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 = TRUI
```

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.

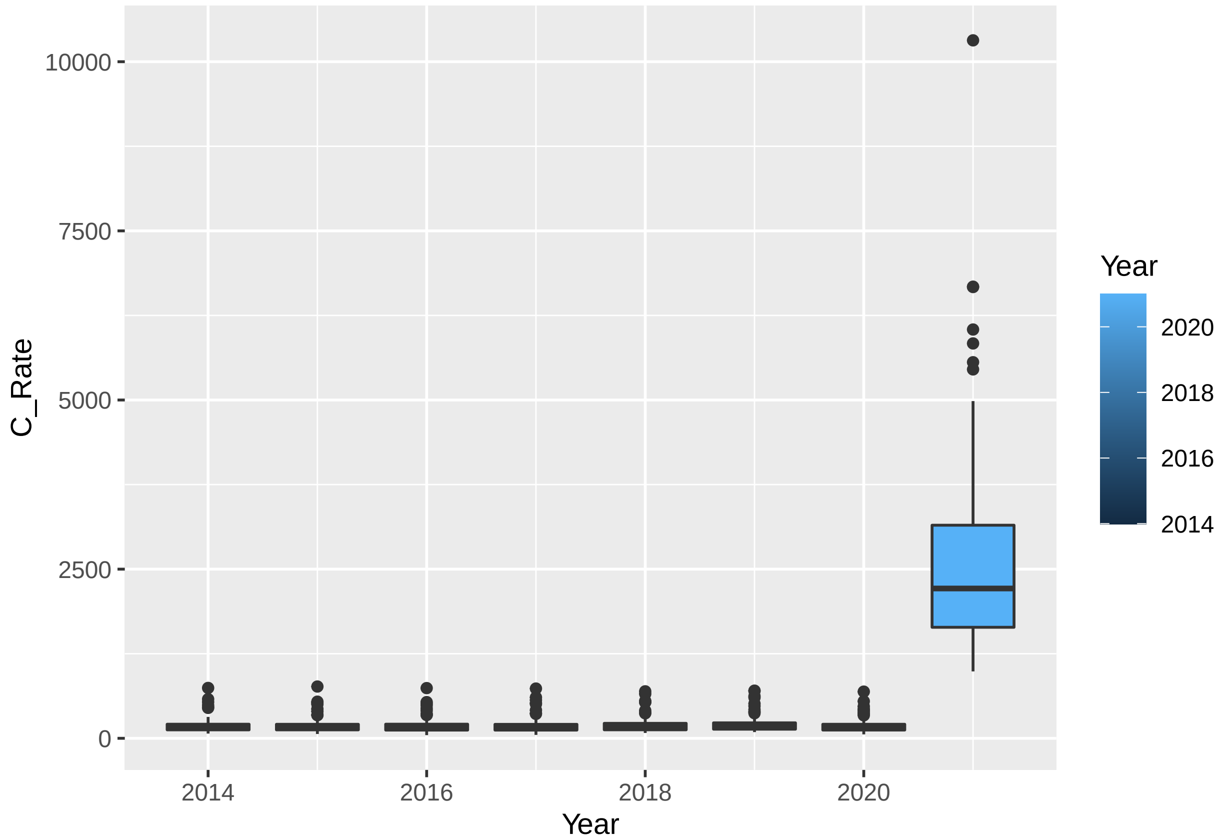
```r
library(ggplot2)
library(tidyverse)
```

```
## -- 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()
```

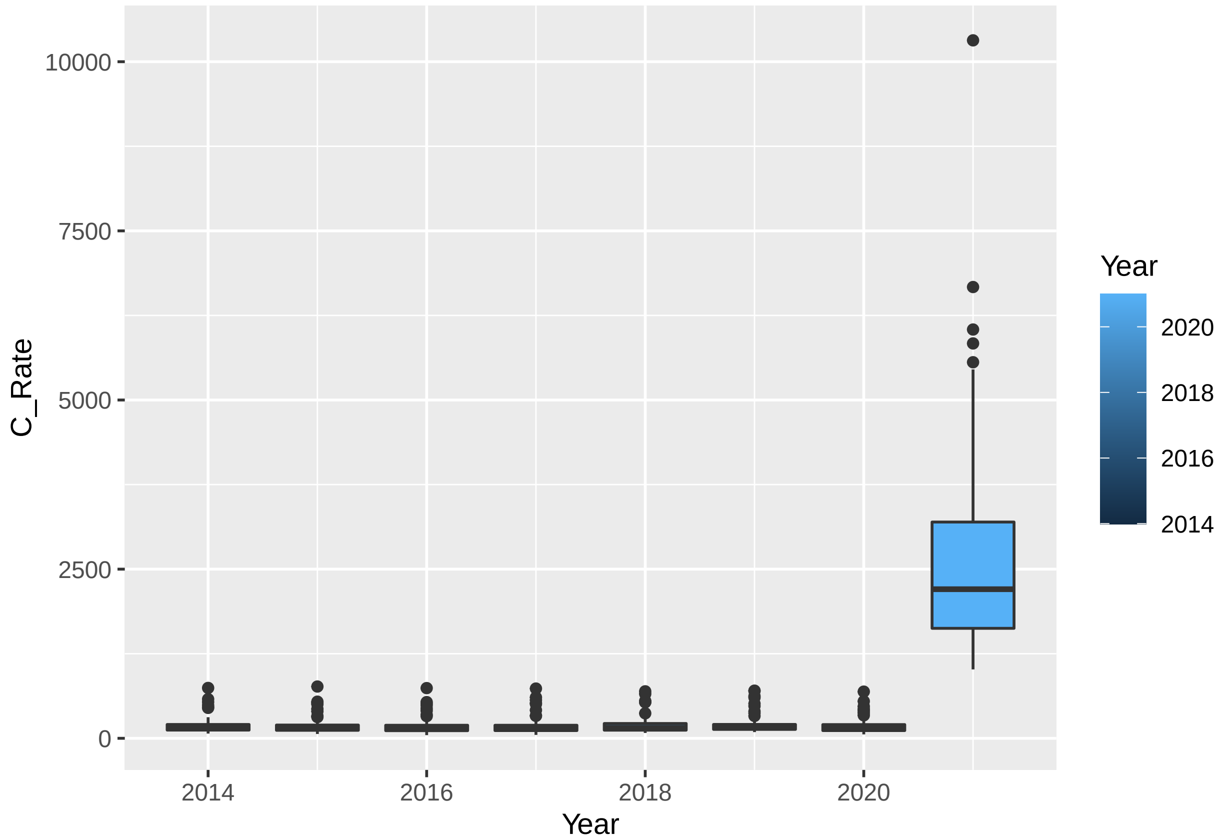
```r
bp1 <- ggplot(crime_data, aes(x=Year, y = C_Rate, group = Year))+
  geom_boxplot(aes(fill = Year))
bp1
```



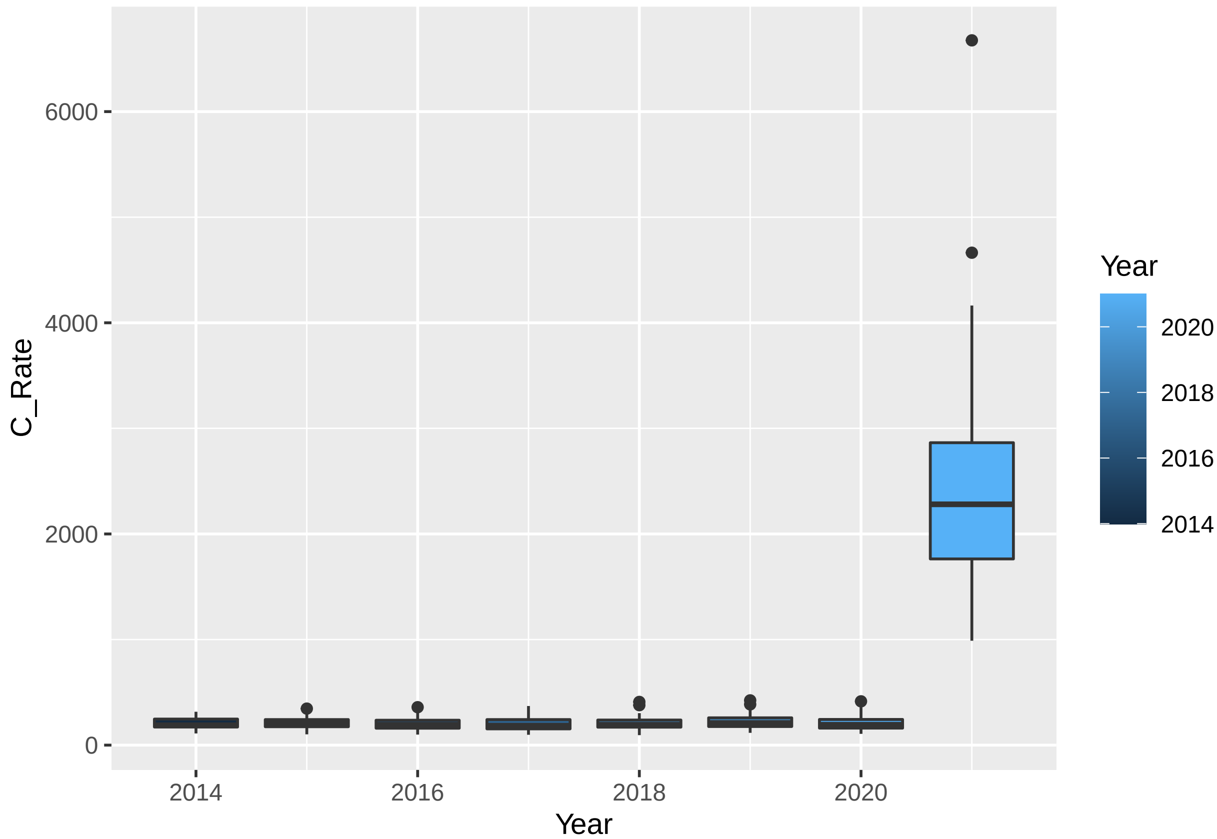
So looking at this we can see that the total crime rate inreased significantly in 2021.

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
```

