CUNY SPS DATA621 FINAL Project: Predicting Violent Crime

Anil Akyildirim, John K. Hancock, John Suh, Emmanuel Hayble-Gomes, Chunjie Nan

05/22/2020

Contents

Abstract

Violent crime is a major social problem which lowers the quality of life for communities. Besieged communities are always looking for ways to understand and predict violent crimes. This project assessed the use of linear regression models to predict the incidences of violent crimes. We chose the Communities and Crime normalized dataset from the UCI Machine Learning Repository. We used four forms of regression analysis, Multiple Regression, Ridge Regression, Lasso Regression, and Elastic Net Regression to build models based on 100 independent variables.

Key words

Violent Crime, Linear Regression, Ridge Regression, Lasso Regression, Elastic Net Regression RMSE, Rsquared, lambda

Introduction:

Communities have long struggled to prevent and constrain their rates of violent crime. Being able to predict the rate of violent crime allows these communities to take the necessary steps and investments in resources to prevent its occurence. Understanding the key variables that contribute to violent crime goes a long way towards allocation of resources, police, educational, recreational, as a means of preventing further incidences.

We selected the Communities and Crime dataset from the UCI Machine Learning Repository[1] Link. "Many variables are included so that algorithms that select or learn weights for attributes could be tested. However, clearly unrelated attributes were not included; attributes were picked if there was any plausible connection to crime (N=122), plus the attribute to be predicted (Per Capita Violent Crimes). The variables included in the dataset involve the community, such as the percent of the population considered urban, and the median family income, and involving law enforcement, such as per capita number of police officers, and percent of officers assigned to drug units. The per capita violent crimes variable was calculated using population and the sum of crime variables considered violent crimes in the United States: murder, rape, robbery, and assault. There was apparently some controversy in some states concerning the counting of rapes. These resulted in missing values for rape, which resulted in incorrect values for per capita violent crime. These cities are not included in the dataset. Many of these omitted communities were from the midwestern USA." Id

This project uses four different regression modeling techniques, Multiple Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net Regression to make predictions on the dataset. Before building the models we prepared the data by looking for missing data points, removing correlated variables, normalizing the skewed variables, and splitting the data into training and test sets.

Our goal was to see if regression analysis could use this data to discover key variables that influence the occurence of violent crime.

Literature review:

For this project, We reviewed, "USING MACHINE LEARNING ALGORITHMS TO ANALYZE CRIME DATA" by Lawrence McClendon and Natarajan Meghanathan (Jackson State University, 1400 Lynch St, Jackson, MS, USA) (Machine Learning and Applications: An International Journal (MLAIJ) Vol.2, No.1, March 2015)[2] They performed a comparative analysis between the UCI dataset and the state of Mississippi's own crime statistics in order to discern patterns of violent crime. They used Linear Regression and Decision Tree algorithms. They found that Linear regression was more effective.

We also reviewed, "A Convex Framework for Fair Regression" by Richard Berk, Hoda Heidari , Shahin Jabbari, Matthew Joseph, Michael Kearns , Jamie Morgenstern, Seth Neel , and Aaron Roth (Department of Statistics, Department of Criminology, Department of Computer and Information Science of the University of Pennsylvania)(June 9, 2017) Citation: arXiv:1706.02409 [cs.LG] [3] In their analysis, they looked at the use of regularization to reduce model complexity and overfitting. Their analysis was the inspiration for us to use the RIDGE, LASSO, and Elastic Net regression models to control over-fitting and reducing model complexity.

Lastly, we reviewed "A Comparative Study to Evaluate Filtering Methods for Crime Data Feature Selection" by Masila Abdul Jalil, Fatihah Mohd, and Noor Maizura Mohamad Noor (School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu, Malaysia) (October 2017)[4] Their objective was to find a attributes from a dataset to classify the crimes into three different categories; low, medium and high. We were inspired by their use of feature selection.

Methodology

The dataset is comprised of 1,994 observations and 1oo variables. Once the variables, communityname, state, OtherPerCap, were removed, all of the variables were numeric. Given the number of variables, it was obvious to us that the models may wind up being overly complex and over-fitted on the training data.

Our next step was to prepare the data for analysis. We did the following:

A. Checked for Missing Data and impute missing values B. Checked for Multicollinearity C. Checked for Normality and normalize the data E. Split the Communities_Crime_Train dataset into Train and Test

The dataset did not contain any missing values, so we moved on to checking for correlations between the now 97 variables ignoring the response variable, ViolentCrimesPerPop. Multicollinearity happens when there is a high correlation between one or more variables which leads to redundant data.

A variance inflation factor(VIF) detects multicollinearity in regression analysis. It estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model. From the car package, we used the function "vif" to score the variables on multicollinearity. A score of 10 or more meant that the variables were removed from the dataset. Here, we removed those independent variables that were correlated which reduced the number of variables from 100 to 34.

Next, we checked for skewed data and transformed negatively and positively skewed data to make them normal. For the neatgively skewed data we created a function which applied this function:log10(max(x+1) - x). For the positively skewed data, we simply took the square root of each value.

Lastly, we split the dataset into training and test sets. The split was done once for the linear regression step model, and split a second time for the other three models, Ridge Regression, LASSO Regression, and the Elastic Net Regression models. For each of these three models, we created a matrix for all of the independent variables and copied the response variable into its own variable, "y". Next,using cross validation we tried a sequence of lambdas, the regularization parameter.

Experimentation and Results

We built four regression models: Linear REGRESSION model (OLS) using stepwise coefficient selection RIDGE REGRESSION model LASSO REGRESSION model ELASTIC NET REGRESSION model.

The OLS linear regression model, VC_Step_model, was built using the 34 independent variables. The step() function used backwards and forwards elimination to determine the best coefficients to build the best model which had an RMSE of 0.136 and a Rsqaure score of .63 when the model predicted on the test set. To reduce the variance of the model, we eliminated 64 correlated variables.[6]

The Ridge regression is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This modification is done by adding a penalty parameter that is equivalent to the square of the magnitude of the coefficients. We reduce the sum of the squares residuals and penalize the size of parameter estimates. We began by splitting the data once again into train and test sets. Next, we performed cross-validation to select the value of lambda that minimizes the cross-validated sum of squared residuals. [6] Using gmlet with an alpha=0 and a lambda=0.009326, We applied Ridge regression and got an RMSE of .133 and an Rsqaure of .64 which is an improvement over the OLS linear regression model.

"Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (the so-called L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty)".As a result, for high values of ??, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.[6] We followed the exact same steps as with the Ridge regression model. We got an RMSE of .133 and an Rsquare of .64 which is identical to the Ridge model.

"Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds."[6]. Elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods. As with the previous two models, Ridge and LASSO, we have to set the lambda parameter, but unlike the previous two, we also have to set the alpha parameter. We used the caret package to automatically set and tune the lambda and alpha parameters using a combination of 25 different lambdas and alpha values. The end result is that the best alpha was .1375 and the best lambda is 0.0178. The RMSE and Rsqaure values were 0.1288544 and 0.66

Discussion and Conclusions

The table below summarizes our findings of the four models. The Elastic Net Regression model,"VC_ENR_model", had the lowest RMSE and the highest Rsquare score.

ModelName	RMSE	Rsquare
VC_Step_Model	0.135697	0.6284
VC_Ridge_Model	0.1330519	0.6398185
VC_Lasso_Model	0.1330618	0.6397077
VC_ENR_model	0.1288544	0.6634376

To start the discussion we want to examine some of the key coefficients of the model. Starting with the top three coefficients of this model:

- "PctTeen2Par" which is the the percentage of kids aged 12 to 17 in two parent homes
- "PctHousOccup", the percentage of housing occupied
- "PctSameCity85", the percentage of people living in the same city

We can infer that teens may be a statistically significant source of the problems with violent crime. Additionally, the higher the occupancy of housing, or density of population also contributes to violent crime. Finally, the less transient the population is–here defined as people living in the same citry for the last 5 years"–shows that it has a negative effect on violent crime.

PctTeen2Par	PctHousOccup	PctSameCity85
1.244844	0.5369223	-0.4459489

The next group of coefficients can be categorized as housing related. These housing coefficients: "MedOwnCostPctIncNoMtg, median owners cost as a percentage of household income - for owners without a mortgage."pctWFarmSelf",percentage of households with farm or self employment income,"MedNumBR", median number of bedrooms, and"PctVacMore6Mos", percent of vacant housing that has been vacant more than 6 months. All have a negative impact on violent crime. We can infer that more housing stability may act to lower the rate of violent crime.

These next housing related coefficients, "PctVacantBoarded", the percent of vacant housing that is boarded up, "NumInShelters", number of people in homeless shelters, and "NumStreet" , number of homeless people counted in the street

MedOwnCostPctIncNoMtg	-0.0547496
pctWFarmSelf	-0.0537252
MedNumBR	-0.0161548
PctVacMore6Mos	-0.0655400
PctVacantBoarded	0.1859722
NumInShelters	0.0979270
NumStreet	0.1724660

The next set of coefficients of interest relate to ethnicity and income. "blackPerCap", "indianPerCap", and "HispPerCap" show that the higher income per each ethnic group lowers the rate of violent crime.

blackPerCap	indianPerCap	HispPerCap
-0.0281611	-0.0028838	-0.0413251

In sum, we can infer from this model that the more stable a community when it comes to housing, specifically home ownership, income levels of ethic groups, these community attributes lower the rate of violent crime. Converesely, if there is a high percentage of teens, aged 12 to 17, there is a greater chance of violent crime.

