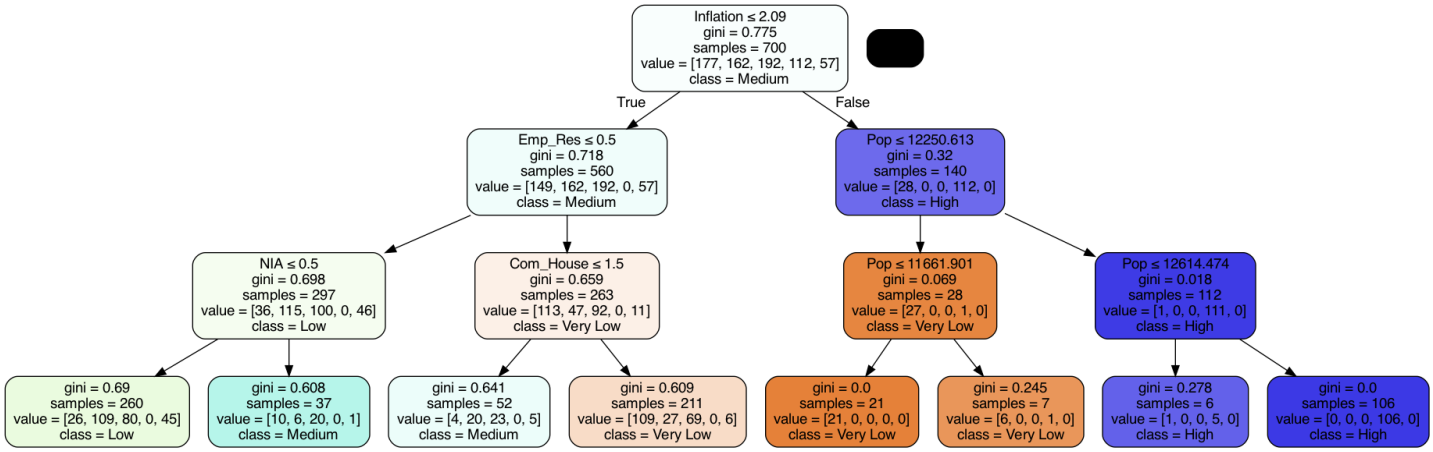
```
import io
from io import StringIO
from sklearn.tree import export_graphviz
# from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
import graphviz
```

```
from sklearn.metrics import classification_report, accuracy_score
from pydot import graph_from_dot_data
```

In [10]:

```
xvar = Train_Data.drop('C_Rate', axis = 1)
feature_cols = xvar.columns
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True, special_chara
cters = True, feature_names = feature_cols, class_names = ['Very Low','Low','Medium','Hi
gh','Very High'])
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[10]:



In [11]:

```
predictions = model.predict(X_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model.predict(X_train)))
```

```
Decision Tree Train Accuracy: 0.57
```

In [12]:

```
#runtime for model
t1 = time.process_time() - start
print(t1)
```

```
0.3821620000000001
```

We will now use permutation feature importance to further evaluate the importance of our features

In [13]:

```
from sklearn.inspection import permutation_importance
```

We will run it on our train set as we dont have response values for our test set

In [14]:

```
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [15]:

```
for i in r_1.importances_mean.argsort()[::-1]:
    if r_1.importances_mean[i] - 2 * r_1.importances_std[i] > 0:
        print(f"{list(Train_Data.columns.values)[i]:<8}"
              f"{r_1.importances_mean[i]:.3f}"
              f" +/- {r_1.importances_std[i]:.3f}")
```

```
Recreation0.216 +/- 0.010
Com_House0.088 +/- 0.014
Pop     0.067 +/- 0.008
Child_Care0.040 +/- 0.012
```

```
Inflation0.024 +/- 0.011
```

Lets see if dropping the "unneccessary" columns improves our model.

In [16]:

```
Train_Data.columns.values
```

Out[16]:

```
array(['Year', 'N_ID', 'Pop', 'C_Rate', 'Ad_Ed', 'Child_Care',
       'Com_House', 'Emp_Res', 'Sub_Trt', 'Trans_House', 'Recreation',
       'Inflation', 'NIA'], dtype=object)
```

In [17]:

```
new_train = X_train.drop(['Year', 'N_ID','Ad_Ed','Emp_Res', 'Sub_Trt', 'Trans_House','NI
A'], axis = 1)
new_test = Test_Data.drop(['Year', 'N_ID','Ad_Ed','Emp_Res', 'Sub_Trt', 'Trans_House','NI
A'], axis = 1)
```

In [18]:

```
new_train.shape, y_train.shape, new_test.shape
```

Out[18]:

```
((700, 5), (700,), (420, 5))
```

In [19]:

```
model_2 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
min_samples_leaf=5)
```

In [20]:

```
model_2.fit(new_train, y_train)
```

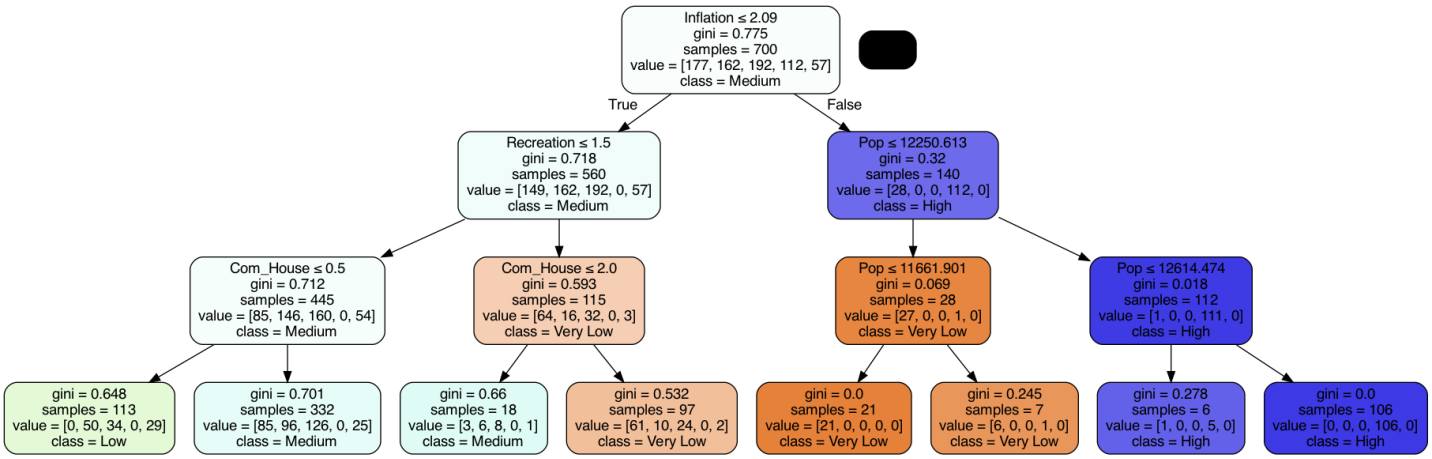
Out[20]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [21]:

```
new_var = new_train
new_feature_cols = new_var.columns
new_dot_data = StringIO()
export_graphviz(model_2, out_file = new_dot_data, filled = True, rounded = True, special
_characters = True, feature_names = new_feature_cols, class_names = ['Very Low','Low','M
edium','High','Very High'])
(graph_1,)= graph_from_dot_data(new_dot_data.getvalue())
Image(graph_1.create_png())
```

Out[21]:



In [22]:

```
predictions = model_2.predict(new_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_2.predict(new_trai
n)))
```

```
Decision Tree Train Accuracy: 0.5471428571428572
```

In [23]:

```
#runtime for model
t2 = time.process_time() - t1
print(t2)
```

```
3.891465
```

Our accurcay for our model on our train set reduced. In the original model Emp_Res was pretty high up in the tree, lets see what happens if we add it back in.