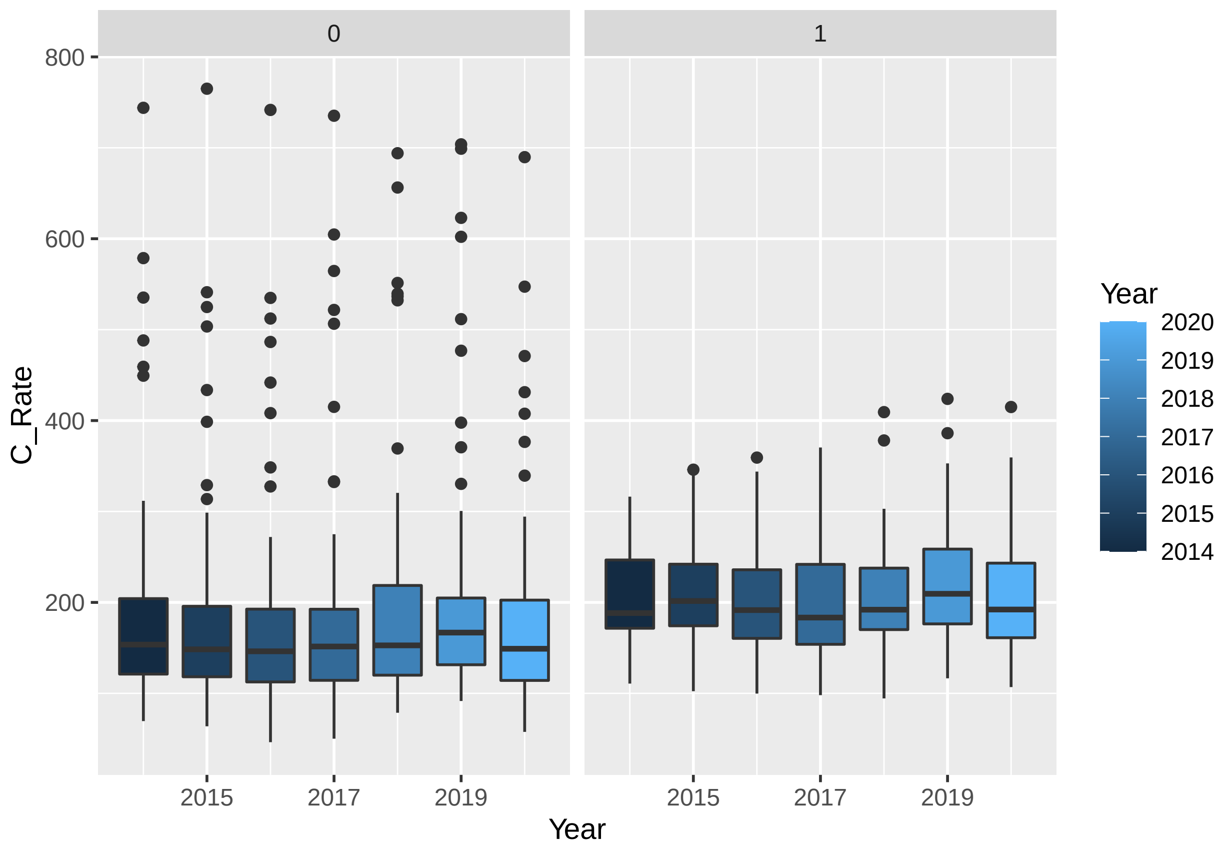



```
bp3 <- ggplot(crime_data[crime_data$NIA == 1,], aes(x= Year, y=C_Rate, group = Year))+
  geom_boxplot(aes(fill=Year))
bp3
```



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)
```

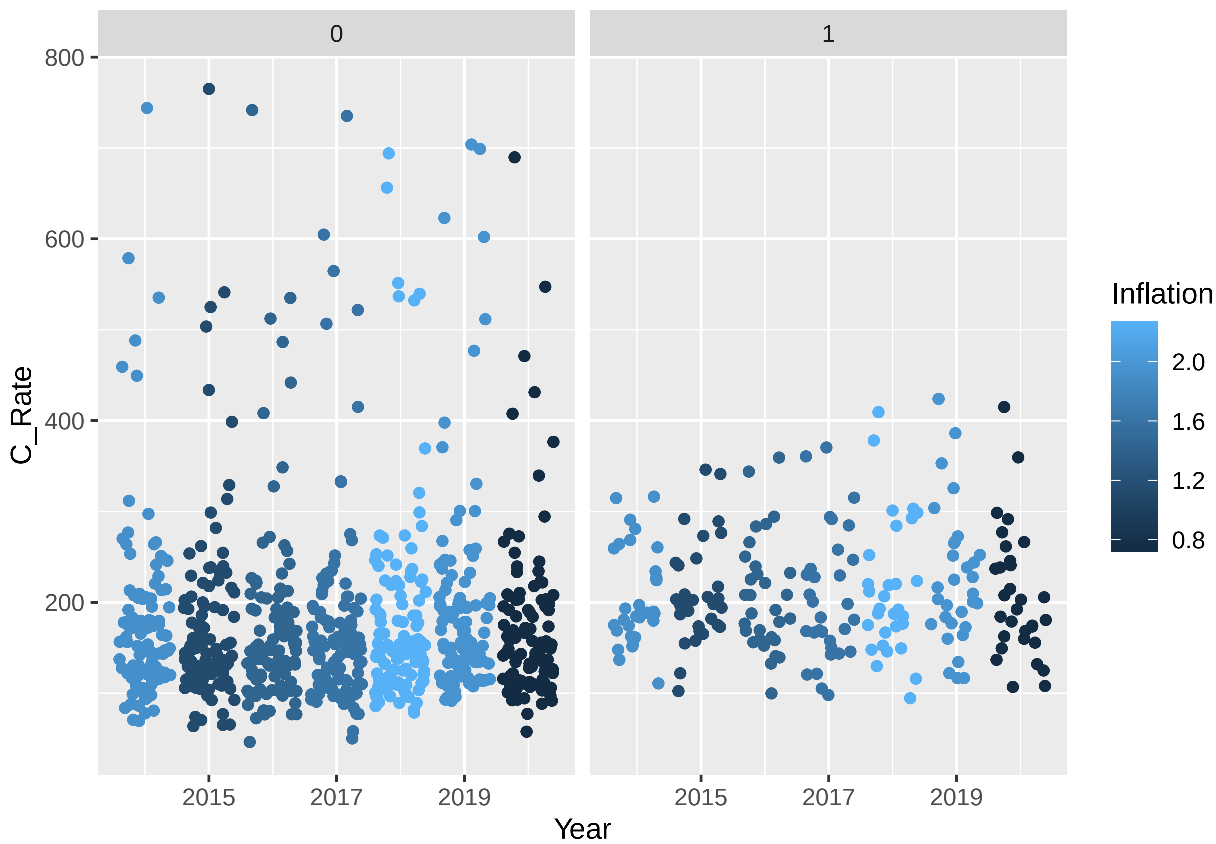
```
##   Year N_ID      Pop    C_Rate Ad_Ed Child_Care Com_House Emp_Res Sub_Trt
## 1 2014   97 11197.33  69.46098     0          0         0       0       0
## 2 2014   27 25528.89 314.67620     0          0         4       1       0
## 3 2014   38 14298.67 102.57366     0          0         0       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)
```

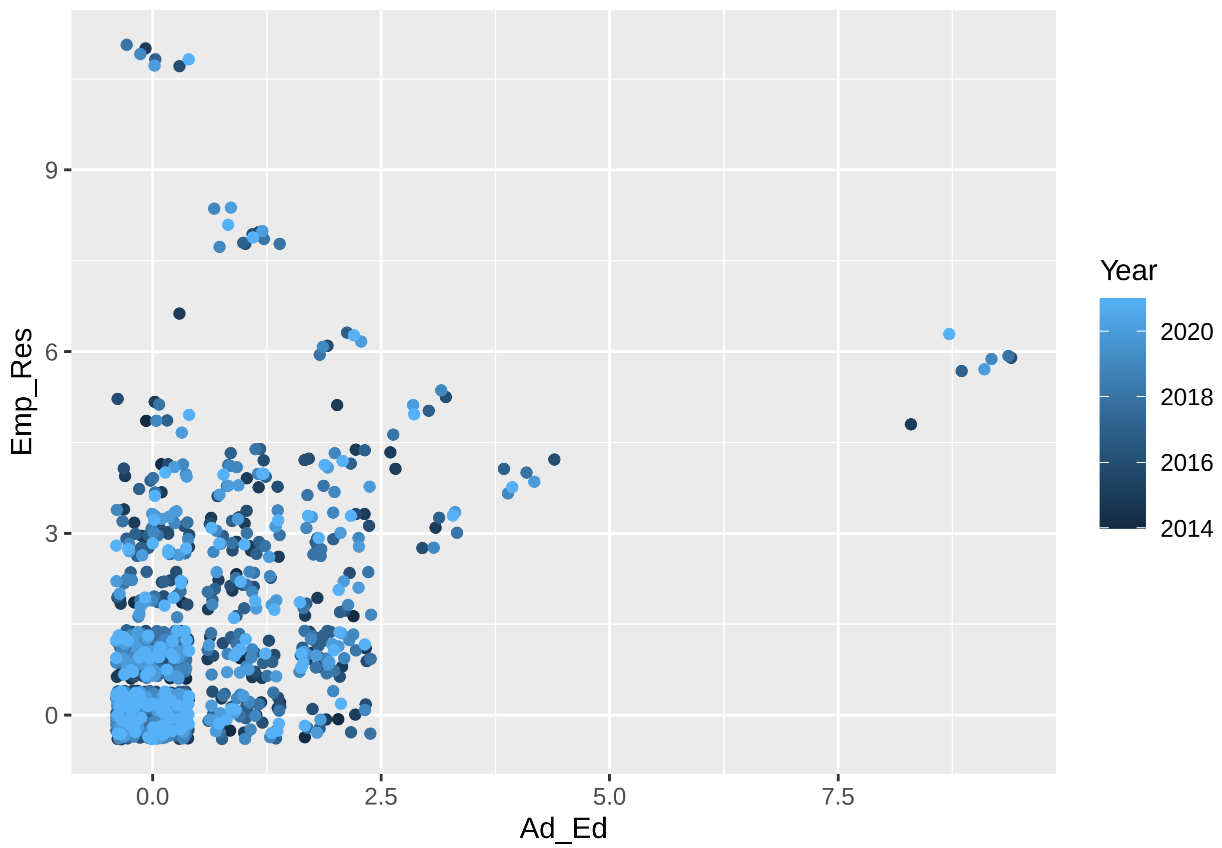
```
##                    Year        N_ID         Pop      C_Rate        Ad_Ed
## Year         1.000000000  0.00000000 -0.23255123  0.495081707  0.095126689
## N_ID         0.000000000  1.00000000  0.08695004  0.015884718  0.015812461
## Pop         -0.232551234  0.08695004  1.00000000 -0.442537501  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)
```



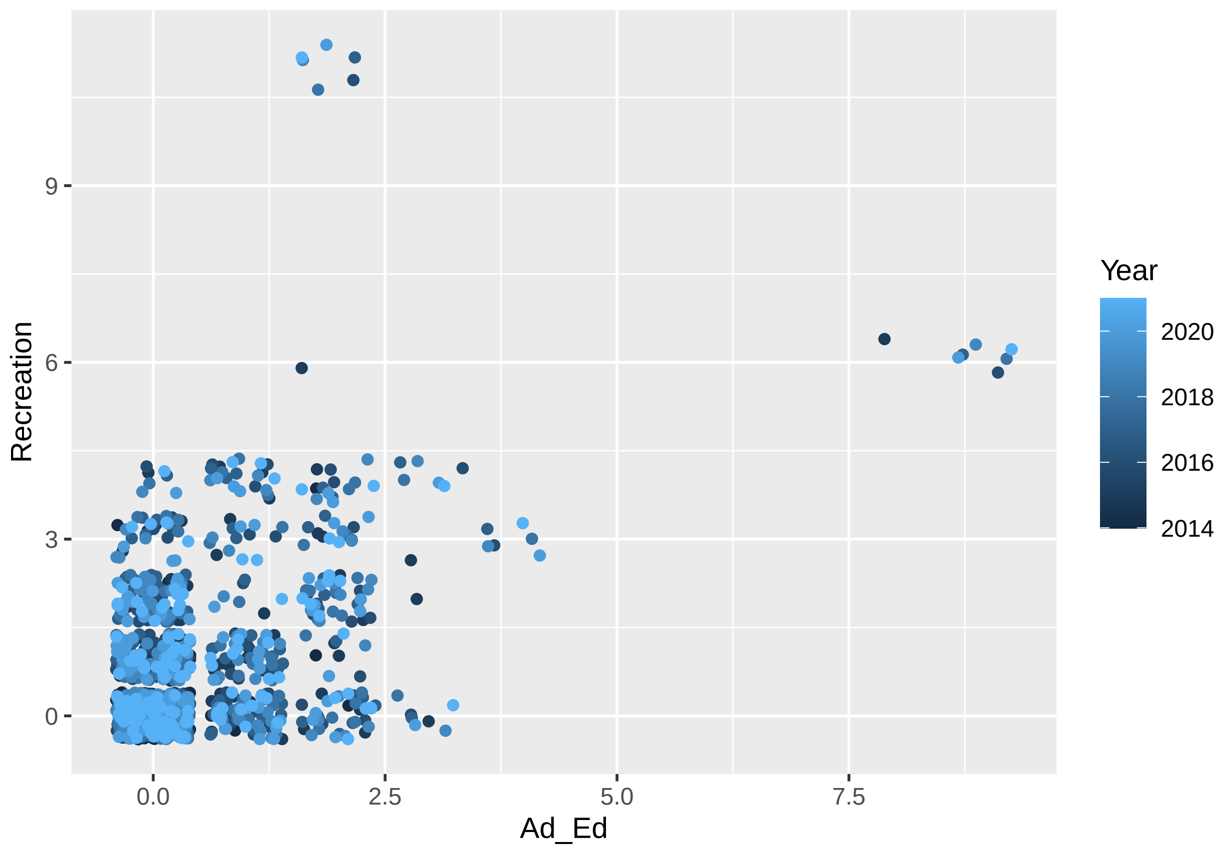