Although the Elastic Net Regression model, "VC_ENR_model", performed the best of all of the models, we cannot infer too much from the model since it's Rsquare score only explains about 66% of the variation.

R statistical programming code.

Load the training data set

Data Exploration

Descriptive Statistics

To start the process, we copied the raw dataset into a new variable called "Communities_Crime_Train". We can start exploring our training data set by looking at basic descriptive statistics. We see that there are 1994 observations and 100 variables.

```
Communities_Crime_Train = as.data.frame(Communities_Crime_raw)
dim(Communities_Crime_Train)
```

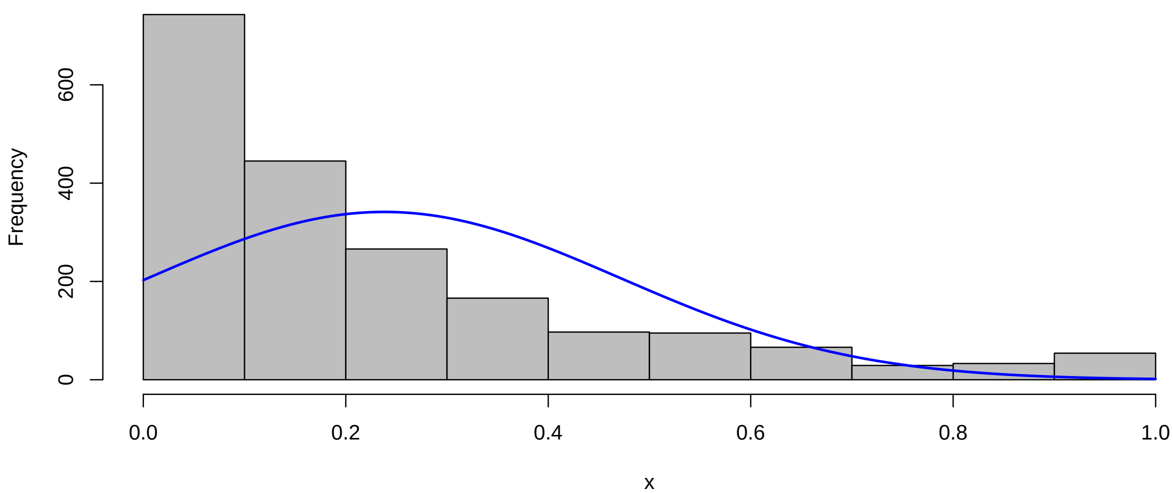
```
## [1] 1994  100
```

communityname, state, OtherPerCap

```
remove <- c("communityname", "state", "OtherPerCap")

Communities_Crime_Train[remove]<-NULL
```

The response variable for this project will be "violentPerPop".

```
plotNormalHistogram(Communities_Crime_Train$ViolentCrimesPerPop)
```



Data Preparation

In this section, we prepared the dataset for linear regression modeling. We did the following:

A. Checked for Missing Data and impute missing values B. Checked for Multicollinearity C. Checked for Normality and normalize the data E. Split the Communities_Crime_Train dataset into Train and Test

```
dimensions <- dim(Communities_Crime_Train)
```

A. Check for Missing Values

Below, we created a metastats table for the dataset and as you can see there are no missing values

```
#Check for Missing Values
metastats <- data.frame(psych::describe(Communities_Crime_Train))
metastats <- tibble::rownames_to_column(metastats, "attributes")
metastats["pct_complete"] <- round(metastats["n"]/dimensions[1], 3)
metastats$attributes <- gsub('\\*', '', metastats$attributes)
metastats %>% dplyr::select(1, 15) %>% filter(pct_complete <1 )
```

```
## [1] attributes   pct_complete
## <0 rows> (or 0-length row.names)
```

B. Check for Multicollinearity

Multicollinearity happens when there is a high correlation between one or more variables which leads to redundant data.

The Variance Inflation Factor

"A variance inflation factor(VIF) detects multicollinearity in regression analysis. Multicollinearity is when there's correlation between predictors (i.e. independent variables) in a model; it's presence can adversely affect your regression results. The VIF estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model."

Variance Inflation Factor[5]

Obtain the Variance Inflation Factor by calling the vif function. A VIF of more than 10, then it indicates multicollinearity.

```
#Check for Multicollinearity

coll_model <- lm(ViolentCrimesPerPop~.,data=Communities_Crime_Train)
alias(coll_model)
```

```
## Model :
## ViolentCrimesPerPop ~ population + householdsize + racepctblack +
##     racePctWhite + racePctAsian + racePctHisp + agePct12t21 +
##     agePct12t29 + agePct16t24 + agePct65up + numbUrban + pctUrban +
##     medIncome + pctWWage + pctWFarmSelf + pctWInvInc + pctWSocSec +
##     pctWPubAsst + pctWRetire + medFamInc + perCapInc + whitePerCap +
##     blackPerCap + indianPerCap + AsianPerCap + HispPerCap + NumUnderPov +
##     PctPopUnderPov + PctLess9thGrade + PctNotHSGrad + PctBSorMore +
##     PctUnemployed + PctEmploy + PctEmplManu + PctEmplProfServ +
##     MalePctDivorce + MalePctNevMarr + FemalePctDiv + TotalPctDiv +
##     PersPerFam + PctFam2Par + PctKids2Par + PctYoungKids2Par +
##     PctTeen2Par + PctWorkMomYoungKids + PctWorkMom + NumIlleg +
##     PctIlleg + NumImmig + PctImmigRecent + PctImmigRec5 + PctImmigRec8 +
##     PctImmigRec10 + PctRecentImmig + PctRecImmig5 + PctRecImmig8 +
##     PctRecImmig10 + PctSpeakEnglOnly + PctNotSpeakEnglWell +
##     PctLargHouseFam + PctLargHouseOccup + PersPerOccupHous +
##     PersPerOwnOccHous + PersPerRentOccHous + PctPersOwnOccup +
##     PctPersDenseHous + PctHousLess3BR + MedNumBR + HousVacant +
##     PctHousOccup + PctHousOwnOcc + PctVacantBoarded + PctVacMore6Mos +
##     MedYrHousBuilt + PctHousNoPhone + PctWOFullPlumb + OwnOccLowQuart +
##     OwnOccMedVal + OwnOccHiQuart + RentLowQ + RentMedian + RentHighQ +
##     MedRent + MedRentPctHousInc + MedOwnCostPctInc + MedOwnCostPctIncNoMtg +
##     NumInShelters + NumStreet + PctForeignBorn + PctBornSameState +
##     PctSameHouse85 + PctSameCity85 + PctSameState85 + LandArea +
##     PopDens + PctUsePubTrans
```

The table below shows the variables and their multicollinearity scores.

```
multicoll <- data.frame(round(vif(coll_model),4))
multicoll$variables <- rownames(multicoll)
colnames(multicoll) <- c("scores", "vars")
rownames(multicoll) <- NULL
multicoll  %>% dplyr::select(2,1) %>%  filter(scores > 10) %>% arrange(desc(scores))
```

```
##                 vars    scores
## 1        TotalPctDiv 1030.2351
## 2        OwnOccMedVal  573.0127
## 3     PctPersOwnOccup  567.2178
## 4       PctHousOwnOcc  547.8875
## 5        PctRecImmig8  476.7561
```

```
## 6         FemalePctDiv  334.1784
## 7         PctRecImmig5  312.3608
## 8        PctRecImmig10  301.3442
## 9           population  289.5185
## 10           numbUrban  280.6762
## 11      OwnOccLowQuart  235.9103
## 12   PctLargHouseOccup  232.7465
## 13      MalePctDivorce  231.5842
## 14     PctLargHouseFam  225.7866
## 15    PersPerOccupHous  204.7054
## 16       OwnOccHiQuart  170.9826
## 17           perCapInc  148.1097
## 18           medIncome  146.6833
## 19          RentMedian  122.5943
## 20          PctFam2Par  117.2577
## 21         PctKids2Par  116.4745
## 22           medFamInc  113.9526
## 23      PctRecentImmig   94.7685
## 24         whitePerCap   91.7120
## 25             MedRent   87.0486
## 26         agePct16t24   84.9199
## 27   PersPerOwnOccHous   79.4734
## 28          PersPerFam   77.2207
## 29         agePct12t29   57.5117
## 30           RentHighQ   52.1989
## 31      PctForeignBorn   49.1970
## 32        PctNotHSGrad   42.2868
## 33          pctWSocSec   39.2326
## 34          agePct65up   39.2197
## 35         NumUnderPov   35.6325
## 36         agePct12t21   30.6353
## 37            pctWWage   30.4703
## 38    PctSpeakEnglOnly   28.9595
## 39    PctPersDenseHous   28.4373
## 40        PctImmigRec8   27.5908
## 41  PersPerRentOccHous   26.7597
## 42 PctNotSpeakEnglWell   25.5983
## 43            RentLowQ   24.6763
## 44     PctLess9thGrade   23.8918
## 45        racePctWhite   23.2863
## 46      PctPopUnderPov   23.2849
## 47       householdsize   22.6418
## 48        PctImmigRec5   22.3845
## 49           PctEmploy   21.4108
## 50        racepctblack   19.0848
## 51         PctBSorMore   18.8546
## 52         racePctHisp   17.6180
## 53          pctWInvInc   16.4477
## 54      MalePctNevMarr   16.1108
## 55            NumIlleg   15.7449
## 56       PctImmigRec10   15.4596
## 57          HousVacant   13.7280
## 58            PctIlleg   13.5013
## 59    PctYoungKids2Par   12.6732
```

```
## 60      PctSameHouse85   12.4428
## 61         pctWPubAsst   11.8678
## 62      PctHousLess3BR   11.6198
## 63          PctWorkMom   10.1676
```