In [24]:

```
train_2 = X_train.drop(['Year', 'N_ID','Ad_Ed', 'Sub_Trt', 'Trans_House','NIA'], axis =
1)
test_2 = Test_Data.drop(['Year', 'N_ID','Ad_Ed', 'Sub_Trt', 'Trans_House','NIA'], axis =
1)
```

In [25]:

```
model_3 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
min_samples_leaf=5)
```

In [26]:

```
model_3.fit(train_2, y_train)
```

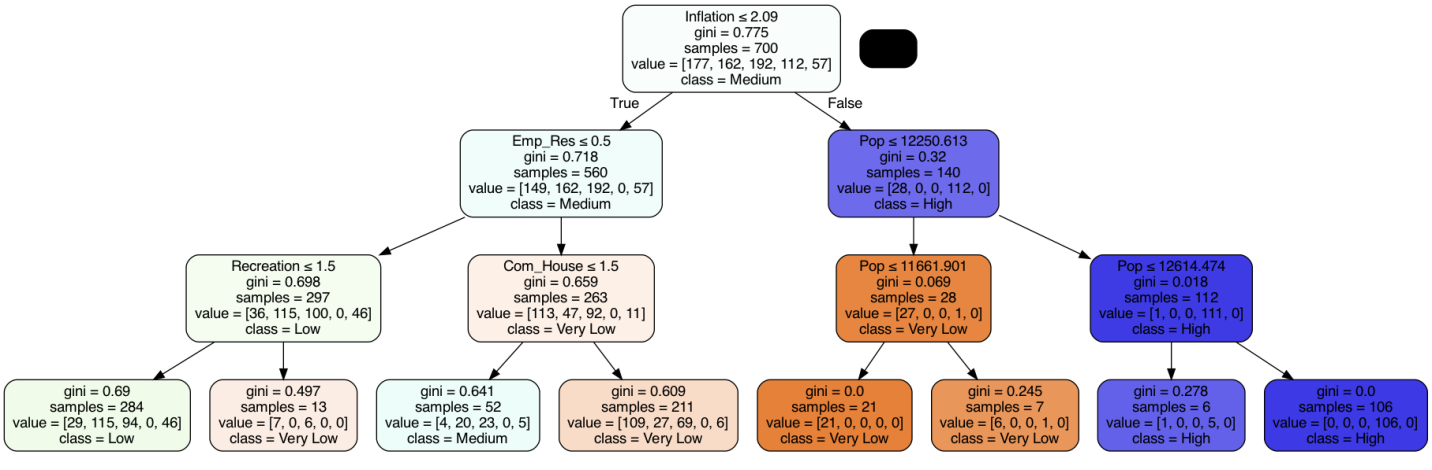
Out[26]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [27]:

```
var_2 = train_2
feature_cols_2 = var_2.columns
dot_data_2 = StringIO()
export_graphviz(model_3, out_file = dot_data_2, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_2, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

Out[27]:



In [28]:

```
predictions = model_3.predict(test_2)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_3.predict(train_2)
))
```

```
Decision Tree Train Accuracy: 0.56
```

In [29]:

```
#runtime for model
t3 = time.process_time() - t2
print(t3)
```

```
0.5457489999999998
```

Post feature Selection

Alyzeh Jiwani

27/06/2022

Post Feature Selection

From our analysis using decision trees and permutation fetaure importance in python, we are able to now remove some features that are unlikely to contribute to our to our models.

For df_new, where the values for a ll the features and our response variable crime_rate are taken in the same year, we decided to keep the following features: Com_House, Child_Care, Pop

For df_2, where values for features are taken three years prior to values for our response variable crime_rate, the features we have decided to keep are: Inflation, Recreation, Com_House, Pop,Emp_Res and Child_Care

```
Data_Current <- read.csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data.csv')
Data_Current <- Data_Current[,c('Com_House','Child_Care','Pop','C_Rate')]
Data_Gap_Train <- read.csv('//Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Year_Gap_Tra
Data_Gap_Test <- read.csv('//Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Year_Gap_Test
Data_Gap_Train <- Data_Gap_Train[,c('Inflation','Recreation','Com_House','Child_Care','Pop','Emp_Res','
Data_Gap_Test <- Data_Gap_Test[,c('Inflation','Recreation','Com_House','Child_Care','Pop','Emp_Res')]
```

We will now try to fit the data to a multinomial regression model

```
library(nnet)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

Splitting into Train/Test

```
index <- createDataPartition(Data_Current$C_Rate, p = 0.7, list = FALSE)
train <- Data_Current[index,]
test <- Data_Current[-index,]
```

```
model_cur_1 <- multinom(C_Rate~., data = Data_Current)
```

```
## # weights:  25 (16 variable)
## initial  value 1802.570462
## iter  10 value 1631.449756
## iter  20 value 1428.371682
## iter  30 value 1380.289944
```

```
## iter  40 value 1379.955843
## iter  50 value 1379.890921
## final  value 1379.875847
## converged
```

```
summary(model_cur_1)
```

```
## Call:
## multinom(formula = C_Rate ~ ., data = Data_Current)
##
## Coefficients:
##           (Intercept)    Com_House   Child_Care           Pop
## Low         0.1417881 -0.009605888 -0.002917352  1.537124e-06
## Medium      0.2372870 -0.006698838  0.008215376 -6.143905e-07
## Very High   3.7080610  0.030277578 -0.024117034 -8.310479e-04
## Very Low   -0.7734812 -0.051137604 -0.008382748  2.029466e-05
##
## Std. Errors:
##            (Intercept)   Com_House  Child_Care          Pop
## Low       6.310972e-05 0.003474224 0.005148986 4.660281e-06
## Medium    8.510603e-05 0.003005985 0.004527637 4.526159e-06
## Very High 2.349552e-04 0.016378640 0.010831335 9.470085e-05
## Very Low  8.717877e-05 0.011527159 0.007461553 6.219032e-06
##
## Residual Deviance: 2759.752
## AIC: 2791.752
```

```
exp(coef(model_cur_1))
```

```
##           (Intercept) Com_House Child_Care       Pop
## Low          1.152332 0.9904401  0.9970869 1.0000015
## Medium       1.267805 0.9933235  1.0082492 0.9999994
## Very High   40.774667 1.0307406  0.9761715 0.9991693
## Very Low     0.461404 0.9501479  0.9916523 1.0000203
```

These are the probabilities of neighbourhoods being having a particular crime rate level

```
head(round(fitted(model_cur_1),3))
```

```
##    High   Low Medium Very High Very Low
## 1 0.249 0.292  0.314     0.001    0.144
## 2 0.250 0.288  0.304     0.000    0.158
## 3 0.247 0.291  0.310     0.000    0.152
## 4 0.369 0.274  0.337     0.001    0.019
## 5 0.229 0.239  0.418     0.000    0.114
## 6 0.284 0.294  0.321     0.000    0.101
```

We now want to see what the accuracy of the model is.

```
train$C_RatePred <- predict(model_cur_1, newdata = train, 'class')
tab <- table(train$C_Rate, train$C_RatePred)
tab
```

```
##
##             High Low Medium Very High Very Low
##   High        42   1    135        18        0
##   Low         16   1    176         0        3
##   Medium      24   0    212         0        0
##   Very High    0   0      1        78        0
##   Very Low     4   1     74         0        0
```

Now we calculate accuracy

```
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 42.37
```

Our accuracy is at a value of 41.98.