```
ggplot(df, aes(x= Ad_Ed, y = Emp_Res, colour = Year))+
  geom_jitter()
```



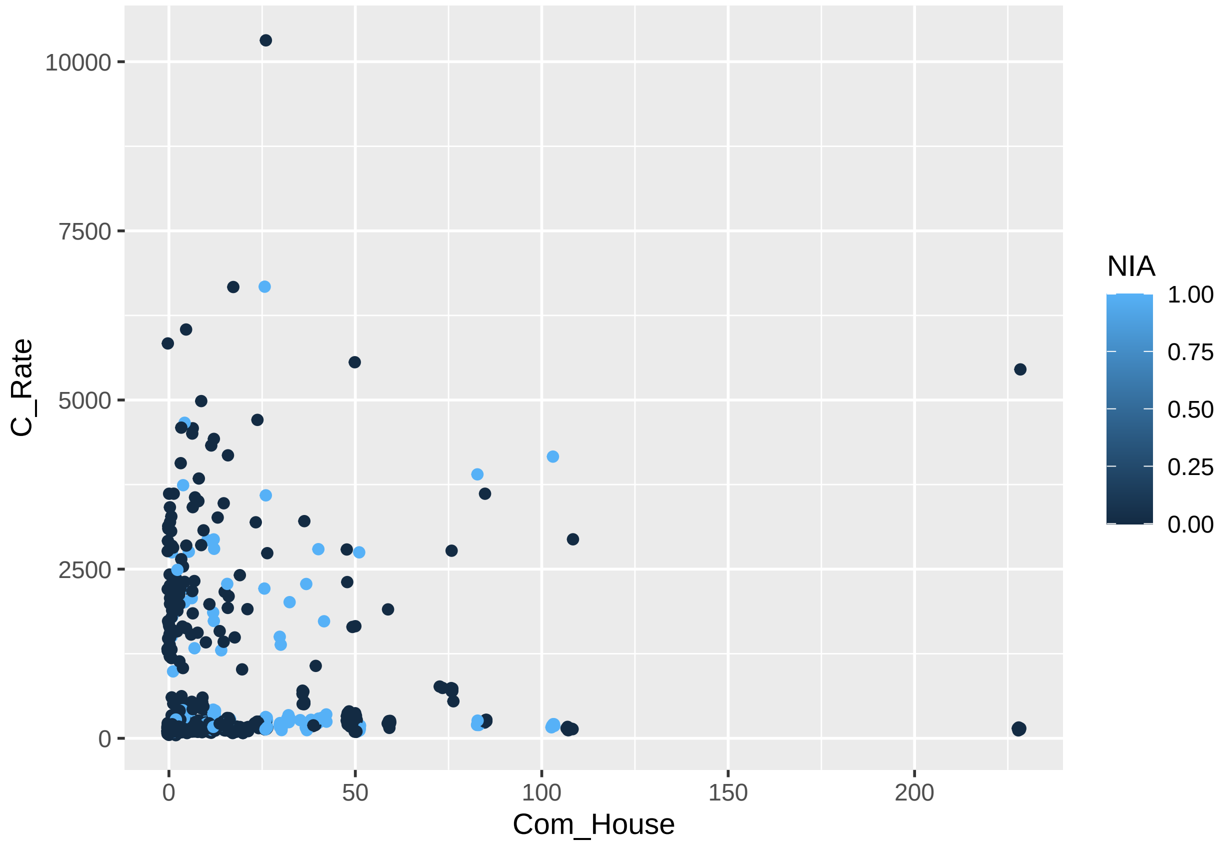
mild positive association between adult education facilities and employment resources. this is likely because the two services are probable to be placed in the same places as one would assume that accessing adult education would lead to being able to start looking for a job

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.

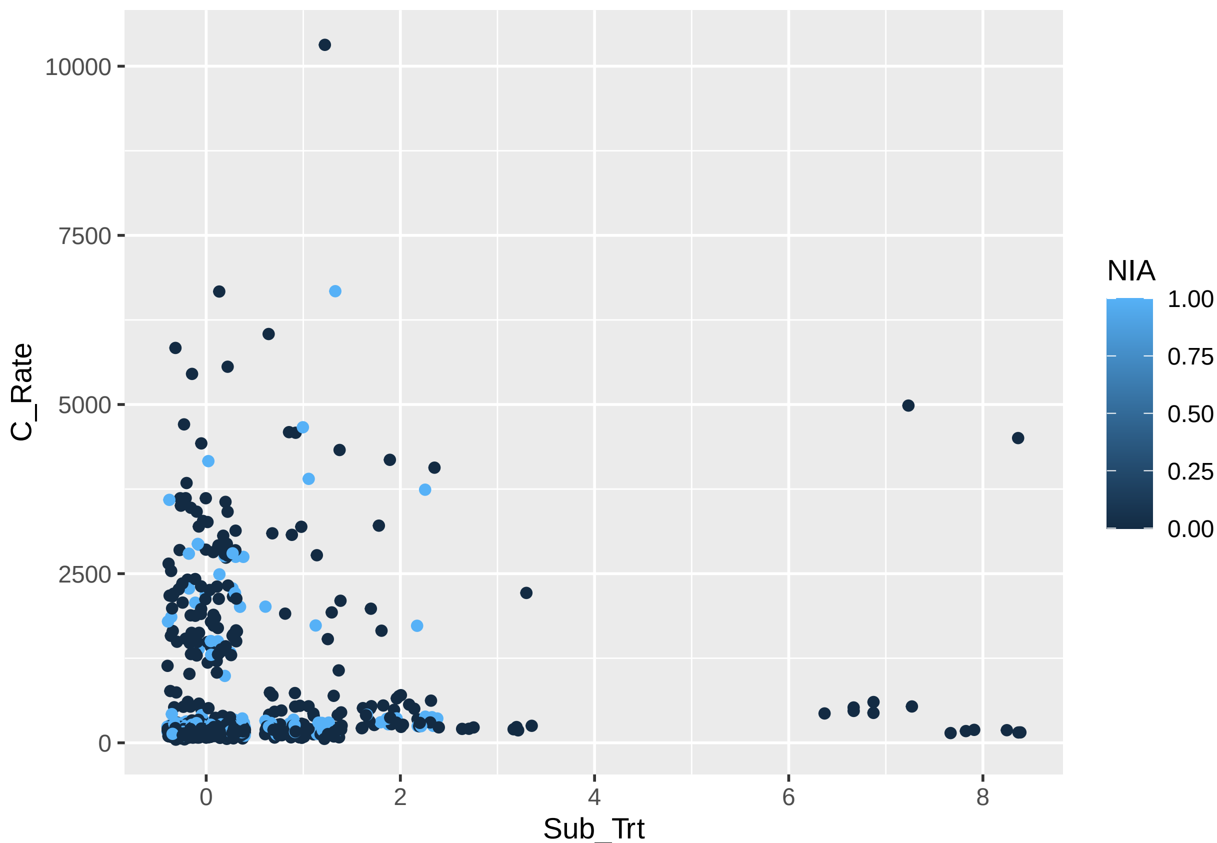
```
ggplot(df, aes(x= Ad_Ed, y = Recreation, colour = Year))+
  geom_jitter()
```




```
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.

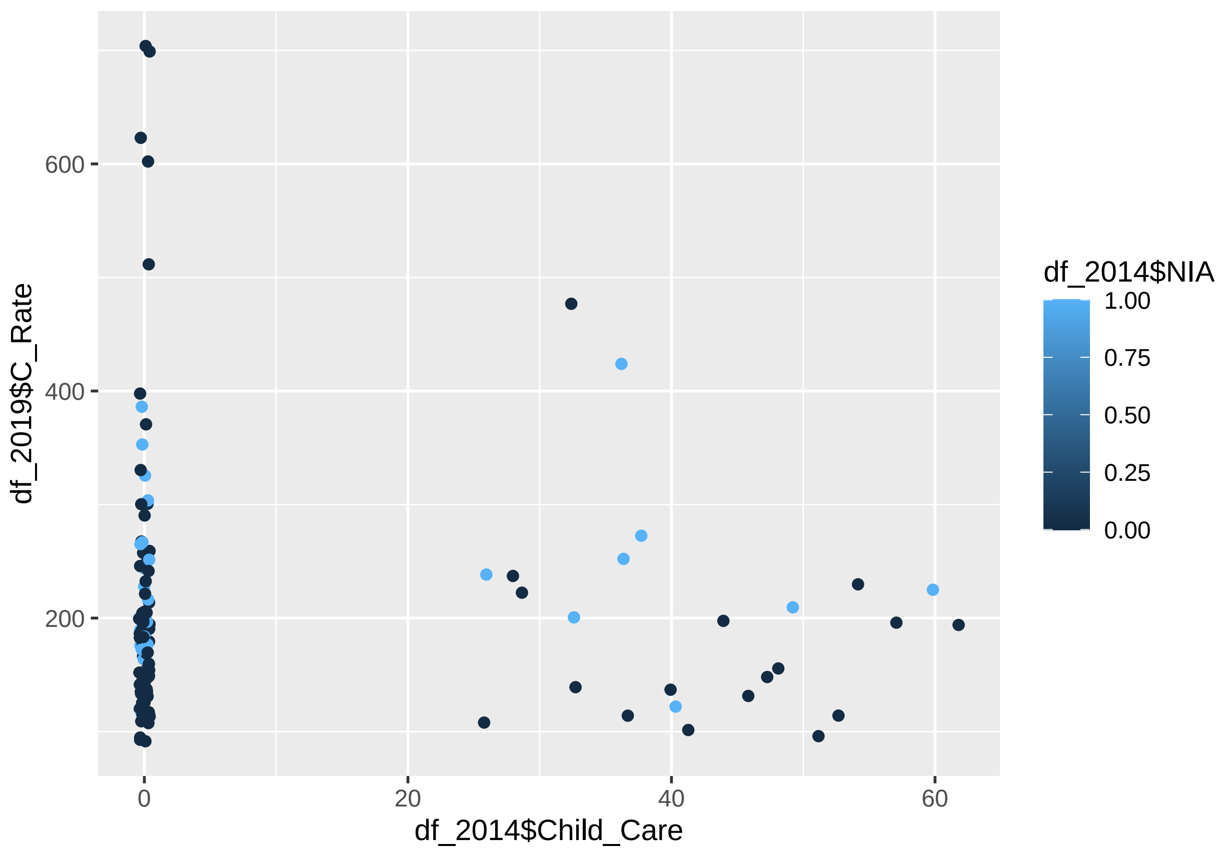
That being said, it may be easier to find relationships between crime rate and neighbourhood resources when measurements are taken years apart.

```
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.
```