In this block, we retained those variables whose vif score was less than 10

```r
ViolentCrimesPerPop <- Communities_Crime_Train$ViolentCrimesPerPop
```

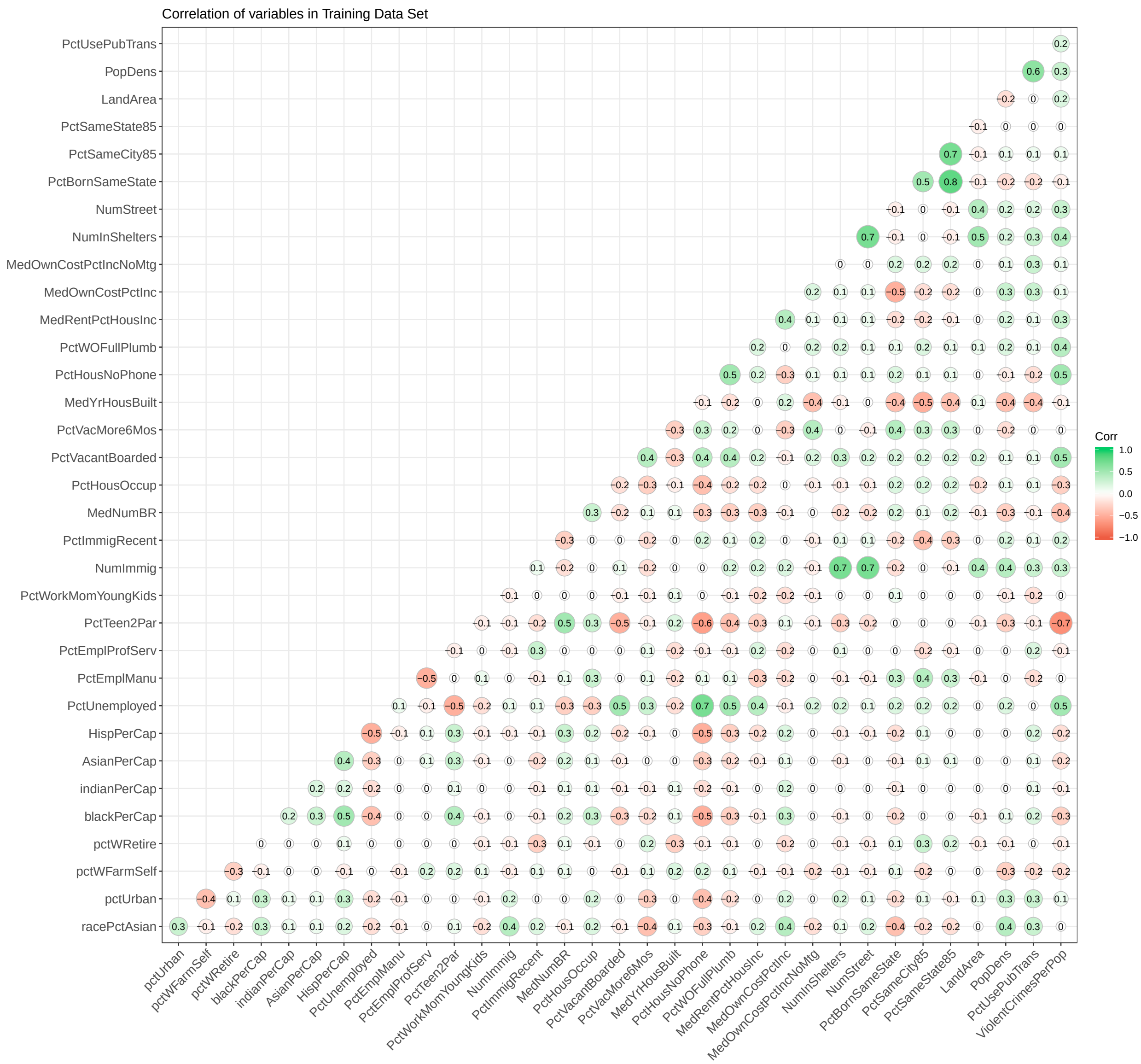
```r
retain <- multicoll$vars[multicoll$scores<10]
retain <- str_replace(retain, "`", "")
retain <- str_replace(retain, "`", "")
Communities_Crime_Train_Cleaned  <- subset(Communities_Crime_Train, select=c(retain))
Communities_Crime_Train_Cleaned <- cbind(Communities_Crime_Train_Cleaned, ViolentCrimesPerPop)
dim(Communities_Crime_Train_Cleaned)
```

```
## [1] 1994   34
```

We now see that by removing the correlated variables reduces their numbers from 100 to 34.

The correlation plot below shows little to no correlation between the attributes.

```r
# Look at correlation between variables

corr <- round(cor(Communities_Crime_Train_Cleaned), 1)

ggcorrplot(corr,
           type="lower",
           lab=TRUE,
           lab_size=3,
           method="circle",
           colors=c("tomato2", "white", "springgreen3"),
           title="Correlation of variables in Training Data Set",
           ggtheme=theme_bw)
```



D. Check for Normality

Below, we update the metastats dataframe for the cleaned dataset, Communities_Crime_Train_Cleaned2.

```
metastats <- data.frame(psych::describe(Communities_Crime_Train_Cleaned))
metastats <- tibble::rownames_to_column(metastats, "attributes")
metastats["pct_complete"] <- round(metastats["n"]/dimensions[1], 3)
metastats$attributes <- gsub('\\*', '', metastats$attributes)
metastats$variance <- (metastats$sd)^2
head(metastats,10)
```

```
##       attributes vars    n      mean        sd median   trimmed      mad min
## 1   racePctAsian    1 1994 0.1536810 0.2088775  0.070 0.1042857 0.074130   0
## 2       pctUrban    2 1994 0.6962688 0.4448105  1.000 0.7452130 0.000000   0
## 3   pctWFarmSelf    3 1994 0.2915697 0.2041076  0.230 0.2605263 0.148260   0
```

```
## 4     pctWRetire    4 1994 0.4792477 0.1675637  0.470 0.4721115 0.163086   0
## 5    blackPerCap    5 1994 0.2910983 0.1715934  0.250 0.2716416 0.133434   0
## 6   indianPerCap    6 1994 0.2035055 0.1647754  0.170 0.1812281 0.103782   0
## 7    AsianPerCap    7 1994 0.3223571 0.1954109  0.280 0.2988784 0.148260   0
## 8     HispPerCap    8 1994 0.3862788 0.1830806  0.345 0.3648622 0.155673   0
## 9  PctUnemployed    9 1994 0.3635306 0.2021713  0.320 0.3435714 0.192738   0
## 10   PctEmplManu   10 1994 0.3963842 0.2023860  0.370 0.3817669 0.192738   0
##    max range       skew    kurtosis          se pct_complete   variance
## 1    1     1  2.6004776  6.78382881 0.004677664            1 0.04362979
## 2    1     1 -0.8843025 -1.16518094 0.009961219            1 0.19785642
## 3    1     1  1.5354725  2.41033058 0.004570846            1 0.04165990
## 4    1     1  0.4513799  0.34643620 0.003752472            1 0.02807761
## 5    1     1  1.3459464  2.67688360 0.003842714            1 0.02944430
## 6    1     1  2.0771638  6.31220433 0.003690029            1 0.02715093
## 7    1     1  1.2913186  2.06365276 0.004376089            1 0.03818541
## 8    1     1  1.1822894  1.59864716 0.004099961            1 0.03351851
## 9    1     1  0.9329714  0.72786686 0.004527484            1 0.04087323
## 10   1     1  0.6512525  0.07293686 0.004532291            1 0.04096008
```