We now predict on the test dataset and see our classification table

```
test$C_RatePred <- predict(model_cur_1, newdata = test, "class")


tab_test <- table(test$C_Rate, test$C_RatePred)
tab_test
```

```
##
##             High Low Medium Very High Very Low
##   High        16   1     57        10        0
##   Low          3   1     80         0        0
##   Medium      12   0     88         0        0
##   Very High    0   0      0        33        0
##   Very Low     2   0     31         0        0
```

accuracy of train model

```
round((sum(diag(tab_test))/sum(tab_test))*100,2)
```

```
## [1] 41.32
```

We now repeat this for the data where values of our features are taken three years prior to the values of our response variable.

```
model_gap_1 <- multinom(C_Rate~., data = Data_Gap_Train)
```

```
## # weights:  40 (28 variable)
## initial  value 1126.606539
## iter  10 value 1079.962845
```

```
## iter  20 value 864.260205
## iter  30 value 770.228815
## iter  40 value 716.371036
## iter  50 value 705.530032
## iter  60 value 702.512986
## iter  70 value 702.492899
## iter  80 value 702.238433
## final  value 702.228725
## converged
```

```
summary(model_gap_1)
```

```
## Call:
## multinom(formula = C_Rate ~ ., data = Data_Gap_Train)
##
## Coefficients:
##            (Intercept)  Inflation Recreation    Com_House   Child_Care
## Low           3.150807  -2.002632 -0.5985208 -0.002496771 -0.011897973
## Medium        2.720133  -1.497873 -0.5127378 -0.002773046 -0.004998913
## Very High -1216.077553 512.015922  1.1156268 -0.020437910  0.022570692
## Very Low      1.813895  -1.566388 -0.6712453 -0.033266857 -0.003974328
##                     Pop    Emp_Res
## Low       5.838618e-05 -0.5978852
## Medium    3.072436e-05 -0.2388855
## Very High 4.458361e-03 -1.1500153
## Very Low  5.497935e-05 -0.8552482
##
## Std. Errors:
##             (Intercept)    Inflation   Recreation    Com_House   Child_Care
## Low       5.030706e-05 9.001385e-05 6.409508e-05 4.757071e-03 0.0068568324
## Medium    6.224132e-05 1.185701e-04 8.166292e-05 4.087756e-03 0.0060243150
## Very High 4.885472e-07 1.140393e-06 4.209951e-06 7.361959e-05 0.0002965789
## Very Low  9.513441e-05 1.739427e-04 2.248637e-04 1.536962e-02 0.0085714111
##                     Pop      Emp_Res
## Low       7.196787e-06 4.801052e-05
## Medium    6.592175e-06 1.279104e-04
## Very High 5.233246e-05 5.374379e-06
## Very Low  1.027359e-05 1.083508e-04
##
## Residual Deviance: 1404.457
## AIC: 1460.457
```

```
exp(coef(model_gap_1))
```

```
##            (Intercept)     Inflation Recreation Com_House Child_Care      Pop
## Low          23.354900  1.349795e-01  0.5496241 0.9975063  0.9881725 1.000058
## Medium       15.182334  2.236052e-01  0.5988538 0.9972308  0.9950136 1.000031
## Very High     0.000000 2.321076e+222  3.0514802 0.9797695  1.0228273 1.004468
## Very Low      6.134294  2.087981e-01  0.5110717 0.9672804  0.9960336 1.000055
##               Emp_Res
## Low       0.5499735
## Medium    0.7875050
## Very High 0.3166319
## Very Low  0.4251776
```

These are the probabilities of neighbourhoods being having a particular crime rate level

```
head(round(fitted(model_gap_1),3))
```

```
##    High   Low Medium Very High Very Low
## 1 0.265 0.260  0.325         0    0.151
## 2 0.239 0.295  0.355         0    0.112
## 3 0.334 0.216  0.354         0    0.096
## 4 0.305 0.303  0.351         0    0.040
## 5 0.227 0.253  0.316         0    0.204
## 6 0.176 0.375  0.318         0    0.131
```

We now want to see what the accuracy of the model is.

```
Data_Gap_Train$C_RatePred <- predict(model_gap_1, newdata = Data_Gap_Train, 'class')
tab_gap <- table(Data_Gap_Train$C_Rate, Data_Gap_Train$C_RatePred)
tab_gap
```

```
##
##             High Low Medium Very High Very Low
##   High       104  16     54         3        0
##   Low         21  77     64         0        0
##   Medium      47  57     88         0        0
##   Very High    1   0      0       111        0
##   Very Low     5  26     26         0        0
```

Now we calculate accuracy

```
round((sum(diag(tab_gap))/sum(tab_gap))*100,2)
```

```
## [1] 54.29
```

Our accuracy for this model is slightly better at 52.93

We now predict on the test dataset. We cannot make a classification table as we have no actual values to compare the predicted values to.

```
head(predict(model_gap_1, newdata = Data_Gap_Test, "class"))
```

```
## [1] Medium High   High   High   High   High
## Levels: High Low Medium Very High Very Low
```

Our values for accuracy are fairly low. This is likely to be primarily due to our lack of data points but can also be a result of class imbalance in the dataset (This would be tough to fix as we cannot change the number of high/low/etc level crime rate neighbourhoods). It is also likely that our inability to source demographic information for each data point is a large contributing factor. While we can see that the features we have chosen here are relevant to our response variable, it can be hypothesised that demographic information is a key element in predicting the crime rates of neighbourhoods, especially features that are poverty indicators such as average household size, income, unemployment rate, etc. Without these, our data is lacking important context.

Feature Selection Using Decision Tree/Random Forest

First we will do this for the regular crime data (no three year gap between features and crime rate) and then we will do it for the three year gap data.

Note: Here C_Rate values are 0 = very low, 1 = low, 2 = medium, 3 = high and 4 = very high

In [1]:

```
import pandas as pd
import time
```

In [2]:

```
Data_1 = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/SMOTE_data.csv')
```

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [4]:

```
start = time.process_time()
```

In [5]:

```
Data_1.columns.values
```

Out[5]:

```
array(['Unnamed: 0', 'Year', 'N_ID', 'Pop', 'Ad_Ed', 'Child_Care',
       'Com_House', 'Emp_Res', 'Sub_Trt', 'Trans_House', 'Recreation',
       'Inflation', 'NIA', 'C_Rate'], dtype=object)
```

In [6]:

```
Data_1 = Data_1.drop('Unnamed: 0',axis = 1)
X_train, X_test, y_train, y_test = train_test_split(Data_1.drop('C_Rate',axis=1), Data_1
['C_Rate'], test_size=.3,random_state=22)
```

In [7]:

```
X_train.shape, X_test.shape
```

Out[7]:

```
((1176, 12), (504, 12))
```

In [8]:

```
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, mi
n_samples_leaf=5)
```

In [9]:

```
model.fit(X_train, y_train)
```

Out[9]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [10]:

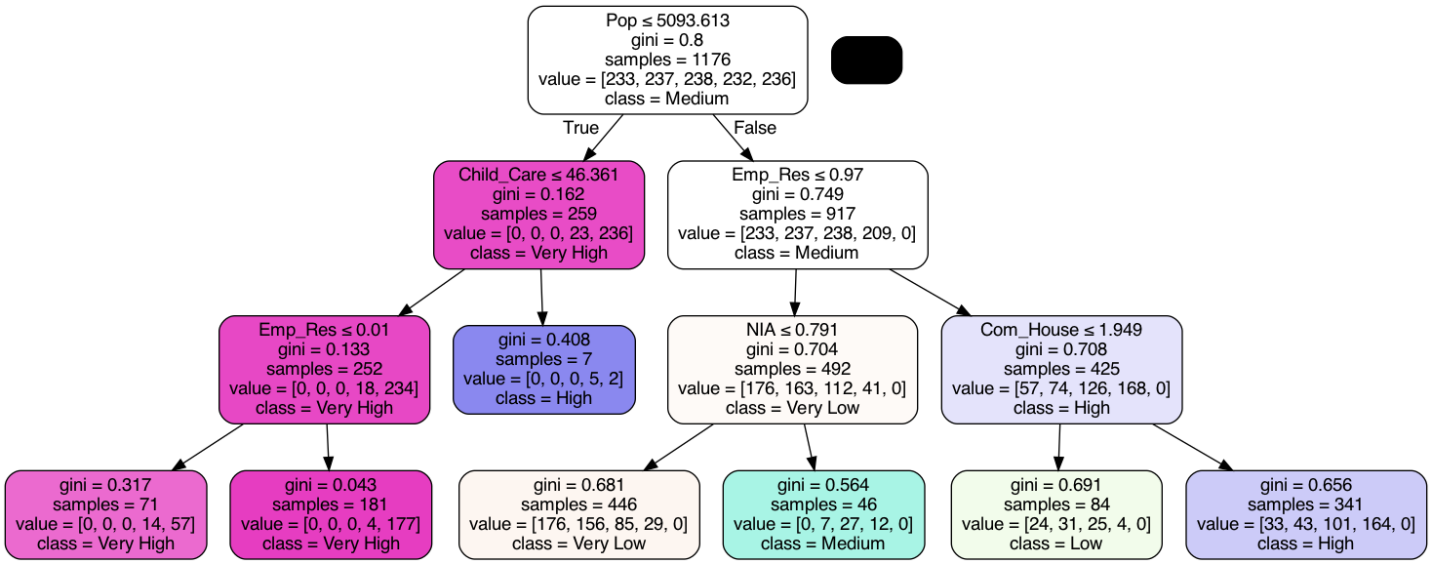
```
import io
```

```
from io import StringIO
from sklearn.tree import export_graphviz
# from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
import graphviz
from sklearn.metrics import classification_report, accuracy_score
from pydot import graph_from_dot_data
```