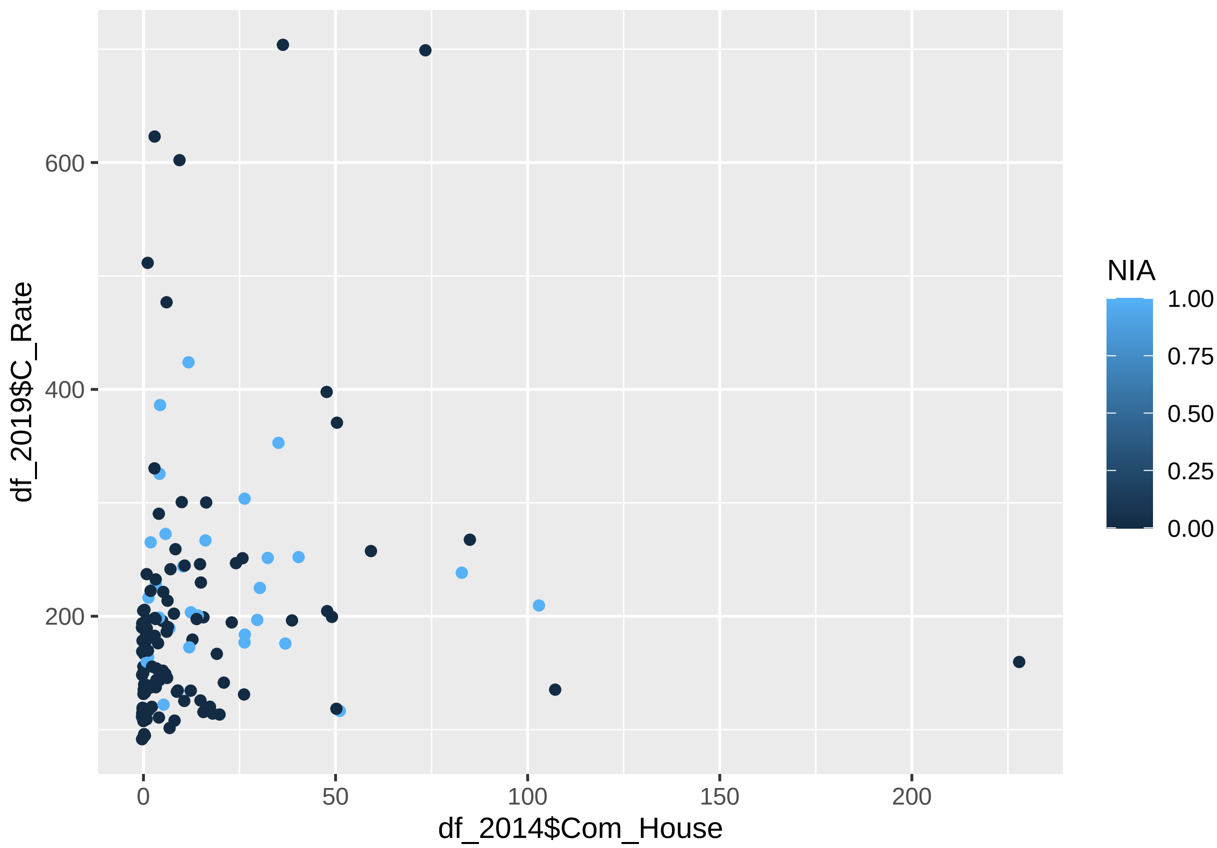
Here we can see that there is somewhat of a negative association. Where 2014 Child Care resources were low, 5 years later those same neighbourhoods had the highest crime rates. Likewise when there were ample child care resources in a neighbourhood, we an see that none of those data points were extremely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

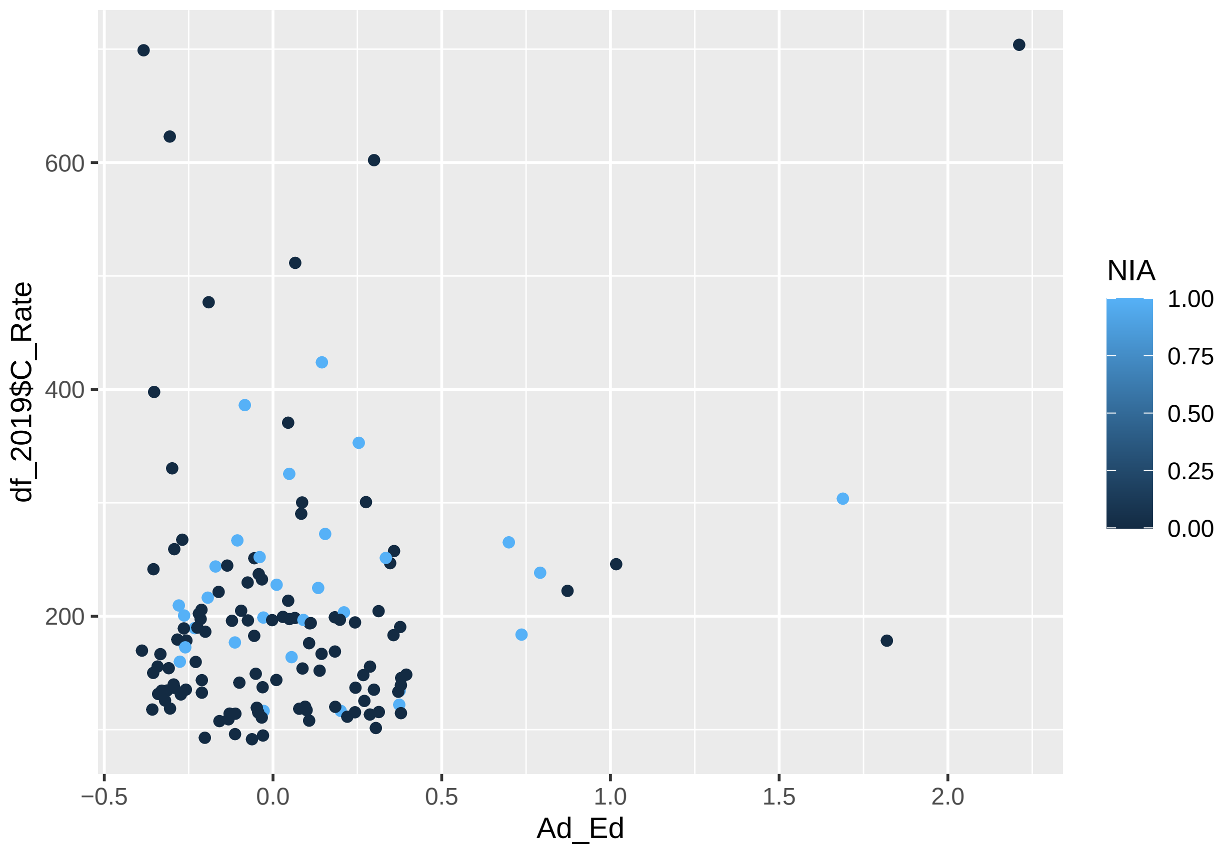
```
## Warning: Use of `df_2014$Com_House` is discouraged. Use `Com_House` instead.
```




```
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.

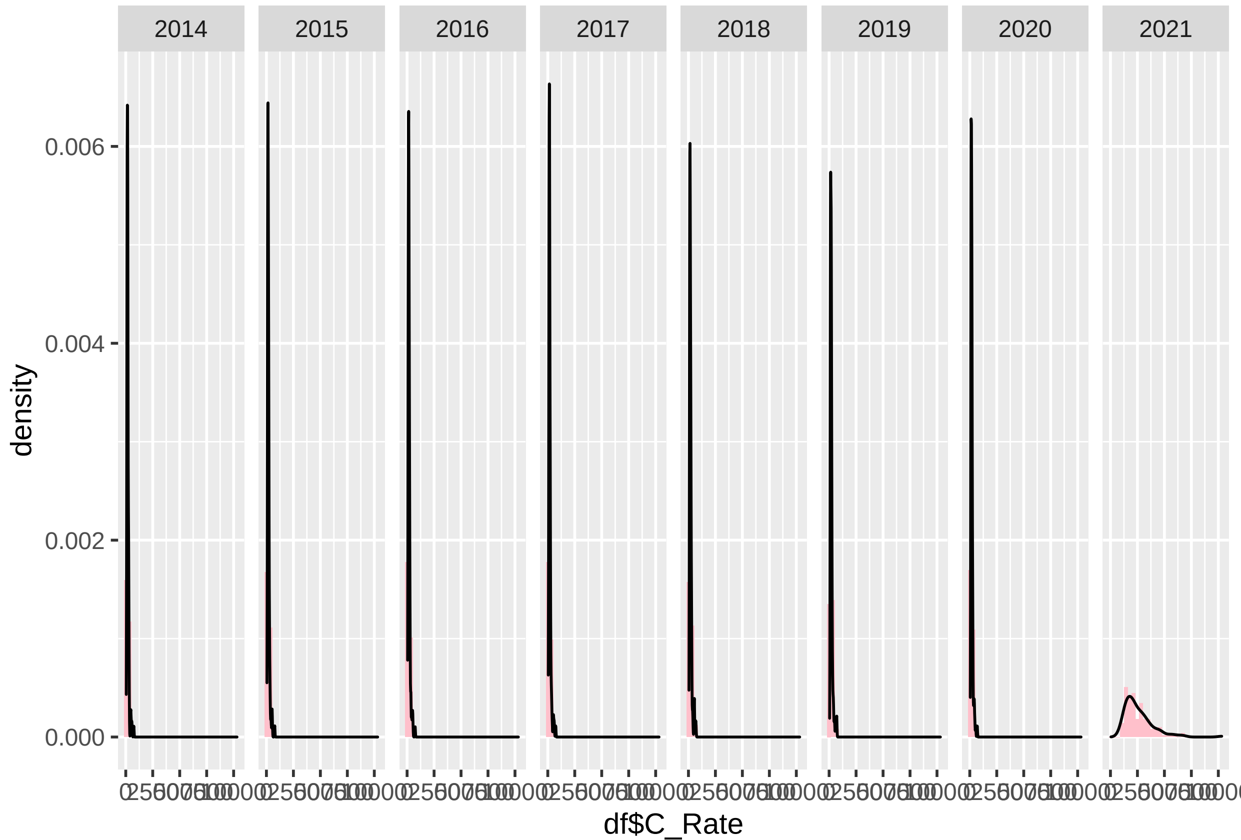
We will look at the distribution of our target variable