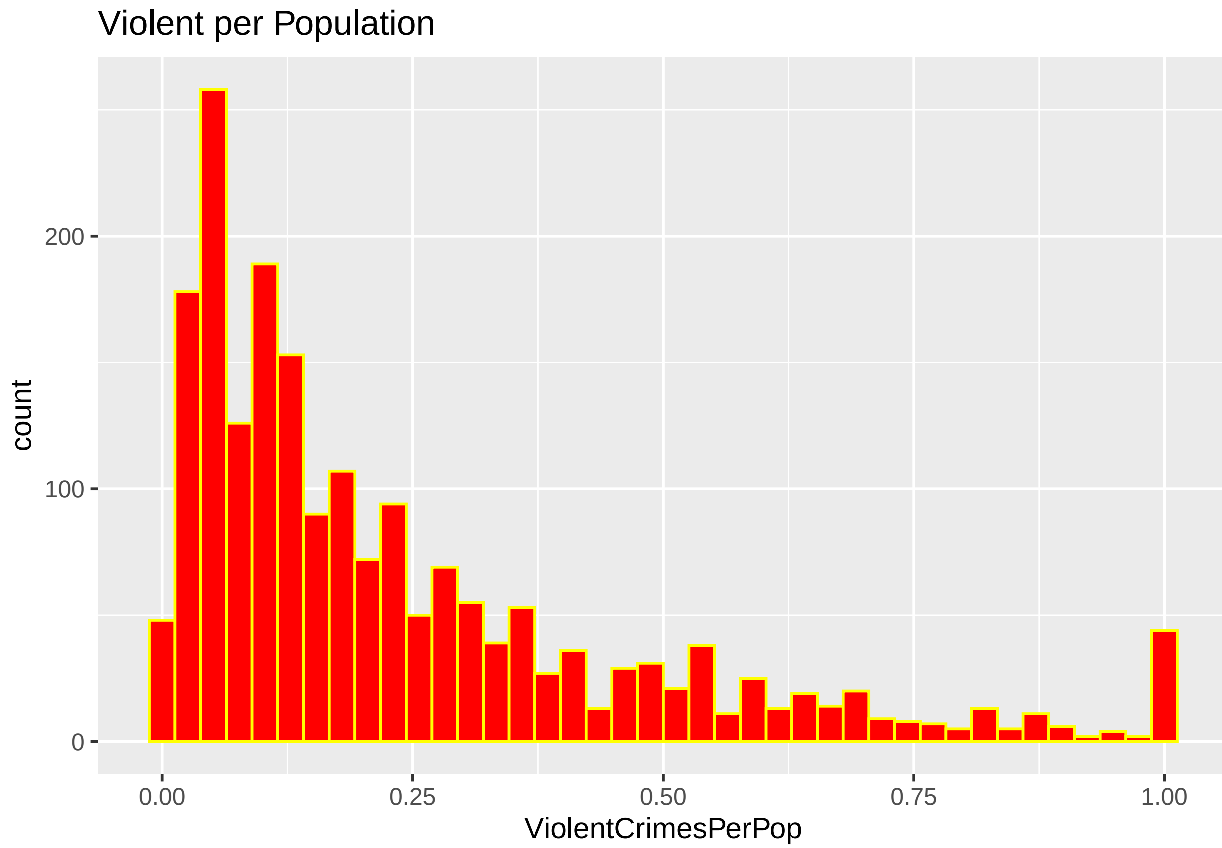
First, we determine and remove from the dataset those variables that have a zero variance, and we found that none of the variables have a zero variance.

```r
removeZeroVariance <- metastats$variables[metastats$variance < 1]
removeZeroVariance
```

```
## NULL
```

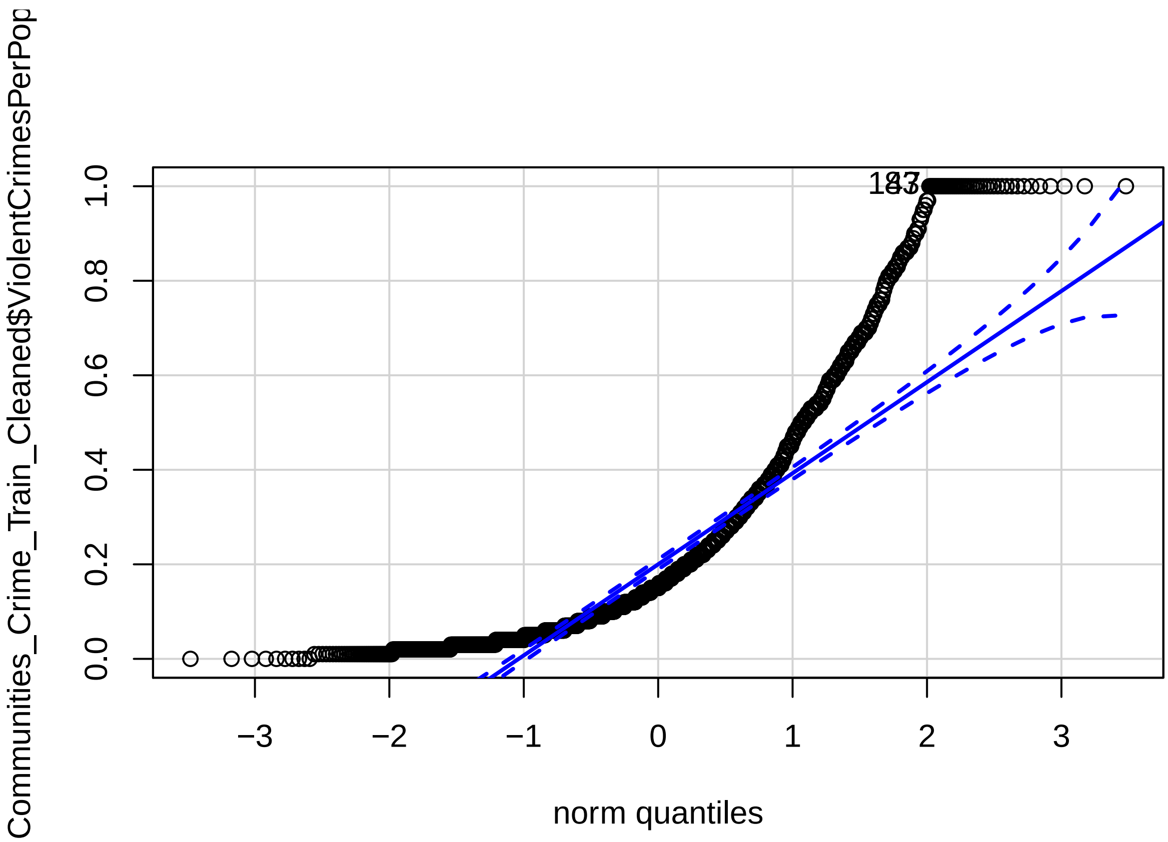
Let's look at some interesting part of the data by exploring the dependent variables: nonViolPerPop and violentPerPop

```r
p2<-ggplot(Communities_Crime_Train_Cleaned, aes(x=ViolentCrimesPerPop)) +
  geom_histogram(color="yellow", fill="red", bins=40) +
  ggtitle("Violent per Population")

p2
```



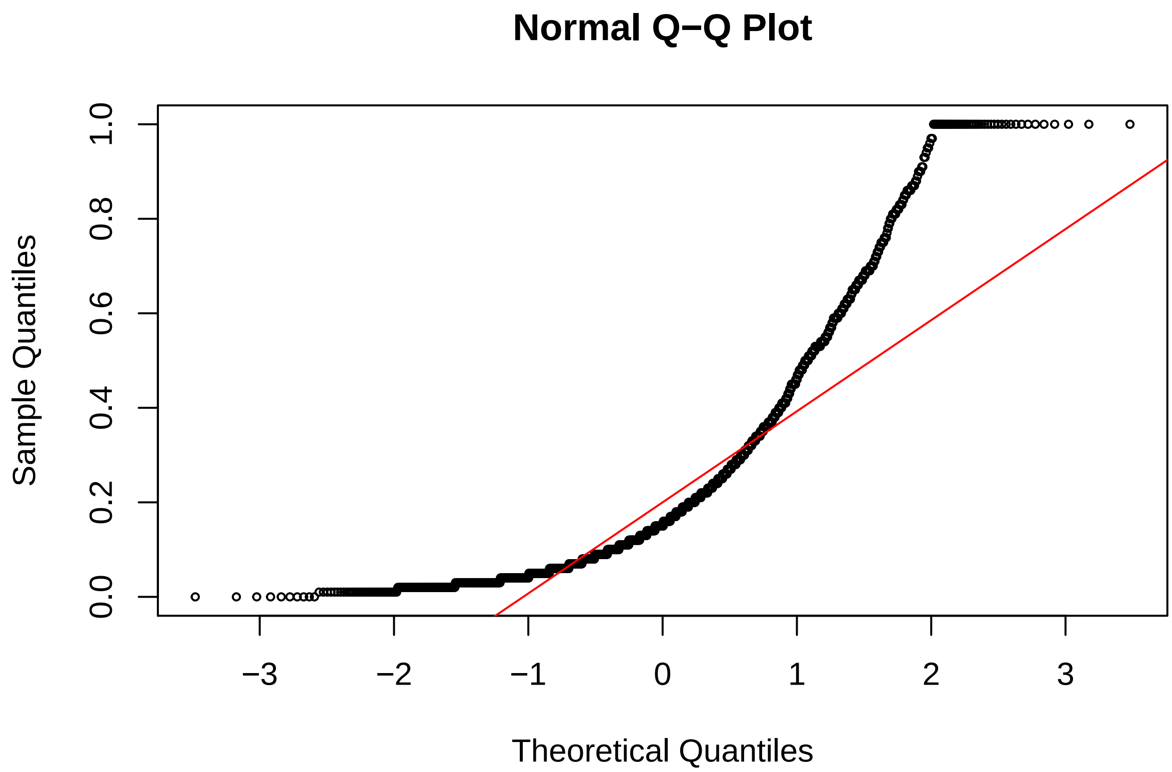
We see that ViolentCrimesPerPop is negatively skewed.

```
qqPlot(Communities_Crime_Train_Cleaned$ViolentCrimesPerPop)
```