In [11]:

```
xvar = Data_1.drop('C_Rate', axis = 1)
feature_cols = xvar.columns
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True, special_chara
cters = True, feature_names = feature_cols, class_names = ['Very Low','Low','Medium','Hi
gh','Very High'])
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[11]:



In [12]:

```
#runtime for model
t1 = time.process_time() - start
print(t1)
```

```
0.3982890000000001
```

Population appears to be the most important feature here

In [13]:

```
predictions = model.predict(X_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model.predict(X_train)))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

```
Decision Tree Train Accuracy: 0.5416666666666666
Decision Tree Test Accuracy: 0.5535714285714286
```

In [14]:

```
print(classification_report(y_test,predictions,zero_division=0))
```

```
              precision    recall  f1-score   support

           0       0.41      0.83      0.55       103
           1       0.45      0.15      0.23        99
           2       0.50      0.08      0.14        98
           3       0.51      0.69      0.59       104
           4       0.93      0.99      0.96       100
```

```
    accuracy                           0.55       504
   macro avg       0.56      0.55      0.49       504
weighted avg       0.56      0.55      0.50       504
```

Our values here are much higher than our values using the original data

We will now use permutation feature importance to further evaluate the importance of our features

In [15]:

```
model.score(X_test,y_test)
```

Out[15]:

```
0.5535714285714286
```

In [16]:

```
from sklearn.inspection import permutation_importance
```

running permutation importance on our test set:

In [17]:

```
r = permutation_importance(model, X_test, y_test, n_repeats = 30, random_state = 0)
```

In [18]:

```
for i in r.importances_mean.argsort()[::-1]:
    if r.importances_mean[i] - 2 * r.importances_std[i] > 0:
        print(f"{list(Data_1.columns.values)[i]:<8}"
              f"{r.importances_mean[i]:.3f}"
              f" +/- {r.importances_std[i]:.3f}")
```

```
Pop     0.228 +/- 0.011
Emp_Res 0.096 +/- 0.016
Com_House0.053 +/- 0.008
NIA     0.025 +/- 0.010
Child_Care0.011 +/- 0.004
```

running it on our train set:

In [19]:

```
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [20]:

```
#runtime for model
t2 = time.process_time() -t1
print(t2)
```

```
4.335967
```

In [21]:

```
for i in r_1.importances_mean.argsort()[::-1]:
    if r_1.importances_mean[i] - 2 * r_1.importances_std[i] > 0:
        print(f"{list(Data_1.columns.values)[i]:<8}"
              f"{r_1.importances_mean[i]:.3f}"
              f" +/- {r_1.importances_std[i]:.3f}")
```

```
Pop     0.231 +/- 0.011
Emp_Res 0.087 +/- 0.009
Com_House0.049 +/- 0.005
NIA     0.033 +/- 0.005
```

```
Child_Care0.011 +/- 0.003
```

Lets see how our model score changes keeping only the most "important" features. We will start by only keeping pop,emp_res,and com_house

In [22]:

```
train_1 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Sub_Trt', 'Trans_House','Recreation','Inflation','NIA','Child_Care'], axis = 1)
test_1 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Sub_Trt', 'Trans_House','Recreation','Inflation','NIA','Child_Care'], axis = 1)
```

In [23]:

```
Data_1.columns.values
```

Out[23]:

```
array(['Year', 'N_ID', 'Pop', 'Ad_Ed', 'Child_Care', 'Com_House',
       'Emp_Res', 'Sub_Trt', 'Trans_House', 'Recreation', 'Inflation',
       'NIA', 'C_Rate'], dtype=object)
```

In [24]:

```
model_1 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3, min_samples_leaf=5)
```

In [25]:

```
model_1.fit(train_1, y_train)
```

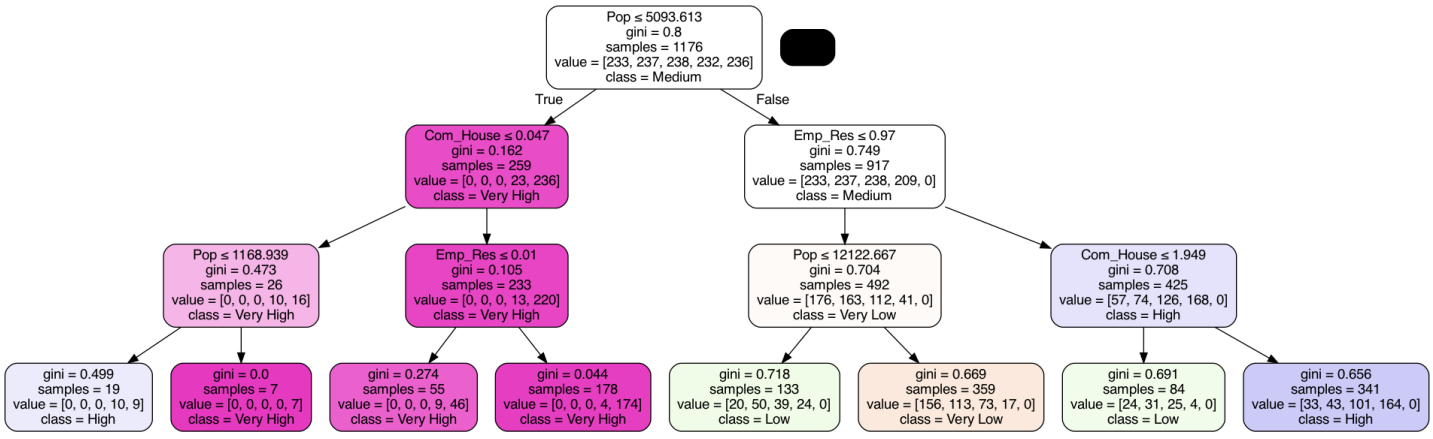
Out[25]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [26]:

```
xvar_1 = Data_1.drop(['Year', 'N_ID', 'Ad_Ed','Child_Care','Sub_Trt', 'Trans_House','Recreation','Inflation','NIA','C_Rate'], axis = 1)
feature_cols_1 = xvar_1.columns
dot_data_1 = StringIO()
export_graphviz(model_1, out_file = dot_data_1, filled = True, rounded = True, special_characters = True, feature_names = feature_cols_1, class_names = ['Very Low','Low','Medium','High','Very High'])
(graph_1,)= graph_from_dot_data(dot_data_1.getvalue())
Image(graph_1.create_png())
```

Out[26]:



In [27]:

```
predictions = model_1.predict(test_1)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_1.predict(train_1)))
```

```
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_1.predict(test_1)))
```

```
Decision Tree Train Accuracy: 0.5425170068027211
Decision Tree Test Accuracy: 0.5555555555555556
```

In [28]:

```
#runtime for model
t3 = time.process_time() - t2
print(t3)
```

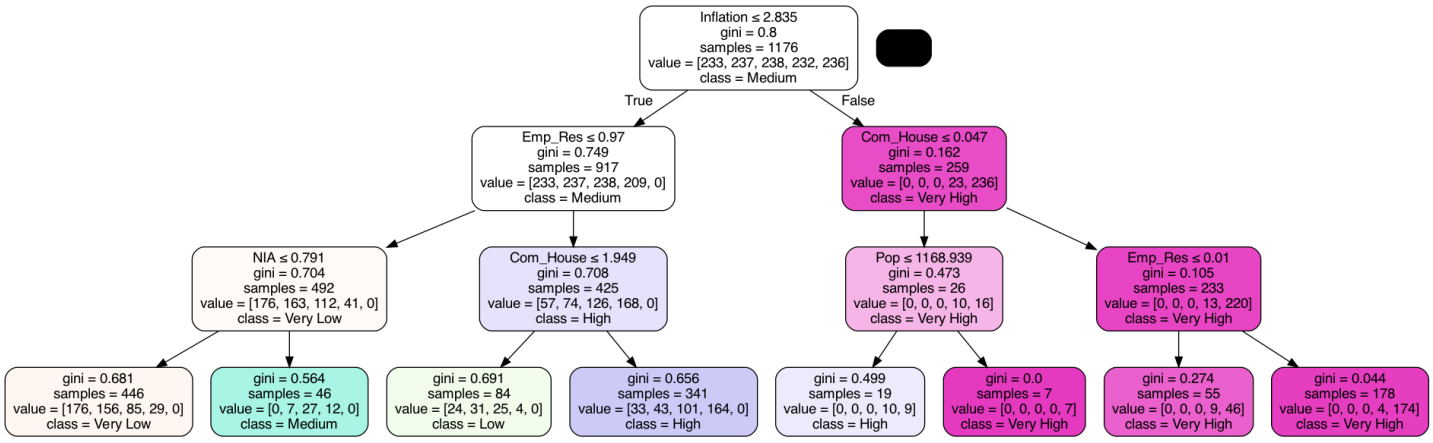
```
0.6057559999999995
```

we will add in NIA

In [29]:

```
train_2 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Child_Care','Sub_Trt', 'Trans_House','R
ecreation'], axis = 1)
test_2 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Child_Care','Sub_Trt', 'Trans_House','Rec
reation'], axis = 1)

model_2 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,
min_samples_leaf=5)

model_2.fit(train_2, y_train)

xvar_2 = Data_1.drop(['Year', 'N_ID', 'Ad_Ed','Child_Care','Sub_Trt', 'Trans_House','Rec
reation','C_Rate'], axis = 1)
feature_cols_2 = xvar_2.columns

dot_data_2 = StringIO()
export_graphviz(model_2, out_file = dot_data_2, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_2, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

Out[29]:



In [30]:

```
predictions = model_2.predict(test_2)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_2.predict(train_2)
))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_2.predict(test_2)))
```

```
Decision Tree Train Accuracy: 0.5399659863945578
Decision Tree Test Accuracy: 0.5476190476190477
```

In [31]:

```
#runtime for model
t4 = time.process_time() - t3
print(t4)
```

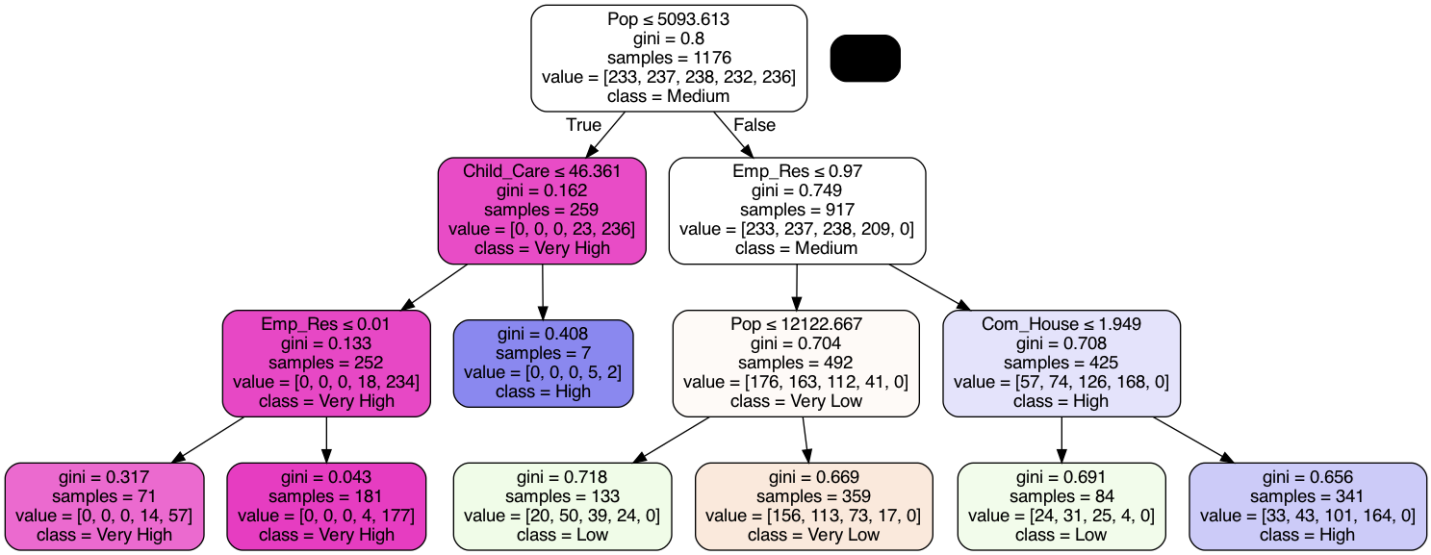
```
4.486612
```

Our Accuracy reduced, so we will remove NIA and add in Child Care

In [32]:

```
train_3 = X_train.drop(['Year', 'N_ID', 'Inflation','Sub_Trt', 'Trans_House','Recreation
','NIA'], axis = 1)
test_3 = X_test.drop(['Year', 'N_ID', 'Inflation','Sub_Trt', 'Trans_House','Recreation',
'NIA'], axis = 1)

model_3 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,
min_samples_leaf=5)

model_3.fit(train_3, y_train)

xvar_3 = Data_1.drop(['Year', 'N_ID', 'Inflation','Sub_Trt', 'Trans_House','Recreation',
'NIA','C_Rate'], axis = 1)
feature_cols_3 = xvar_3.columns
dot_data_3 = StringIO()
export_graphviz(model_3, out_file = dot_data_3, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_3, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_3,)= graph_from_dot_data(dot_data_3.getvalue())
Image(graph_3.create_png())
```

Out[32]:



In [33]:

```
predictions = model_3.predict(test_3)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_3.predict(train_3)
))
print ("Decision Tree Test Accuracy:", accuracy_score(y_test, model_3.predict(test_3)))
```

```
Decision Tree Train Accuracy: 0.54421768707483
Decision Tree Test Accuracy: 0.5615079365079365
```

In [34]:

```
#runtime for model
t5 = time.process_time() - t4
print(t5)
```

```
0.7906279999999999
```

According to our results, Com_House, Child_Care, Emp_Res and Pop appear to be the most importnat features for this model

Feature Selection Using Decision Tree/Random Forest

We will now conduct feature selection using decision trees on the data where there is a three year gap between feature values and crime rate.