```
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_Re
```

```
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
```

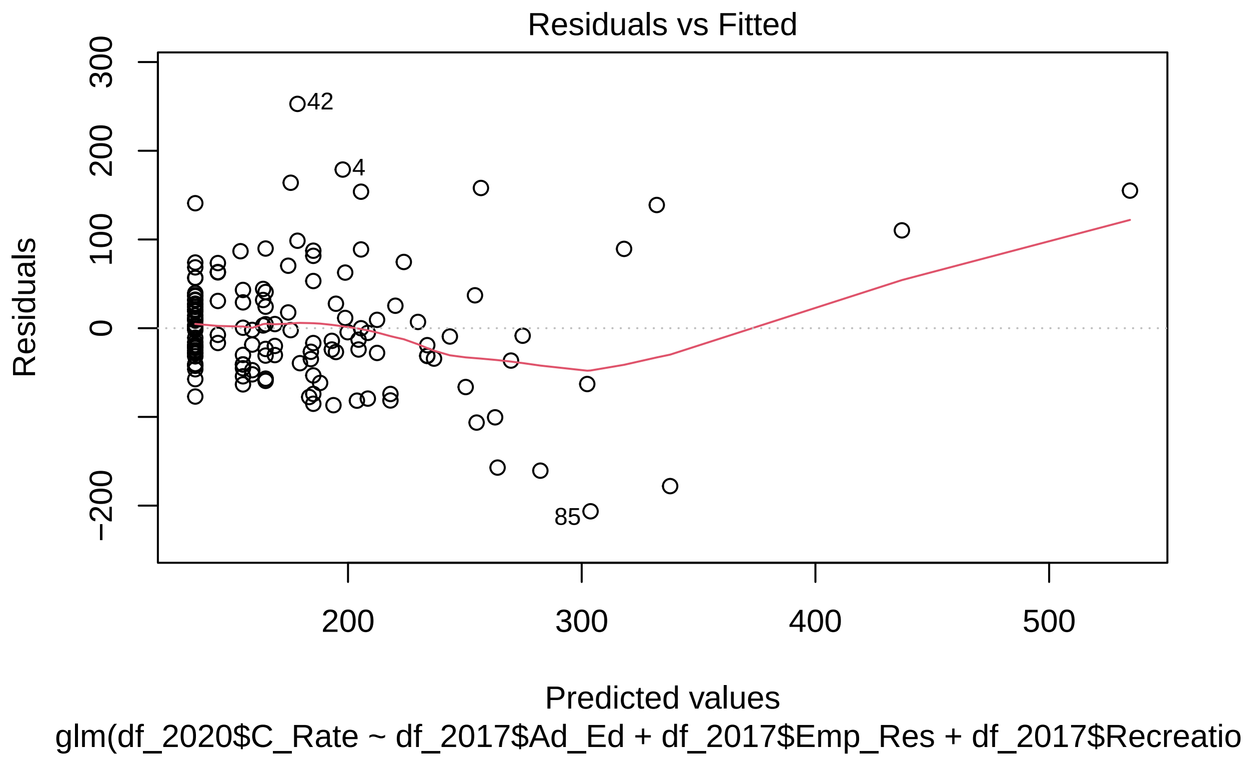
employment resources plays a more sgnificant role in this model when time between recreation facilties and time at which the response is measured is reduced.

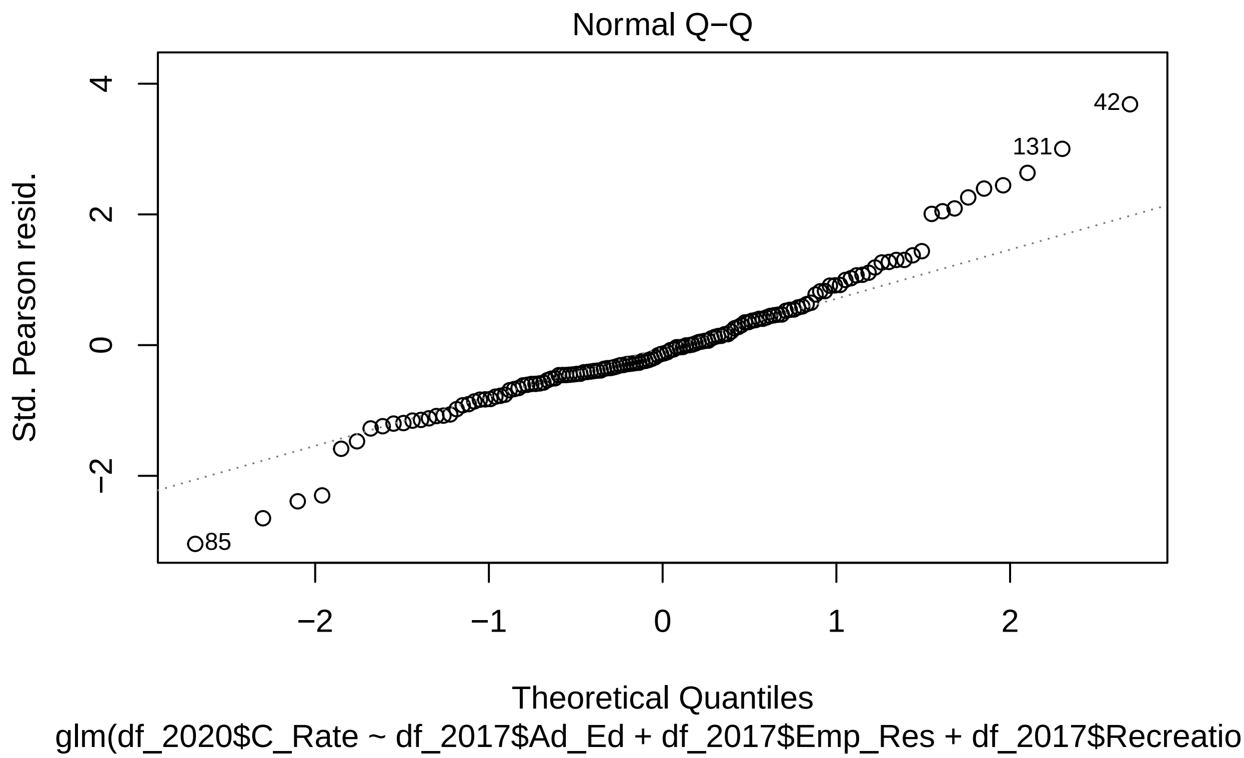
```
model_3 <- glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreation)
summary(model_3)
```

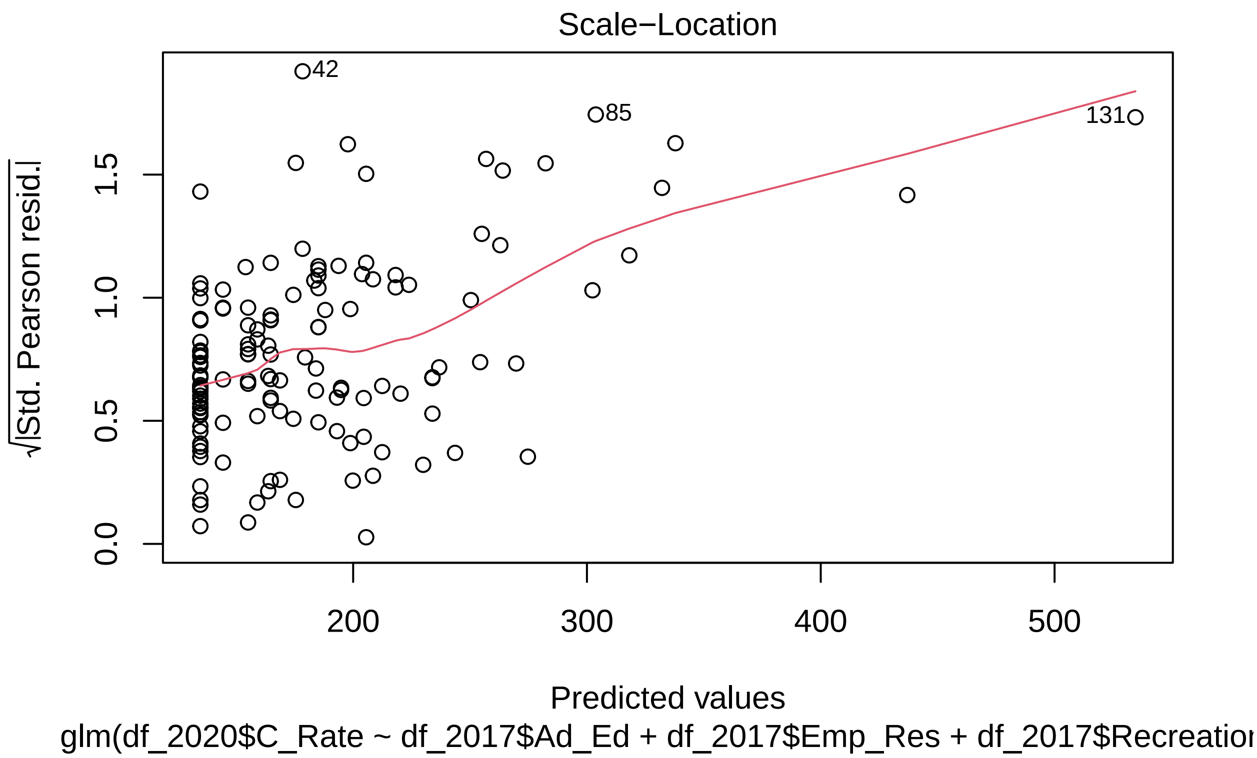
```
##
## Call:
## glm(formula = df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res +
##     df_2017$Recreation)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -206.378   -37.218    -8.911    31.889   252.839
##
## Coefficients:
##                    Estimate Std. Error 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