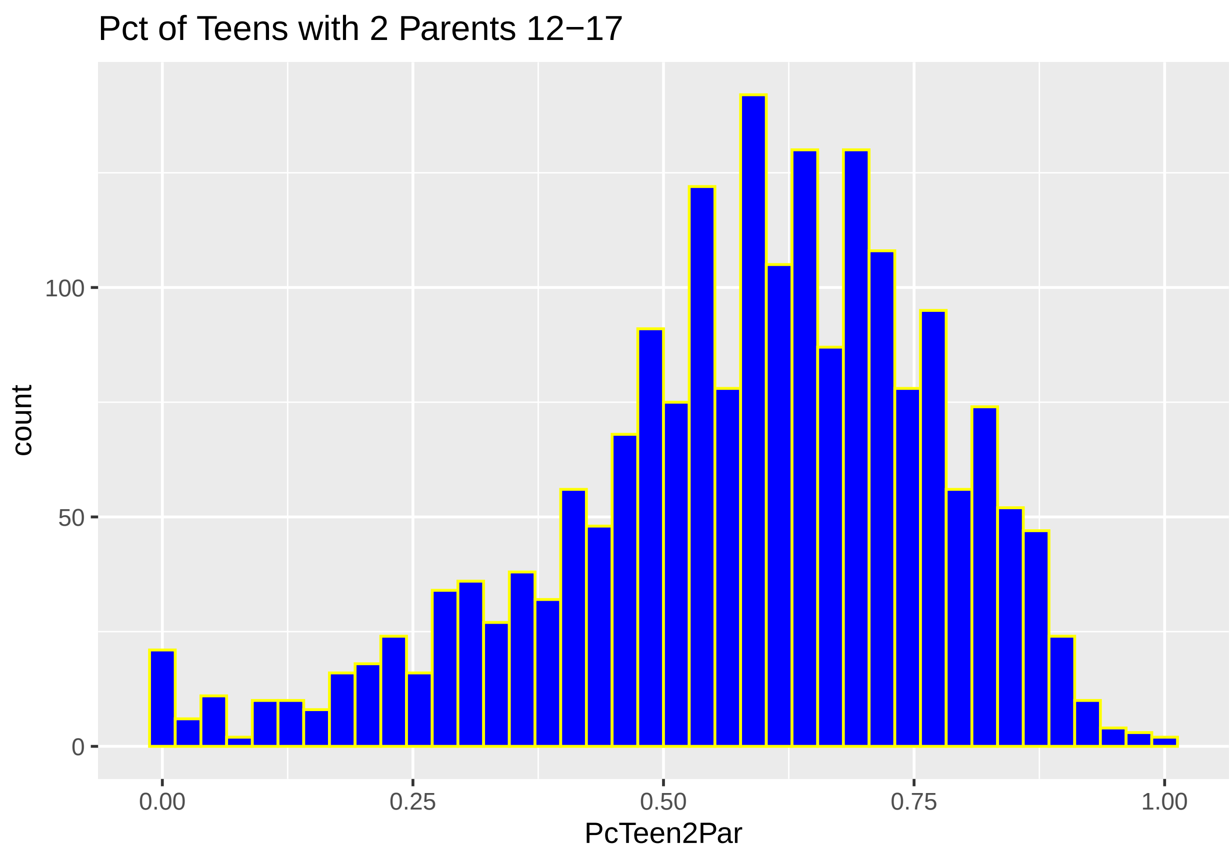

```
## [1]  83 147
```

```r
qqnorm(Communities_Crime_Train_Cleaned$ViolentCrimesPerPop,pch = 1, cex = 0.5)
qqline(Communities_Crime_Train_Cleaned$ViolentCrimesPerPop, col = "red", lwd = 1)
```



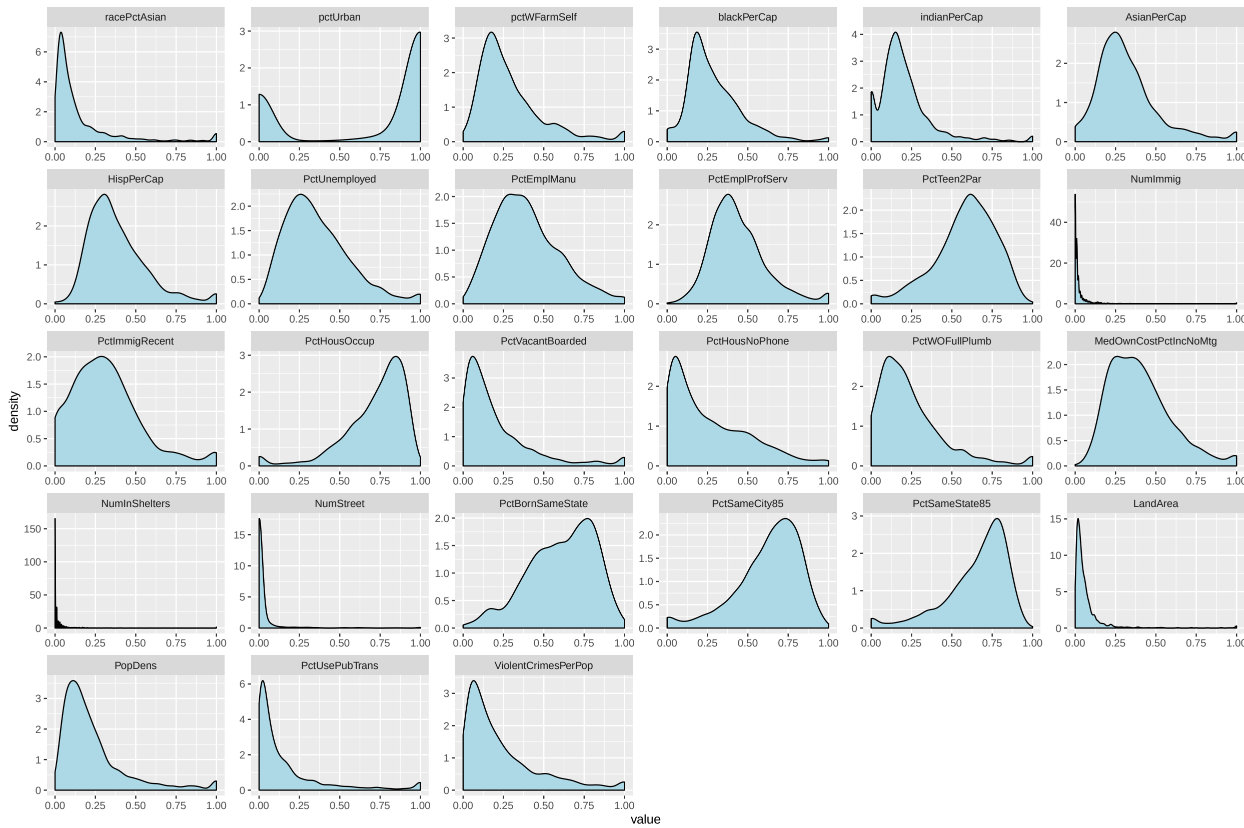
A look at a skewed variable

```
PcTeen2Par <- Communities_Crime_Train_Cleaned$PctTeen2Par
v1 <-ggplot(Communities_Crime_Train_Cleaned, aes(x=PcTeen2Par)) +
  geom_histogram(color="yellow", fill="blue", bins=40) +
  ggtitle("Pct of Teens with 2 Parents 12-17")


v1
```




```
skewedVars <- metastats %>% dplyr::select(1,12,13) %>% filter(skew > .5 | skew < -.5)
correctSkew <- skewedVars$attributes
NormalizedVars <- Communities_Crime_Train_Cleaned[correctSkew]
```