In [1]:

```
import pandas as pd
import time
```

In [2]:

```
Train_Data = pd.read_csv('//Users/alyzehjiwani/Downloads/Data/Final Data Used/Year_Gap_SM
OTE_Train_data.csv')
Test_Data = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Ye
ar_Gap_Test.csv')
```

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [4]:

```
start = time.process_time()
```

In [5]:

```
Train_Data = Train_Data.drop('Unnamed: 0',axis = 1)
Test_Data = Test_Data.drop('Unnamed: 0', axis = 1)
X_train = Train_Data.loc[:,Train_Data.columns!='C_Rate']
X_test = Test_Data
y_train = Train_Data['C_Rate']
```

In [6]:

```
X_train.shape, X_test.shape, y_train.shape
```

Out[6]:

```
((960, 12), (420, 12), (960,))
```

In [7]:

```
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, mi
n_samples_leaf=5)
```

In [8]:

```
model.fit(X_train, y_train)
```

Out[8]:

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [9]:

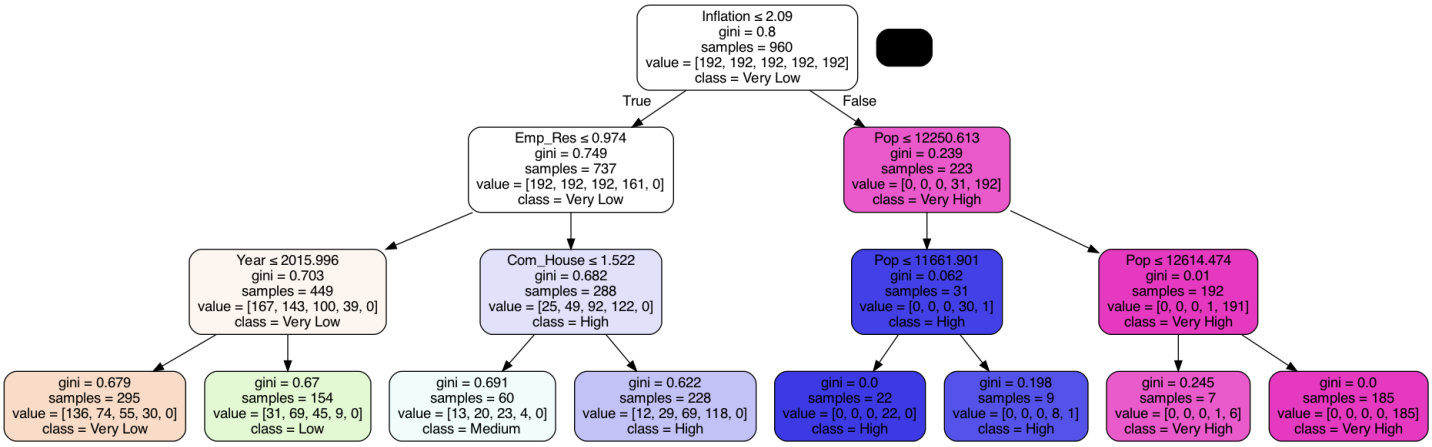
```
import io
from io import StringIO
from sklearn.tree import export_graphviz
# from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
import graphviz
```

```
from sklearn.metrics import classification_report, accuracy_score
from pydot import graph_from_dot_data
```

In [10]:

```
xvar = Train_Data.drop('C_Rate', axis = 1)
feature_cols = xvar.columns
dot_data = StringIO()
export_graphviz(model, out_file = dot_data, filled = True, rounded = True, special_chara
cters = True, feature_names = feature_cols, class_names = ['Very Low','Low','Medium','Hi
gh','Very High'])
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[10]:



In [11]:

```
predictions = model.predict(X_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model.predict(X_train)))
```

```
Decision Tree Train Accuracy: 0.590625
```

In [12]:

```
#runtime for model
t1 = time.process_time() - start
print(t1)
```

```
0.3914620000000002
```

Again, our accuracy improved when using the data containing synthesised data. We will now use permutation feature importance to further evaluate the importance of our features

In [13]:

```
from sklearn.inspection import permutation_importance
```

We will run it on our train set as we dont have response values for our test set

In [14]:

```
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [15]:

```
for i in r_1.importances_mean.argsort()[::-1]:
    if r_1.importances_mean[i] - 2 * r_1.importances_std[i] > 0:
        print(f"{list(Train_Data.columns.values)[i]:<8}"
              f"{r_1.importances_mean[i]:.3f}"
              f" +/- {r_1.importances_std[i]:.3f}")
```

```
Inflation0.250 +/- 0.010
Emp_Res 0.119 +/- 0.009
Pop     0.067 +/- 0.006
```

```
Year     0.058 +/- 0.008
Com_House0.029 +/- 0.007
```

Lets see if dropping the "unneccessary" columns improves our model.

In [16]:

```
Train_Data.columns.values
```

Out[16]:

```
array(['Year', 'N_ID', 'Pop', 'Ad_Ed', 'Child_Care', 'Com_House',
       'Emp_Res', 'Sub_Trt', 'Trans_House', 'Recreation', 'Inflation',
       'NIA', 'C_Rate'], dtype=object)
```

In [17]:

```
Test_Data.head()
```

Out[17]:

	Year	N_ID	Pop	Ad_Ed	Child_Care	Com_House	Emp_Res	Sub_Trt	Trans_House	Recreation	Inflation	NIA
0	**2019**	**97**	**13789.999686**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1.95**	**0**
1	**2019**	**27**	**29846.996720**	**1**	**0**	**4**	**8**	**2**	**0**	**1**	**1.95**	**1**
2	**2019**	**38**	**17784.999438**	**3**	**0**	**0**	**3**	**0**	**0**	**0**	**1.95**	**0**
3	**2019**	**31**	**16971.000148**	**1**	**0**	**48**	**4**	**0**	**0**	**0**	**1.95**	**0**
4	**2019**	**16**	**27007.001187**	**0**	**46**	**0**	**1**	**0**	**0**	**1**	**1.95**	**0**

In [18]:

```
new_train = X_train.drop(['N_ID','Ad_Ed', 'Child_Care','Sub_Trt', 'Trans_House', 'Recrea
tion', 'NIA'], axis = 1)
new_test = Test_Data.drop(['N_ID',  'Ad_Ed', 'Child_Care', 'Sub_Trt', 'Trans_House', 'Re
creation', 'NIA'], axis = 1)
```

In [19]:

```
new_train.shape, y_train.shape, new_test.shape
```

Out[19]:

```
((960, 5), (960,), (420, 5))
```

In [20]:

```
model_2 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
min_samples_leaf=5)
```

In [21]:

```
model_2.fit(new_train, y_train)
```

Out[21]:

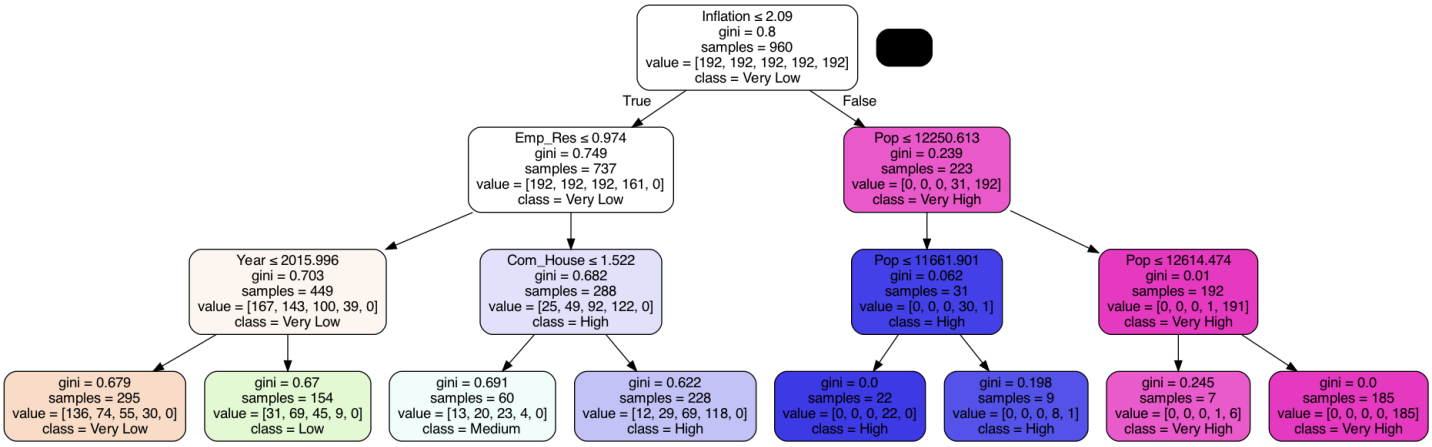
```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [22]:

```
new_var = new_train
new_feature_cols = new_var.columns
new_dot_data = StringIO()
export_graphviz(model_2, out_file = new_dot_data, filled = True, rounded = True, special
_characters = True, feature_names = new_feature_cols, class_names = ['Very Low','Low','M
edium','High','Very High'])
(graph_1,)= graph_from_dot_data(new_dot_data.getvalue())
```