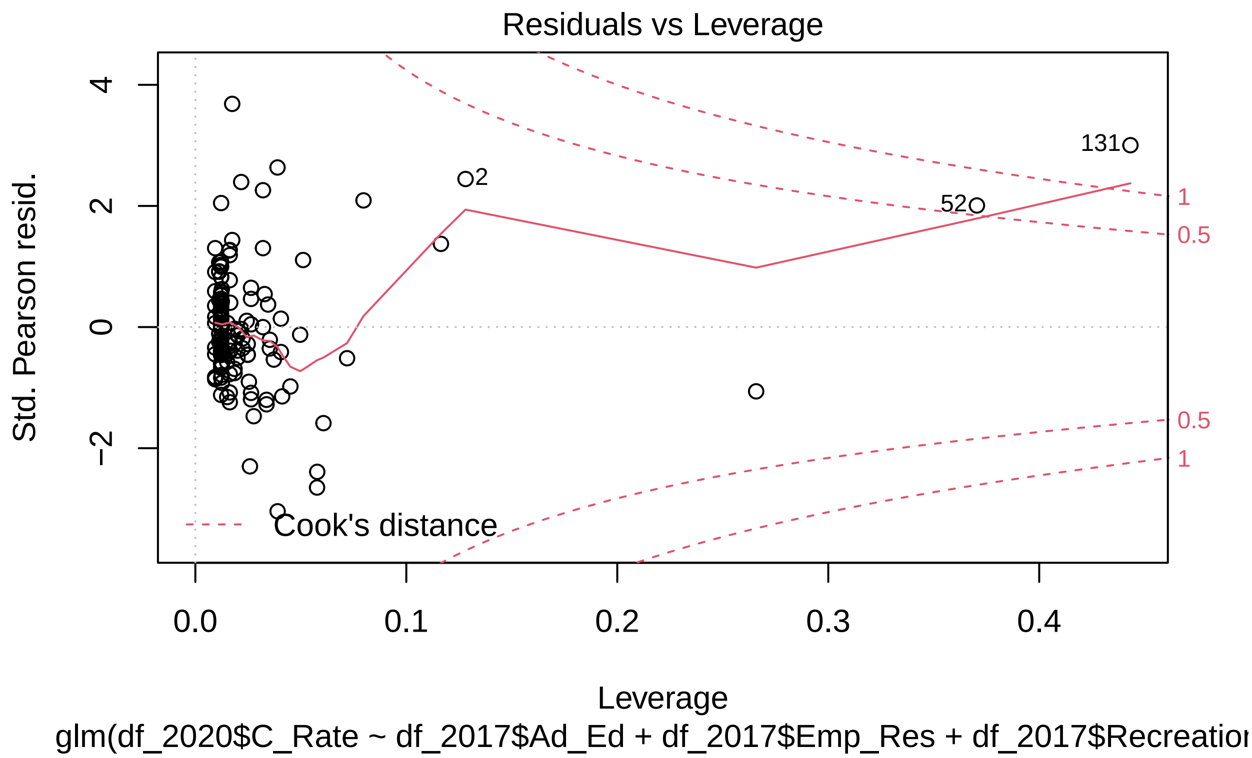
```
plot(model_3)
```


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

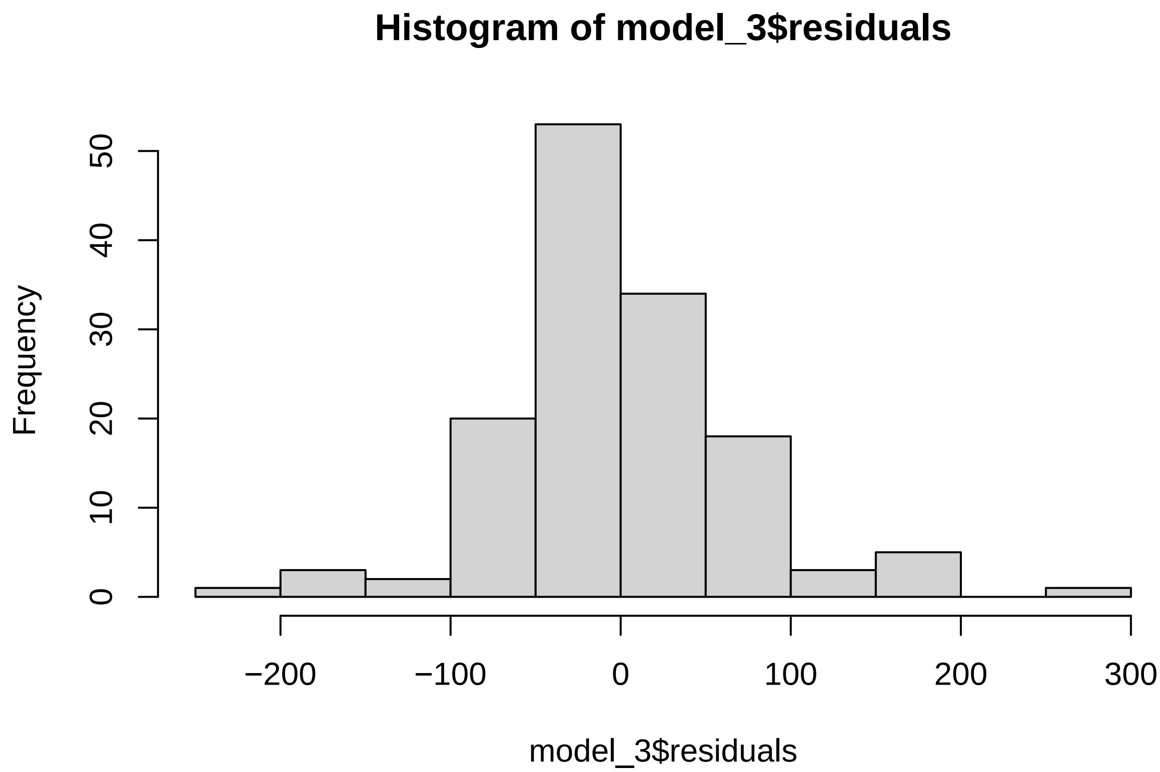

Normal Q−Q
Std. Pearson resid.
4
2
0
−2
42
131
85
−2
−1
0
1
2
Theoretical Quantiles
glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreatio



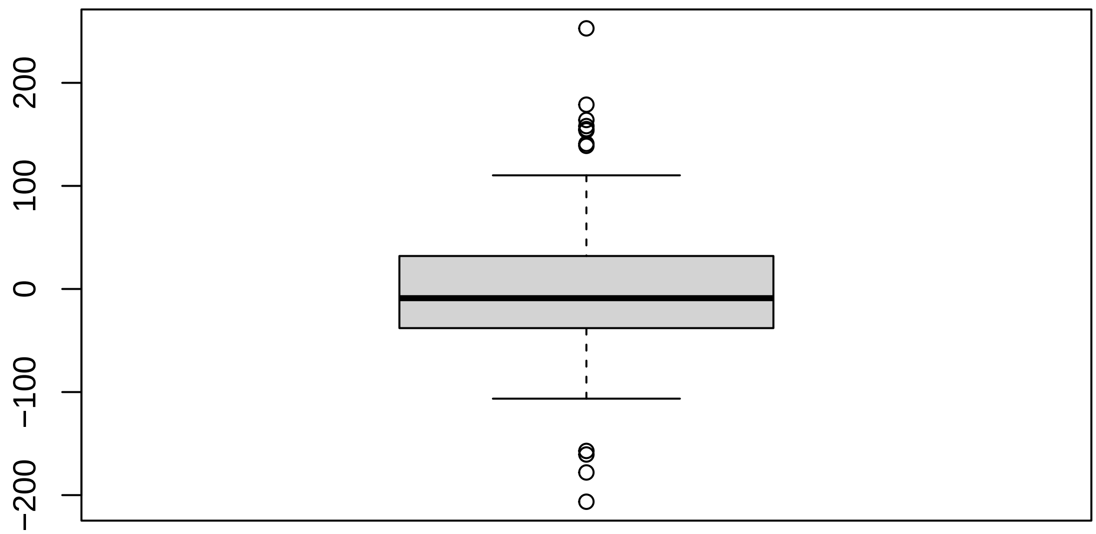
Scale−Location
42
85
131
1.5
1.0
0.5
0.0
√|Std. Pearson resid.|
200
300
400
500
Predicted values
glm(df_2020$C_Rate ~ df_2017$Ad_Ed + df_2017$Emp_Res + df_2017$Recreatio




```
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)
```

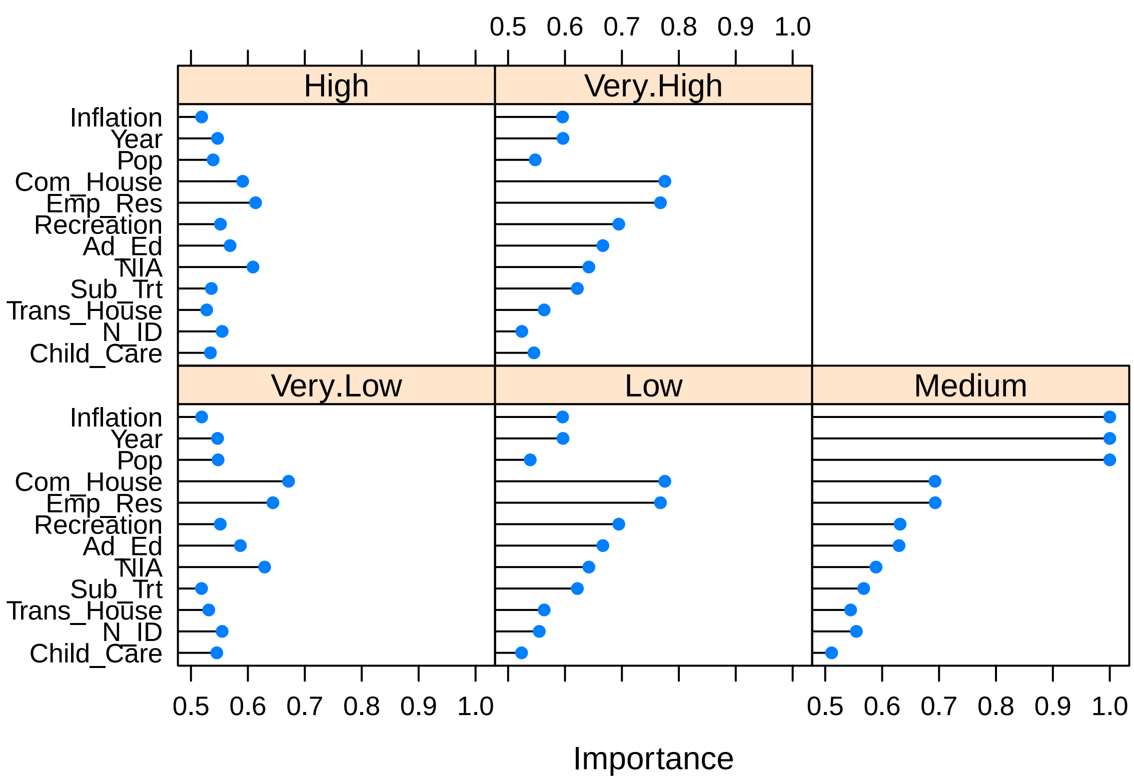
For df_new:

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.

```r
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
```

```r
plot(importance)
```

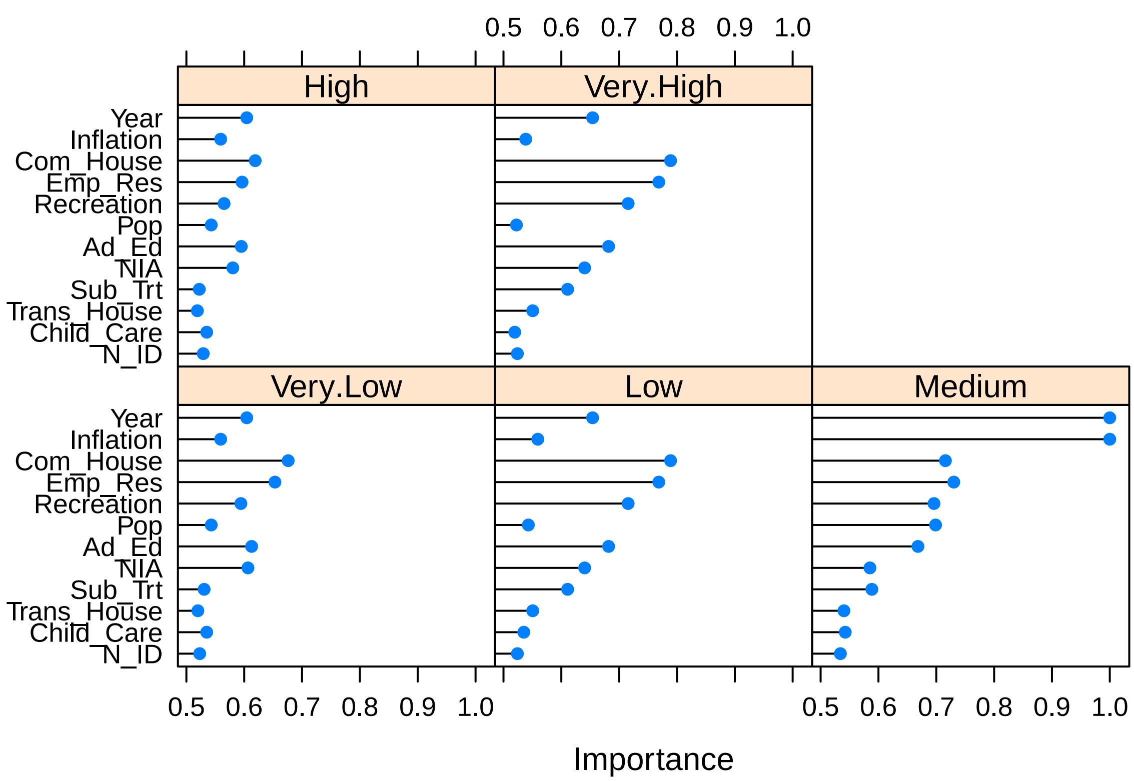


For df_2:

```
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)
```



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]:
```python
import pandas as pd
```

In [2]:
```python
Data_1 = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data.csv')
```

In [3]:
```python
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [4]:
```python
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]:
```python
X_train.shape, X_test.shape
```

Out[5]:
```
((784, 12), (336, 12))
```

In [6]:
```python
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, min_samples_leaf=5)
```