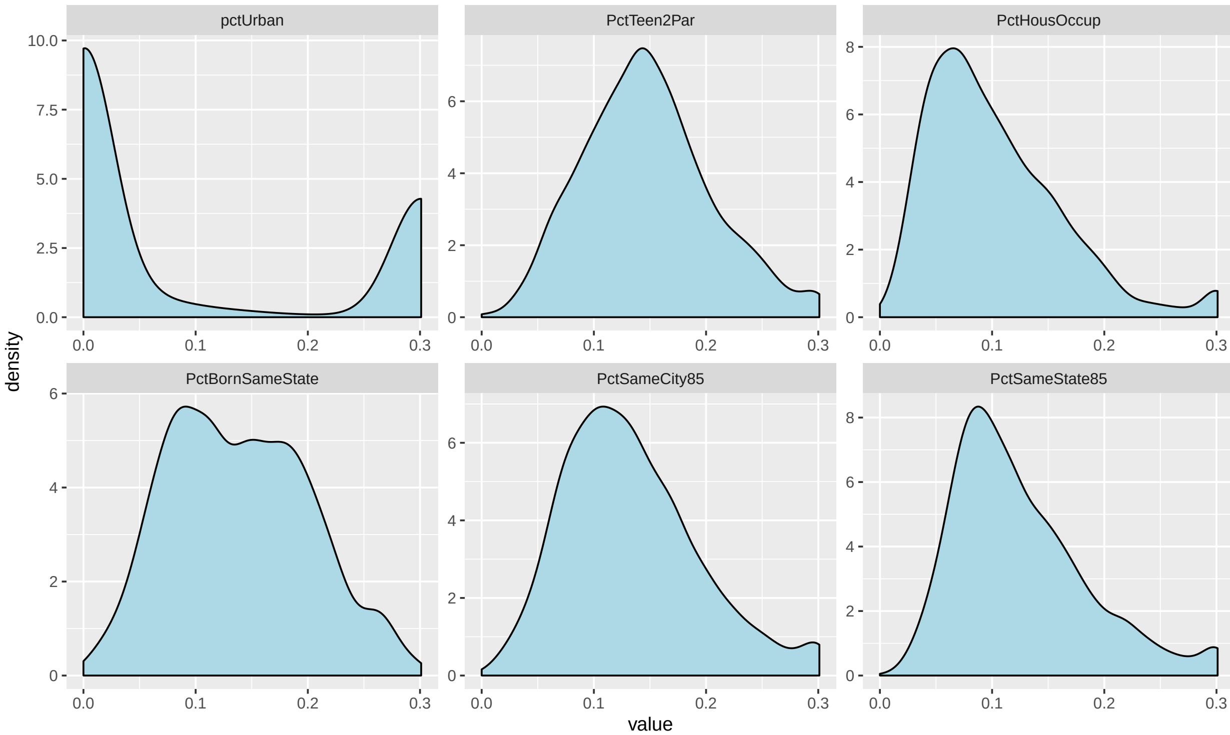
```
par(mfrow = c(3, 3))
datasub = melt(NormalizedVars)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='lightblue') + facet_wrap(~variable, scales = 'free') )
```



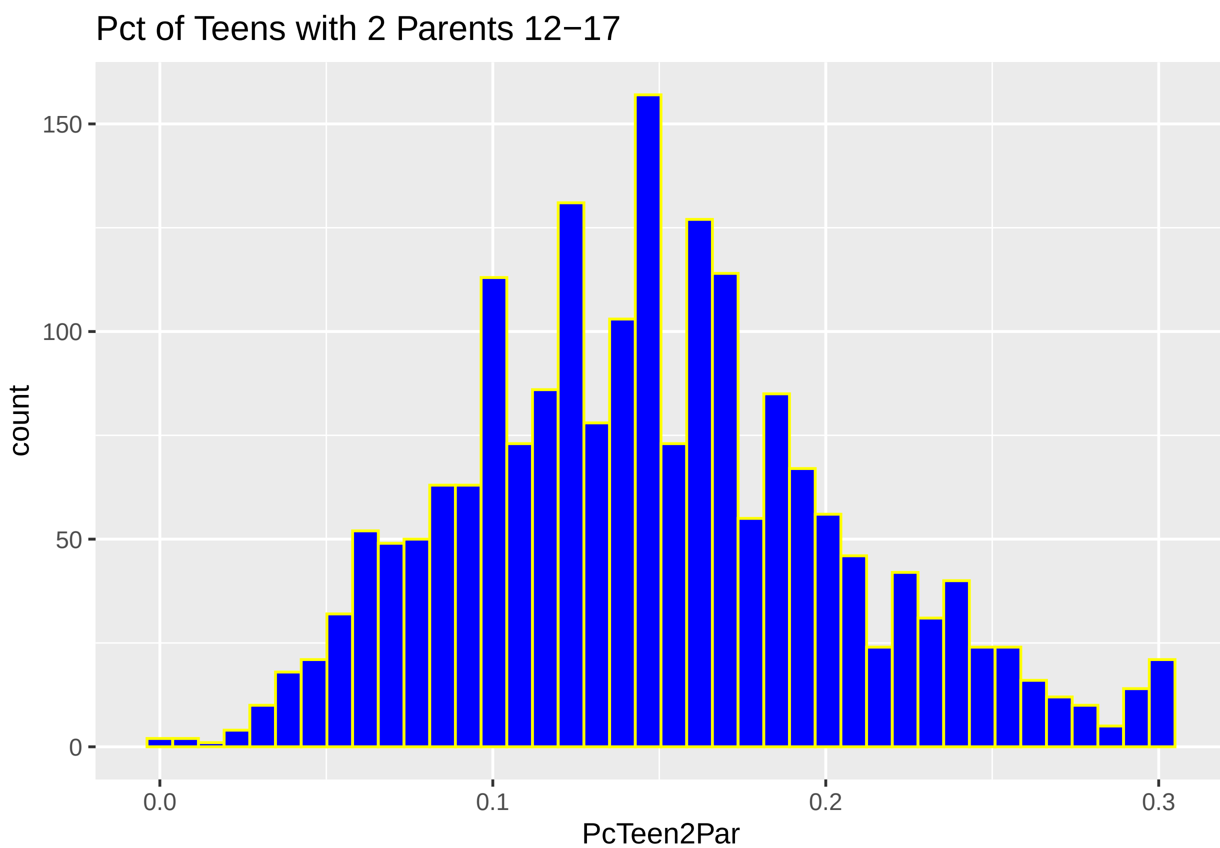

```
negTransform <- function(x){
      x <- log10(max(x+1) - x)
        return (x)
}
```

```
neg_skewedVars <- metastats %>% dplyr::select(1,12,13) %>% filter(skew < -.5)
correctNegSkew <- neg_skewedVars$attributes
Normalized_Neg_Vars <- Communities_Crime_Train_Cleaned[correctNegSkew]


Normalized_Neg_Vars <- as.data.frame(apply(Normalized_Neg_Vars,2,negTransform))
```

```
par(mfrow = c(3, 3))
datasub = melt(Normalized_Neg_Vars)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='lightblue') + facet_wrap(~variable, scales = 'free') )
```




```r
PcTeen2Par <- Normalized_Neg_Vars$PctTeen2Par
v1 <-ggplot(Communities_Crime_Train_Cleaned, aes(x=PcTeen2Par)) +
  geom_histogram(color="yellow", fill="blue", bins=40) +
  ggtitle("Pct of Teens with 2 Parents 12-17")


v1
```