```
Image(graph_1.create_png())
```

Out[22]:



In [23]:

```
predictions = model_2.predict(new_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_2.predict(new_trai
n)))
```

```
Decision Tree Train Accuracy: 0.590625
```

In [24]:

```
#runtime for model
t2 = time.process_time() - t1
print(t2)
```

```
3.5278009999999997
```

Our accuracy for our model on our train set remained the same. Let's see if dropping Com_House changes anything.

In [25]:

```
train_2 = X_train.drop(['N_ID','Ad_Ed', 'Child_Care','Sub_Trt', 'Trans_House', 'Recreati
on', 'NIA','Com_House'], axis = 1)
test_2 = Test_Data.drop(['N_ID','Ad_Ed', 'Child_Care','Sub_Trt', 'Trans_House', 'Recreat
ion', 'NIA','Com_House'], axis = 1)
```

In [26]:

```
model_3 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
min_samples_leaf=5)
```

In [27]:

```
model_3.fit(train_2, y_train)
```

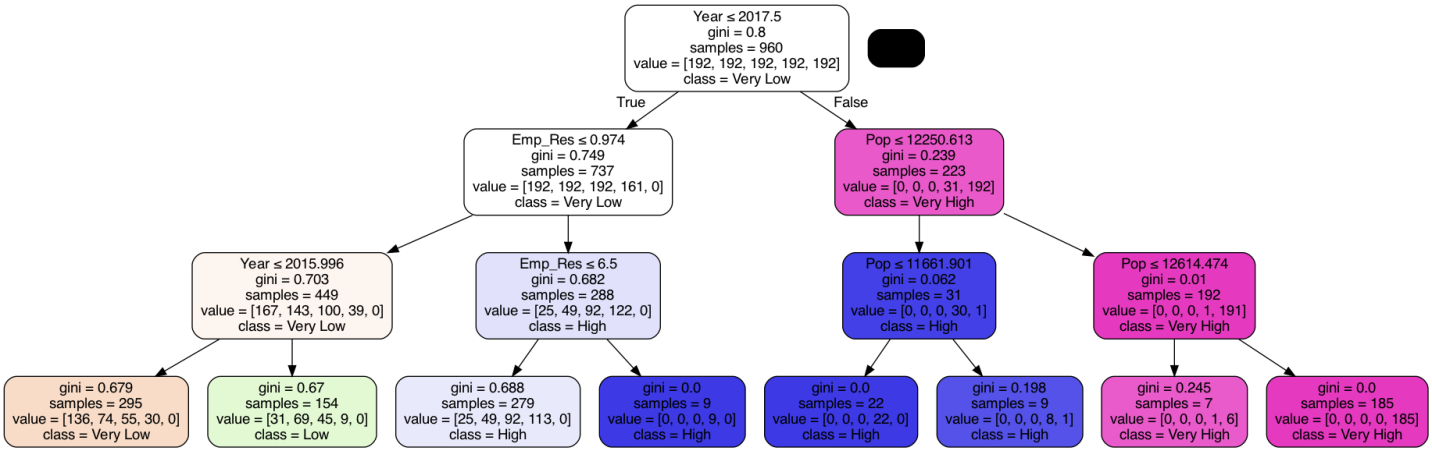
Out[27]:

```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [28]:

```
var_2 = train_2
feature_cols_2 = var_2.columns
dot_data_2 = StringIO()
export_graphviz(model_3, out_file = dot_data_2, filled = True, rounded = True, special_c
haracters = True, feature_names = feature_cols_2, class_names = ['Very Low','Low','Mediu
m','High','Very High'])
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

Out[28]:



In [29]:

```
predictions = model_3.predict(test_2)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model_3.predict(train_2)
))
```

```
Decision Tree Train Accuracy: 0.5708333333333333
```

In [30]:

```
#runtime for model
t3 = time.process_time() - t2
print(t3)
```

```
0.554052
```

Our Accuracy reduced, so we will keep the features Inflation, Year, Emp_Res, Pop and Com_House.

In [ ]:

```

```

SMOTE Multinomial Model Fitting

Alyzeh Jiwani

25/07/2022

We will now repeat the entire process of post feature selection and multinomial model fitting with our new, larger, datasets containing our synthesised data. From our process of feature selection that we did in our Python Notebook, we already saw a slight improvement in our decsion tree model accuracy scores. Our hope is that we will see an improvement in the effectiveness of our multinomial model as well.

```
SData_Current <- read.csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/SMOTE_data.csv')
SData_Gap_Train <- read.csv('//Users/alyzehjiwani/Downloads/Data/Final Data Used/Year_Gap_SMOTE_Train_da
SData_Gap_Test <- read.csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Year_Gap_Test
```

Post Feature Selection

From our analysis using decision trees and permutation fetaure importance in python, we are able to now remove some features that are unlikely to contribute to our to our models.

For SMOTE_data, where the values for a ll the features and our response variable crime_rate are taken in the same year, we decided to keep the following features: Com_House, Child_Care, Emp_Res and Pop

For df_2, where values for features are taken three years prior to values for our response variable crime_rate, the features we have decided to keep are: Inflation, Year, Emp_Res, Pop and Com_House

```
SData_Current <- SData_Current[,c('Com_House','Child_Care','Pop','Emp_Res','C_Rate')]
SData_Gap_Train <- SData_Gap_Train[,c('Inflation', 'Year', 'Com_House', 'Pop','Emp_Res', 'C_Rate')]
SData_Gap_Test <-SData_Gap_Test[,c('Inflation', 'Year', 'Com_House', 'Pop','Emp_Res')]
```

We will now try to fit the data to a multinomial regression model

```
library(nnet)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

Working with SData_Current:

Splitting into Train/Test

```
index <- createDataPartition(SData_Current$C_Rate, p = 0.7, list = FALSE)
train <- SData_Current[index,]
test <- SData_Current[-index,]
```

```
model_Scur_1 <- multinom(C_Rate~., data = SData_Current)
```

```
## # weights:  30 (20 variable)
## initial  value 2703.855693
## iter  10 value 2306.648738
## iter  20 value 1932.199874
## iter  30 value 1813.401523
## iter  40 value 1810.620305
## final  value 1810.602419
## converged
```

```
summary(model_Scur_1)
```

```
## Call:
## multinom(formula = C_Rate ~ ., data = SData_Current)
##
## Coefficients:
##   (Intercept)  Com_House   Child_Care           Pop   Emp_Res
## 1  0.03543474 0.04476201  0.004588498 -2.449197e-05 0.1333186
## 2  0.04703063 0.04696612  0.016648530 -4.644601e-05 0.4647420
## 3  0.21698055 0.05322401  0.010265796 -8.225881e-05 0.7917630
## 4  4.43596237 0.08196770 -0.020493060 -1.165475e-03 1.3111520
##
## Std. Errors:
##    (Intercept)   Com_House  Child_Care          Pop      Emp_Res
## 1 4.870650e-05 0.007765118 0.005213027 4.482774e-06 9.722959e-05
## 2 4.710003e-05 0.007748095 0.004906921 4.700298e-06 1.122569e-04
## 3 1.039807e-04 0.007765348 0.005324381 5.053545e-06 2.214359e-04
## 4 2.098251e-04 0.018796385 0.010934331 1.276946e-04 3.343709e-04
##
## Residual Deviance: 3621.205
## AIC: 3661.205
```

```
exp(coef(model_Scur_1))
```

```
##   (Intercept) Com_House Child_Care       Pop  Emp_Res
## 1    1.036070  1.045779  1.0045990 0.9999755 1.142614
## 2    1.048154  1.048086  1.0167879 0.9999536 1.591603
## 3    1.242320  1.054666  1.0103187 0.9999177 2.207284
## 4   84.433342  1.085421  0.9797155 0.9988352 3.710446
```