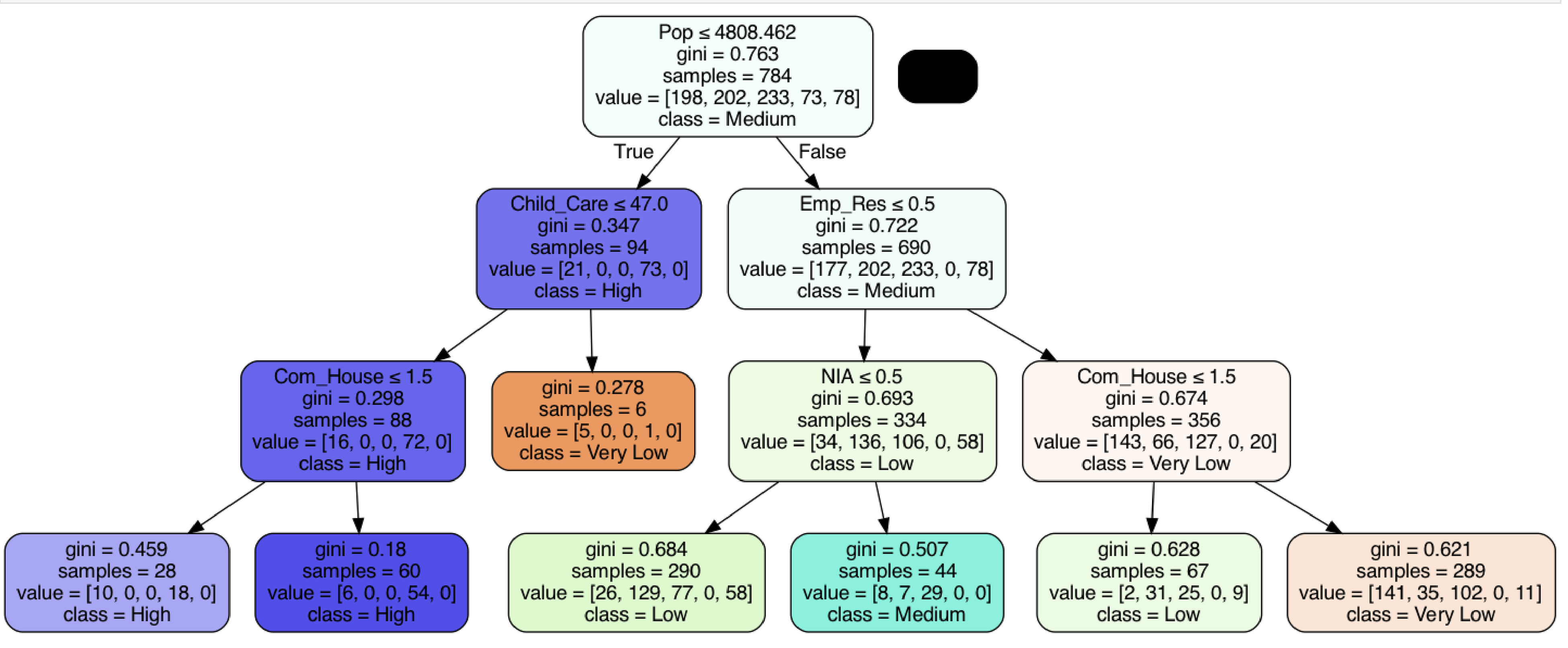
In [7]:
```python
model.fit(X_train, y_train)
```

Out[7]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [8]:
```python
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 [9]:
```python
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_characters = True, feature_names = feature_cols, class_n
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[9]:



Population appears to be the most important feature here

In [10]:
```python
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: 0.5191326530612245
Decision Tree Test Accuracy: 0.49107142857142855
```

In [11]:
```python
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

        High       0.49      0.67      0.56        82
         Low       0.39      0.85      0.54        78
      Medium       0.55      0.06      0.11       103
   Very High       0.86      0.97      0.92        39
    Very Low       0.00      0.00      0.00        34

    accuracy                           0.49       336
   macro avg       0.46      0.51      0.42       336
weighted avg       0.48      0.49      0.40       336
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1327: UndefinedMetricWar
ning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1327: UndefinedMetricWar
ning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1327: UndefinedMetricWar
ning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

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

In [12]:
```python
model.score(X_test,y_test)
```

Out[12]:
```
0.49107142857142855
```

In [13]:
```python
from sklearn.inspection import permutation_importance
```

running permutation importance on our test set:

In [14]:
```python
r = permutation_importance(model, X_test, y_test, n_repeats = 30, random_state = 0)
```

In [15]:
```python
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     0.147 +/- 0.011
Com_House0.088 +/- 0.017
Child_Care0.054 +/- 0.012
```

running it on our train set:

In [16]:
```python
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [17]:
```python
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     0.132 +/- 0.008
Com_House0.089 +/- 0.013
Child_Care0.071 +/- 0.009
Inflation0.039 +/- 0.008
Ad_Ed   0.009 +/- 0.002
```

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 [18]:
```python
train_1 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recreation','Inflation','NIA'], axis = 1)
test_1 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recreation','Inflation','NIA'], axis = 1)
```

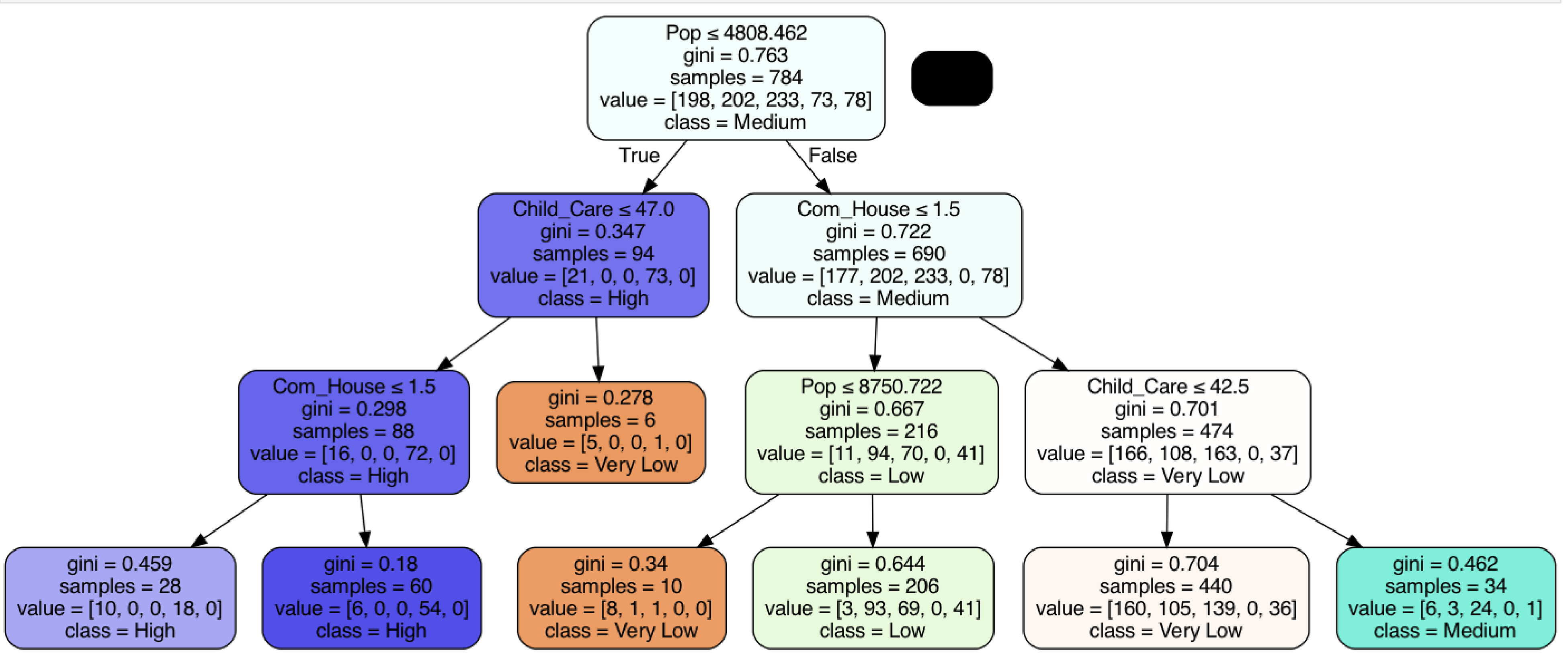
In [19]:
```python
model_1 = DecisionTreeClassifier(criterion = 'gini', random_state = 100,max_depth = 3,  min_samples_leaf=5)
```

In [20]:
```python
model_1.fit(train_1, y_train)
```

Out[20]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [21]:
```python
xvar_1 = Data_1.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','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, c
(graph_1,)= graph_from_dot_data(dot_data_1.getvalue())
Image(graph_1.create_png())
```

Out[21]:



In [22]:
```python
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: 0.461734693877551
Decision Tree Test Accuracy: 0.4375
```

we will add in inflation

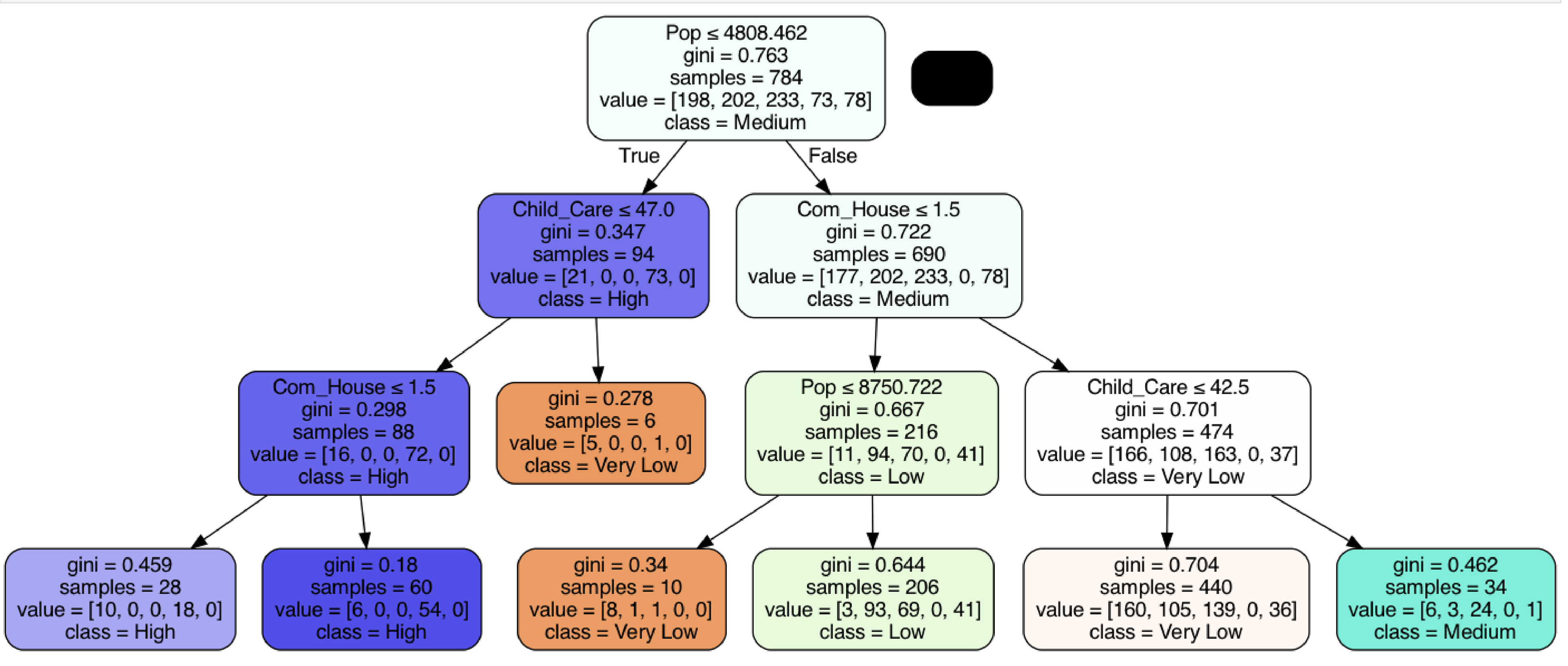
In [23]:
```python
train_2 = X_train.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recreation','NIA'], axis = 1)
test_2 = X_test.drop(['Year', 'N_ID', 'Ad_Ed','Emp_Res','Sub_Trt', 'Trans_House','Recreation','NIA'], axis = 1)

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','Recreation','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_characters = True, feature_names = feature_cols_2, c
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

Out[23]:



In [24]:
```python
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: 0.461734693877551
Decision Tree Test Accuracy: 0.4375
```

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

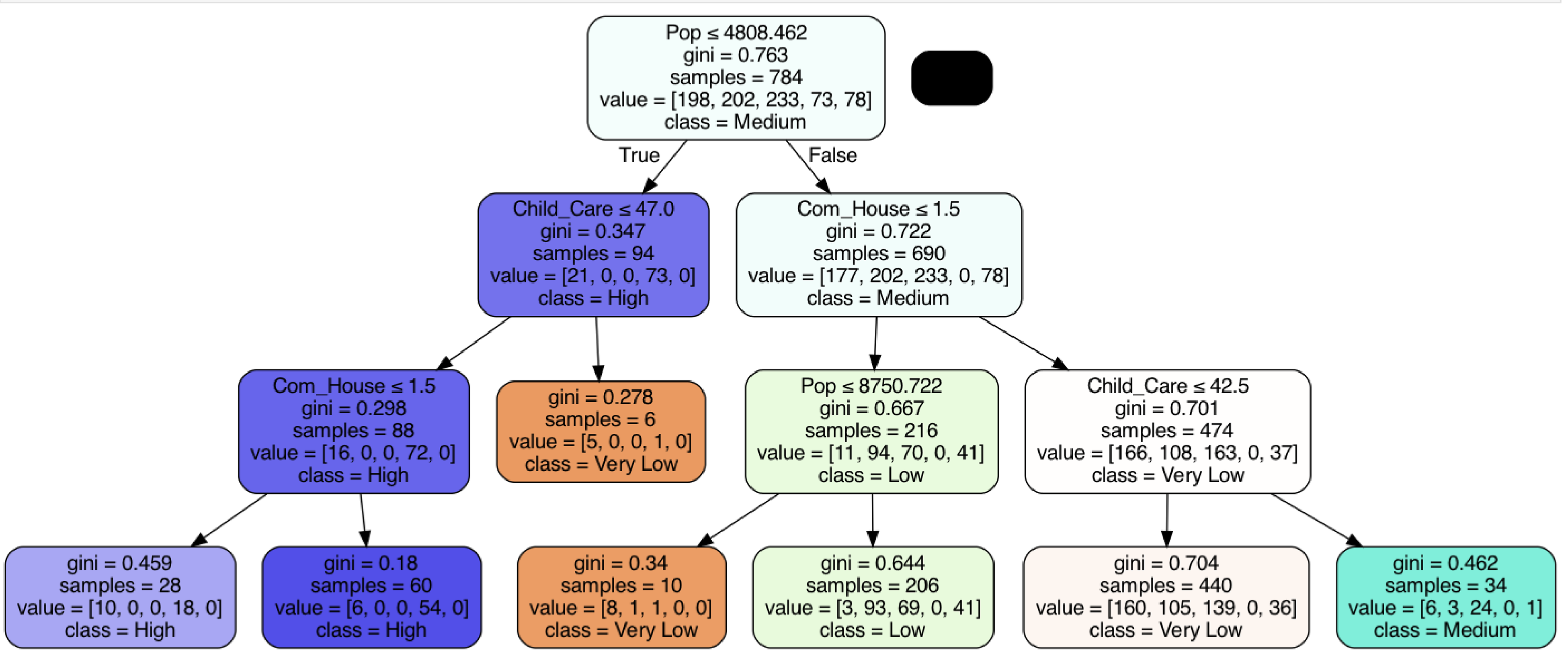
In [25]:
```python
train_3 = X_train.drop(['Year', 'N_ID', 'Inflation','Emp_Res','Sub_Trt', 'Trans_House','Recreation','NIA'], axis = 1)
test_3 = X_test.drop(['Year', 'N_ID', 'Inflation','Emp_Res','Sub_Trt', 'Trans_House','Recreation','NIA'], axis = 1)