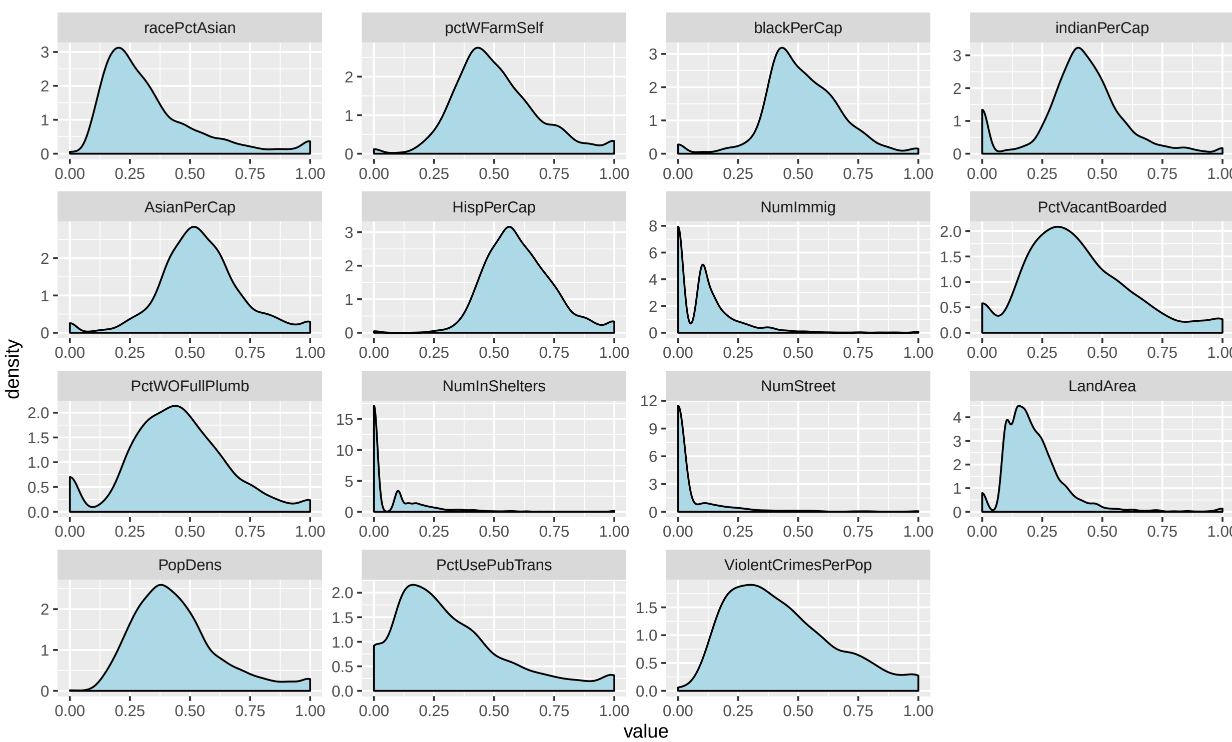



```
pos_skewedVars <- metastats %>% dplyr::select(1,12,13) %>% filter(skew > 1)
correctPosskew <- pos_skewedVars$attributes
Normalized_Pos_Vars <- Communities_Crime_Train_Cleaned[correctPosskew]


Normalized_Pos_Vars <- as.data.frame(apply(Normalized_Pos_Vars,2,sqrt))

head(Normalized_Pos_Vars,10)
```

```
##    racePctAsian pctWFarmSelf blackPerCap indianPerCap AsianPerCap HispPerCap
## 1     0.3464102    0.5830952   0.5656854    0.5196152   0.5196152  0.6403124
## 2     0.6708204    0.3316625   0.5744563    0.4000000   0.5477226  0.5916080
## 3     0.4123106    0.4358899   0.5196152    0.2645751   0.5385165  0.6244998
## 4     0.3464102    0.4582576   0.6244998    0.4000000   0.5000000  0.6633250
## 5     0.3000000    0.4000000   0.5291503    0.0000000   0.8602325  0.6928203
## 6     1.0000000    0.4472136   0.8774964    0.5291503   0.7211103  0.7745967
## 7     0.2449490    0.4795832   0.6324555    0.4898979   0.9273618  0.6000000
## 8     0.4472136    1.0000000   0.2828427    0.4123106   0.5196152  0.4582576
## 9     0.1414214    0.6000000   0.4358899    0.3162278   0.5099020  0.4690416
## 10    0.5477226    0.4690416   0.3316625    0.3000000   0.5744563  0.8944272
##      NumImmig PctVacantBoarded PctWOFullPlumb NumInShelters NumStreet  LandArea
## 1  0.1732051        0.2236068      0.2449490           0.2         0 0.3464102
## 2  0.1000000        0.1414214      0.0000000           0.0         0 0.1414214
## 3  0.0000000        0.5385165      0.6708204           0.0         0 0.1000000
## 4  0.1414214        0.7745967      0.3316625           0.0         0 0.1414214
## 5  0.0000000        0.2000000      0.3741657           0.0         0 0.2000000
```

```
## 6  0.2000000        0.4000000      0.2236068          0.0          0 0.1000000
## 7  0.1000000        0.3000000      0.2236068          0.0          0 0.2236068
## 8  0.1414214        0.4690416      0.4795832          0.0          0 0.1000000
## 9  0.0000000        0.2236068      0.4690416          0.1          0 0.2000000
## 10 0.1000000        0.2645751      0.0000000          0.0          0 0.0000000
##      PopDens PctUsePubTrans ViolentCrimesPerPop
## 1  0.5099020      0.4472136           0.4472136
## 2  0.3464102      0.6708204           0.8185353
## 3  0.4582576      0.1414214           0.6557439
## 4  0.6244998      0.5291503           0.3464102
## 5  0.3000000      0.1414214           0.1732051
## 6  0.7615773      0.3162278           0.3741657
## 7  0.2828427      0.2449490           0.1732051
## 8  0.5744563      0.0000000           0.7416198
## 9  0.4123106      0.2000000           0.7280110
## 10 0.6855655      0.3316625           0.3872983
```

```
par(mfrow = c(3, 3))
datasub = melt(Normalized_Pos_Vars)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='lightblue') + facet_wrap(~variable, scales = 'free') )
```



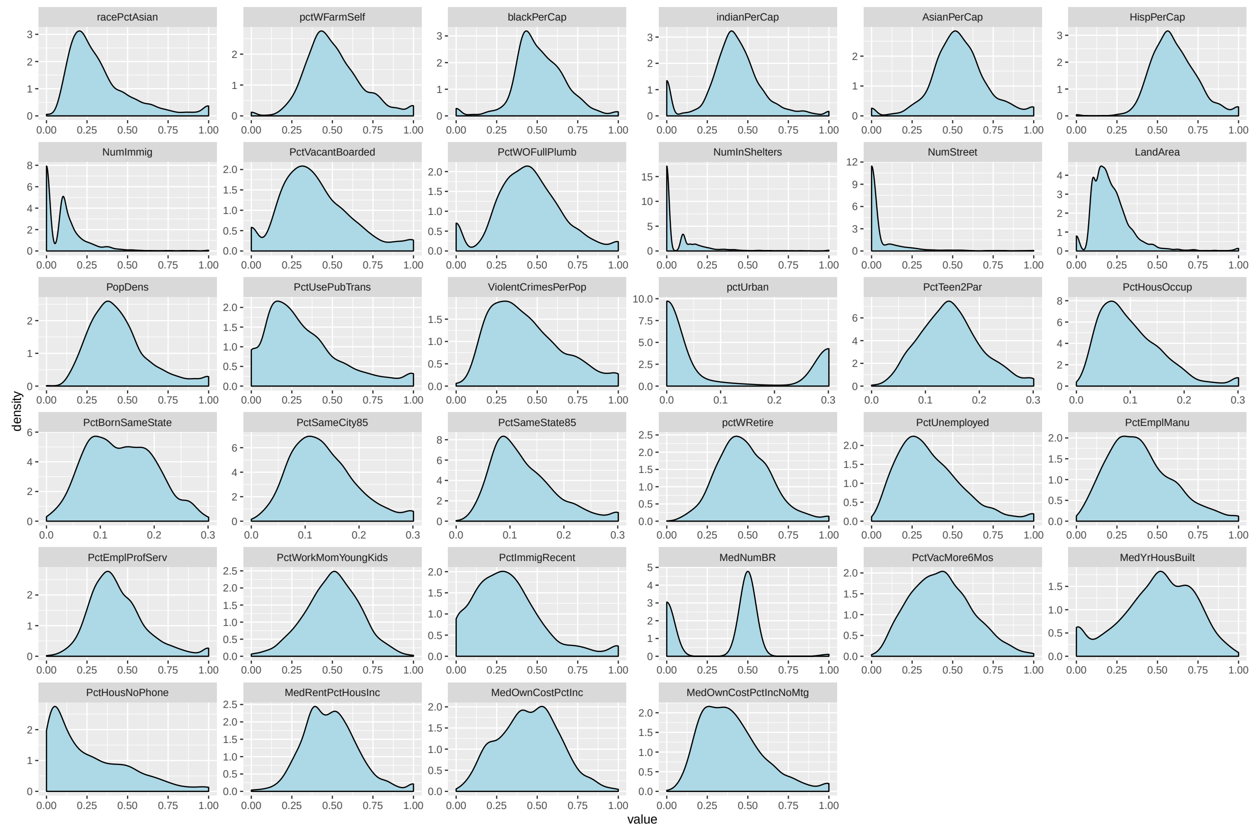

```
normalizedVars <- c(colnames(Normalized_Pos_Vars), colnames(Normalized_Neg_Vars))
Communities_Crime_Train_Cleaned[normalizedVars] <- NULL
Communities_Crime_Train_Cleaned_Normalized <- cbind(Normalized_Pos_Vars,Normalized_Neg_Vars,Communities_
```

```
head(Communities_Crime_Train_Cleaned_Normalized)
```

```
##   racePctAsian pctWFarmSelf blackPerCap indianPerCap AsianPerCap HispPerCap
## 1    0.3464102    0.5830952   0.5656854    0.5196152   0.5196152  0.6403124
```

```
## 2    0.6708204    0.3316625   0.5744563    0.4000000   0.5477226  0.5916080
## 3    0.4123106    0.4358899   0.5196152    0.2645751   0.5385165  0.6244998
## 4    0.3464102    0.4582576   0.6244998    0.4000000   0.5000000  0.6633250
## 5    0.3000000    0.4000000   0.5291503    0.0000000   0.8602325  0.6928203
## 6    1.0000000    0.4472136   0.8774964    0.5291503   0.7211103  0.7745967
##    NumImmig PctVacantBoarded PctWOFullPlumb NumInShelters NumStreet  LandArea
## 1 0.1732051        0.2236068      0.2449490           0.2         0 0.3464102
## 2 0.1000000        0.1414214      0.0000000           0.0         0 0.1414214
## 3 0.0000000        0.5385165      0.6708204           0.0         0 0.1000000
## 4 0.1414214        0.7745967      0.3316625           0.0         0 0.1414214
## 5 0.0000000        0.2000000      0.3741657           0.0         0 0.2000000
## 6 0.2000000        0.4000000      0.2236068           0.0         0 0.1000000
##     PopDens PctUsePubTrans ViolentCrimesPerPop   pctUrban PctTeen2Par
## 1 0.5099020      0.4472136           0.4472136 0.00000000  0.15836249
## 2 0.3464102      0.6708204           0.8185353 0.00000000  0.20682588
## 3 0.4582576      0.1414214           0.6557439 0.30103000  0.19589965
## 4 0.6244998      0.5291503           0.3464102 0.00000000  0.13033377
## 5 0.3000000      0.1414214           0.1732051 0.04139269  0.06069784
## 6 0.7615773      0.3162278           0.3741657 0.00000000  0.16731733
##   PctHousOccup PctBornSameState PctSameCity85 PctSameState85 pctWRetire
## 1   0.11058971        0.1986571     0.1731863      0.1335389       0.43
## 2   0.08278537        0.1760913     0.1461280      0.1702617       0.39
## 3   0.05690485        0.1789769     0.1238516      0.1583625       0.84
## 4   0.01283722        0.2304489     0.1335389      0.1303338       0.82
## 5   0.04532298        0.1072100     0.1430148      0.1673173       0.71
## 6   0.06445799        0.1986571     0.1038037      0.1335389       0.25
##   PctUnemployed PctEmplManu PctEmplProfServ PctWorkMomYoungKids PctImmigRecent
## 1          0.27        0.23            0.41                0.74           0.24
## 2          0.27        0.57            0.15                0.46           0.52
## 3          0.36        0.32            0.29                0.71           0.07
## 4          0.33        0.36            0.45                0.85           0.11
## 5          0.12        0.67            0.38                0.40           0.03
## 6          0.10        0.19            0.77                0.30           0.30
##   MedNumBR PctVacMore6Mos MedYrHousBuilt PctHousNoPhone MedRentPctHousInc
## 1      0.5           0.26           0.65           0.14              0.38
## 2      0.0           0.25           0.65           0.16              0.29
## 3      0.5           0.30           0.52           0.47              0.48
## 4      0.5           0.47           0.52           0.11              0.63
## 5      0.5           0.55           0.73           0.05              0.22
## 6      0.0           0.28           0.25           0.02              0.47
##   MedOwnCostPctInc MedOwnCostPctIncNoMtg
## 1             0.46                  0.25
## 2             0.32                  0.18
## 3             0.39                  0.28
## 4             0.51                  0.47
## 5             0.51                  0.21
## 6             0.59                  0.11
```

```
par(mfrow = c(3, 3))
datasub = melt(Communities_Crime_Train_Cleaned_Normalized)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='lightblue') + facet_wrap(~variable, scales = 'free') )
```




```r
metastats <- data.frame(psych::describe(Communities_Crime_Train_Cleaned_Normalized))
metastats <- tibble::rownames_to_column(metastats, "attributes")
metastats["pct_complete"] <- round(metastats["n"]/dimensions[1], 3)
metastats$attributes <- gsub('\\*', '', metastats$attributes)
metastats$variance <- (metastats$sd)^2
metastats
```

```
##                attributes vars    n       mean         sd     median    trimmed
## 1            racePctAsian    1 1994 0.33322380 0.20655347 0.26457513 0.30003785
## 2            pctWFarmSelf    2 1994 0.50981878 0.17796180 0.47958315 0.49845608
## 3             blackPerCap    3 1994 0.51456733 0.16227126 0.50000000 0.51274786
## 4            indianPerCap    4 1994 0.41019760 0.18777948 0.41231056 0.41560478
## 5             AsianPerCap    5 1994 0.54014789 0.17496481 0.52915026 0.53794500
## 6              HispPerCap    6 1994 0.60486891 0.14290792 0.58735158 0.59739864
## 7                NumImmig    7 1994 0.10757349 0.13600515 0.10000000 0.08282317
## 8        PctVacantBoarded    8 1994 0.39362725 0.22273540 0.36055513 0.37898894
## 9          PctWOFullPlumb    9 1994 0.44489269 0.21249074 0.43588989 0.44322446
## 10          NumInShelters   10 1994 0.08238139 0.15054231 0.00000000 0.04794711
## 11              NumStreet   11 1994 0.05281319 0.14141829 0.00000000 0.01559634
## 12               LandArea   12 1994 0.21695904 0.13479087 0.20000000 0.19938244
## 13                PopDens   13 1994 0.44587414 0.18457220 0.41231056 0.42669313
## 14         PctUsePubTrans   14 1994 0.32027721 0.24318146 0.26457513 0.29185892
## 15    ViolentCrimesPerPop   15 1994 0.43452016 0.22180135 0.38729833 0.41543398
## 16               pctUrban   16 1994 0.09300138 0.13388312 0.00000000 0.07865901
## 17            PctTeen2Par   17 1994 0.14759222 0.05711711 0.14301480 0.14524844
## 18           PctHousOccup   18 1994 0.10292546 0.06047877 0.08990511 0.09601114
## 19       PctBornSameState   19 1994 0.13880033 0.06262982 0.13672057 0.13681403
## 20          PctSameCity85   20 1994 0.13353008 0.06038028 0.12385164 0.12908024
```