These are the probabilities of neighbourhoods being having a particular crime rate level

```
head(round(fitted(model_Scur_1),3))
```

```
##       0     1     2     3 4
## 1 0.344 0.271 0.214 0.170 0
## 2 0.359 0.272 0.221 0.149 0
## 3 0.283 0.236 0.243 0.239 0
## 4 0.056 0.355 0.297 0.293 0
## 5 0.355 0.260 0.277 0.108 0
## 6 0.322 0.353 0.213 0.113 0
```

We now want to see what the accuracy of the model is.

```
train$C_RatePred <- predict(model_Scur_1, newdata = train, 'class')
tab <- table(train$C_Rate, train$C_RatePred)
tab
```

```
##
##       0   1   2   3   4
##   0 173  17  32  18   0
##   1 133  32  31  35   0
##   2  82  36  43  75   0
##   3  36  41  34 105  20
##   4   0   0   0   0 236
```

Now we calculate accuracy

```
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 49.96
```

Our accuracy is at a value of 50.21, which is higher than our original value of 42.75

We now predict on the test dataset and see our classification table

```
test$C_RatePred <- predict(model_Scur_1, newdata = test, "class")


tab_test <- table(test$C_Rate, test$C_RatePred)
tab_test
```

```
##
##       0   1   2   3   4
##   0  66  11  10   9   0
##   1  62  13   9  21   0
##   2  35  17  18  30   0
##   3  15   8  14  51  12
##   4   0   0   0   0 100
```

accuracy of test model

```
round((sum(diag(tab_test))/sum(tab_test))*100,2)
```

```
## [1] 49.5
```

Again, this is higher than our original value of 40.42

We now repeat this for the data where values of our features are taken three years prior to the values of our response variable.

```
model_Sgap_1 <- multinom(C_Rate~., data = SData_Gap_Train)
```

```
## # weights:  35 (24 variable)
## initial  value 1545.060396
## iter  10 value 1521.471813
## iter  20 value 1168.729640
## iter  30 value 1016.369746
## iter  40 value 985.509795
## iter  50 value 977.818090
## iter  60 value 976.798117
## iter  70 value 975.890379
## iter  80 value 975.808460
## iter  90 value 970.056251
## iter 100 value 969.947559
## final  value 969.947559
## stopped after 100 iterations
```

```
summary(model_Sgap_1)
```

```
## Call:
## multinom(formula = C_Rate ~ ., data = SData_Gap_Train)
##
## Coefficients:
##   (Intercept)   Inflation        Year  Com_House           Pop     Emp_Res
## 1  -321.52245  -0.2324984  0.15940493 0.05404951  1.105814e-05  0.03374738
## 2    71.64339   0.1946673 -0.03582643 0.05346558 -2.522182e-05  0.56206765
## 3  -531.60875   1.2953466  0.26255469 0.06066671 -4.563481e-05  0.84901705
## 4   -50.47378 591.6783241 -0.66903247 0.09143197  4.841996e-03 -0.01320817
##
## Std. Errors:
##    (Intercept)    Inflation         Year   Com_House          Pop      Emp_Res
## 1 6.840353e-08 2.508771e-05 0.0001220574 0.011236088 1.226962e-05 1.227861e-04
## 2 6.447748e-08 1.397687e-05 0.0001239996 0.011228947 1.292896e-05 1.524275e-04
## 3 6.728749e-08 6.201546e-06 0.0001298659 0.011232439 1.363454e-05 7.097931e-05
## 4 3.983029e-06 9.227320e-06 0.0080398183 0.002075954 1.331075e-03 3.522069e-05
##
## Residual Deviance: 1939.895
## AIC: 1987.895
```

```
exp(coef(model_Sgap_1))
```

```
##     (Intercept)     Inflation      Year Com_House       Pop   Emp_Res
## 1 2.315117e-140  7.925510e-01 1.1728128  1.055537 1.0000111 1.0343233
## 2  1.301156e+31  1.214907e+00 0.9648077  1.054921 0.9999748 1.7542960
## 3 1.334295e-231  3.652262e+00 1.3002476  1.062545 0.9999544 2.3373482
## 4  1.200929e-22 9.175532e+256 0.5122039  1.095742 1.0048537 0.9868787
```

These are the probabilities of neighbourhoods being having a particular crime rate level

```
head(round(fitted(model_Sgap_1),3))
```

```
##       0     1     2     3 4
## 1 0.396 0.178 0.260 0.167 0
## 2 0.301 0.203 0.299 0.197 0
```

```
## 3 0.294 0.141 0.313 0.252 0
## 4 0.045 0.277 0.363 0.315 0
## 5 0.445 0.227 0.218 0.111 0
## 6 0.266 0.313 0.274 0.148 0
```

We now want to see what the accuracy of the model is.

```
SData_Gap_Train$C_RatePred <- predict(model_Sgap_1, newdata = SData_Gap_Train, 'class')
tab_gap <- table(SData_Gap_Train$C_Rate, SData_Gap_Train$C_RatePred)
tab_gap
```

```
##
##       0   1   2   3   4
##   0 148  20  15   9   0
##   1 101  49  18  24   0
##   2  81  28  41  42   0
##   3  25  25  42  97   3
##   4   0   0   0   2 190
```

Now we calculate accuracy

```
round((sum(diag(tab_gap))/sum(tab_gap))*100,2)
```

```
## [1] 54.69
```

Our accuracy for this model is slightly better at 54.69

We now predict on the test dataset.

```
SData_Gap_Test$C_RatePred <- predict(model_Sgap_1, newdata = SData_Gap_Test, "class")

head(SData_Gap_Test$C_RatePred)
```

```
## [1] 3 3 3 3 3 3
## Levels: 0 1 2 3 4
```

Our values for accuracy have improved using the synthesized data.

Conclusion

Overall we can conclude that the final model in which we used synthesized data was the most effective at predicting the crime rates of different Toronto neighbourhoods, as it had the greatest accuracy score of 54.69. This model identified community housing access, inflation, employment resources, and population as the most relevant factors in predicting crime rates in the future. The model, although having the best accuracy score of all the models, still has a fairly low score, this is likely to be a result of lack of data. Perhaps the features we used in our analysis were unable to fully assist us in drawing conclusions about crime rates in different Toronto neighbourhoods, and more data on different features should be collected and assessed.

Throughout our project we encountered many struggles due to the nature of our data. Lack of accessibility to important contextual data and inconsistencies in records across time resulted in us not only having a much smaller dataset to work with, but also caused us to be unable to fully explore the research question at hand. Factors such as average household income, ambulance calls, demographics of communities, demographics of those committing crimes, etc were all inaccessible even though one would hypothesis that data on such elements would likely provide great insight into answering our research question.

Whilst having increased access to more contextual data would likely assist in improving the effectiveness of our models, data inconsistency over time would likely still cause major issues in the continuation of this study. As there are only a few neighbourhoods, there are a limited number of datapoints available per year. As such, even with access to more relevant and contextual attributes, our dataset would still be very small. Thus in order be able to model the data properly, disparities in the ways in which socioeconomic and demographic information are collected over time need to be fixed, so as to allow more accurate and robust analysis over a much greater period of time.

References

Government of Canada, Statistics Canada. (2022, April 20). Operating expenditures for adult correctional services. Retrieved May 16, 2022, from https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3510001301&pickMembers%5B0%5D=1.8&cubeTimeFrame.startYear=2016%2B%2F%2B2017&cubeTimeFrame.endYear=2020%2B%2F%2B2021&referencePeriods=20160101%2C20200101

Charron, M., Canadian Centre for Justice Statistics, Statistics Canada, & Depository Services Program (Canada). (2011). Neighbourhood Characteristics and the Distribution of Crime in Toronto [E-book]. Statistics Canada, Canadian Centre for Justice Statistics.

City of Toronto. (2020). Toronto Strong 2020. https://www.toronto.ca/legdocs/mmis/2017/cd/bgrd/backgroundfile-108051.pdf

5 Ways to Deal with the Lack of Data in Machine Learning. (n.d.). KDnuggets. https://www.kdnuggets.com/2019/06/5-ways-lack-data-machine-learning.html