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','Emp_Res','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_characters = True, feature_names = feature_cols_3, c
(graph_3,)= graph_from_dot_data(dot_data_3.getvalue())
Image(graph_3.create_png())
```

Out[25]:



In [26]:
```python
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: 0.461734693877551
Decision Tree Test Accuracy: 0.4375
```

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
```

In [2]:
```
Train_Data = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_
Test_Data = pd.read_csv('/Users/alyzehjiwani/Downloads/Data/Final Data Used/Crime_Data_Y
```

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]:
```
model = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3, mi
```

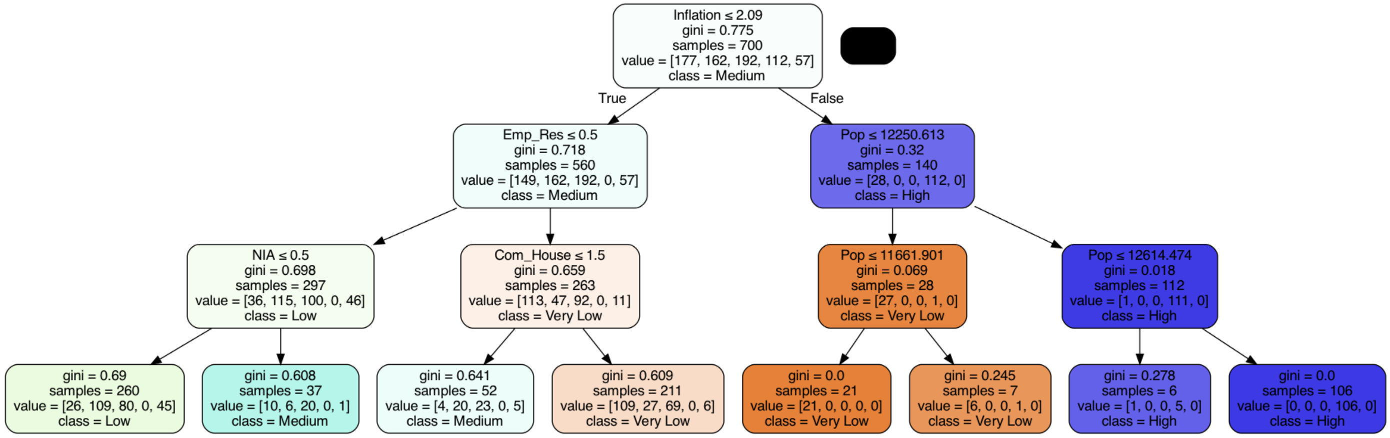
In [7]:
```
model.fit(X_train, y_train)
```

Out[7]:
```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [8]:
```
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 [26]:
```
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
(graph,)= graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[26]:



In [27]:
```
predictions = model.predict(X_test)
print ("Decision Tree Train Accuracy:", accuracy_score(y_train, model.predict(X_train)))
```

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

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

In [28]:
```
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 [29]:
```
r_1 = permutation_importance(model, X_train, y_train, n_repeats = 30, random_state = 0)
```

In [30]:
```
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 [31]:
```
Train_Data.columns.values
```

Out[31]:
```
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 [32]:
```
new_train = X_train.drop(['Year', 'N_ID','Ad_Ed','Emp_Res', 'Sub_Trt', 'Trans_House','NI
new_test = Test_Data.drop(['Year', 'N_ID','Ad_Ed','Emp_Res', 'Sub_Trt', 'Trans_House','N
```

In [33]:
```
new_train.shape, y_train.shape, new_test.shape
```

Out[33]:
```
((700, 5), (700,), (420, 5))
```

In [34]:
```
model_2 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
```

In [35]:
```
model_2.fit(new_train, y_train)
```

Loading [MathJax]/extensions/Safe.js

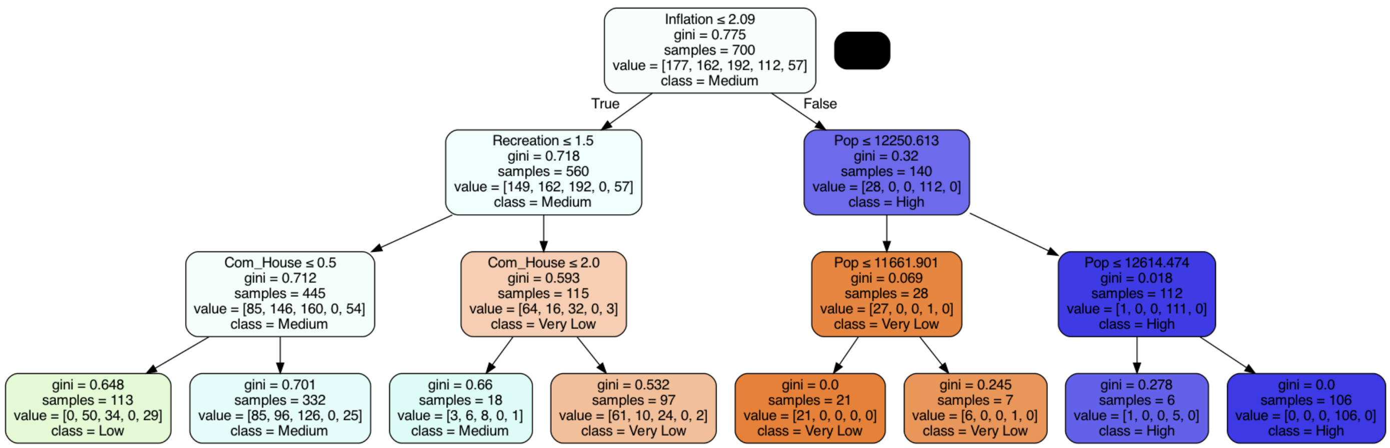
Out[35]:

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

In [36]:

```python
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
(graph_1,)= graph_from_dot_data(new_dot_data.getvalue())
Image(graph_1.create_png())
```

Out[36]:



In [37]:

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

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

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 [38]:

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

In [39]:

```python
model_3 = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 3,
```

In [40]:

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

Out[40]:

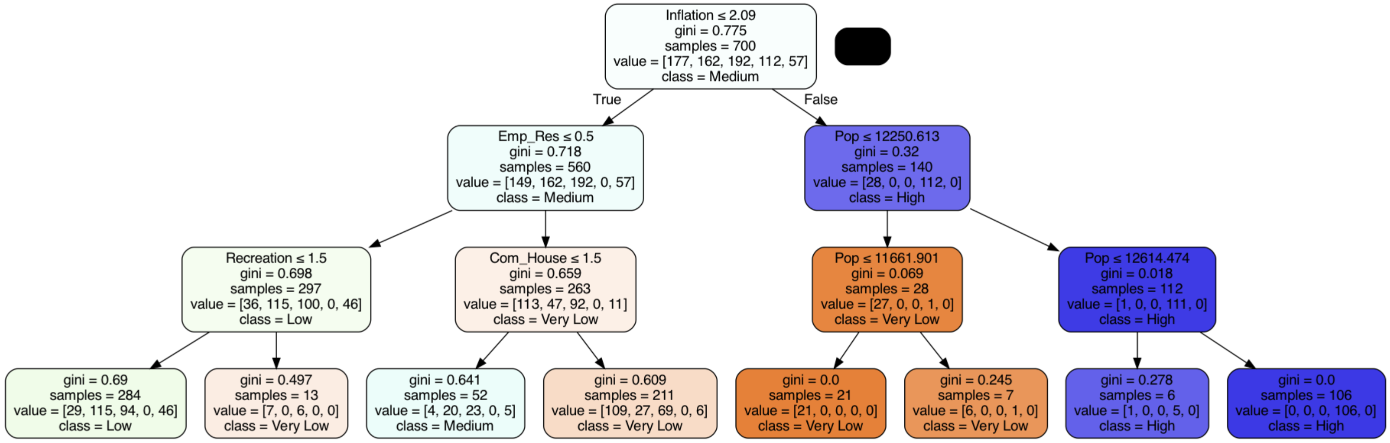
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In [41]:

```python
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
(graph_2,)= graph_from_dot_data(dot_data_2.getvalue())
Image(graph_2.create_png())
```

Loading [MathJax]/extensions/Safe.js

Out[41]:



In [42]:

```
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 [ ]:

Loading [MathJax]/extensions/Safe.js