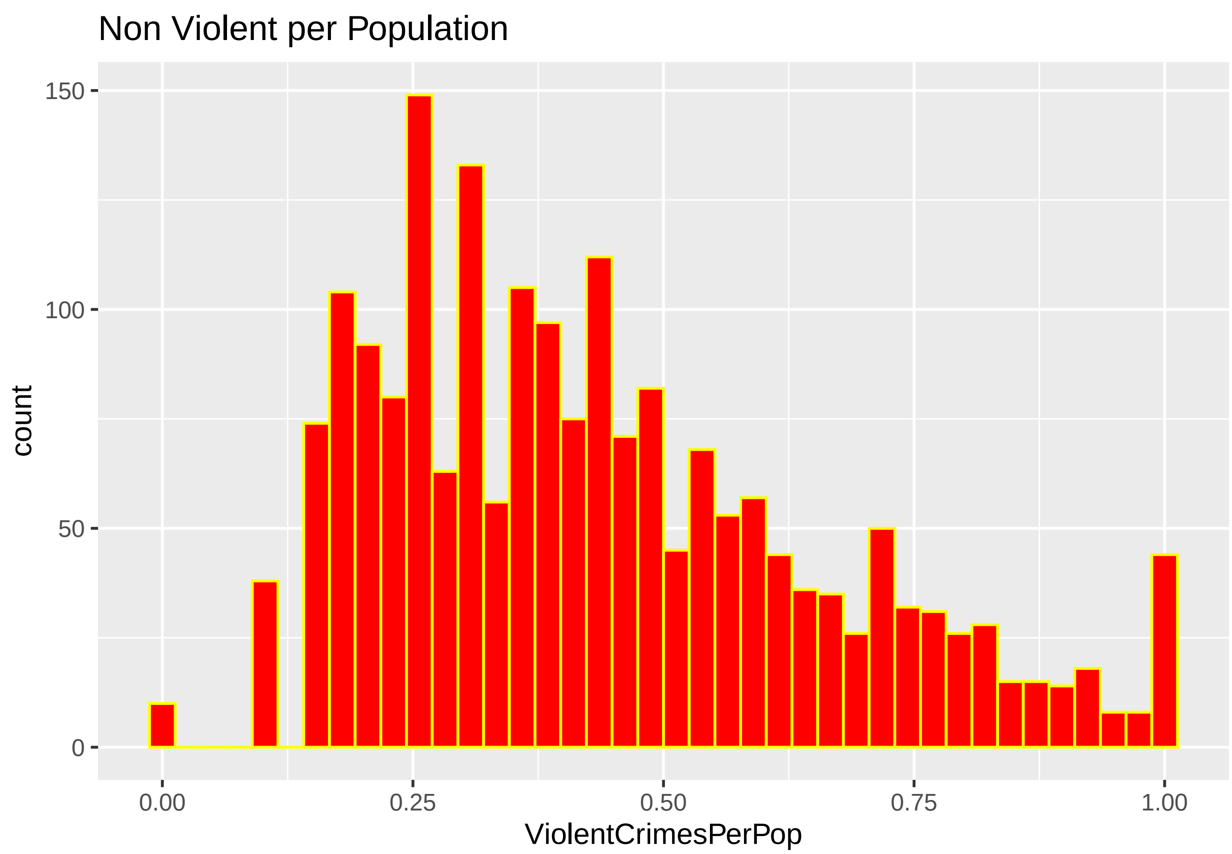
```
## 21        PctSameState85   21 1994 0.12556696 0.05966055 0.11394335 0.11929706
## 22            pctWRetire   22 1994 0.47924774 0.16756374 0.47000000 0.47211153
## 23         PctUnemployed   23 1994 0.36353059 0.20217129 0.32000000 0.34357143
## 24           PctEmplManu   24 1994 0.39638415 0.20238596 0.37000000 0.38176692
## 25       PctEmplProfServ   25 1994 0.44059679 0.17545695 0.41000000 0.42471178
## 26   PctWorkMomYoungKids   26 1994 0.50144935 0.16861157 0.51000000 0.50380326
## 27        PctImmigRecent   27 1994 0.32021063 0.21908846 0.29000000 0.29869048
## 28              MedNumBR   28 1994 0.31469408 0.25518159 0.50000000 0.32236842
## 29        PctVacMore6Mos   29 1994 0.43333501 0.18898562 0.42000000 0.42544486
## 30        MedYrHousBuilt   30 1994 0.49417753 0.23246676 0.52000000 0.50730576
## 31        PctHousNoPhone   31 1994 0.26447844 0.24284700 0.18500000 0.23221178
## 32     MedRentPctHousInc   32 1994 0.49012538 0.16949982 0.48000000 0.48245614
## 33      MedOwnCostPctInc   33 1994 0.44975426 0.18727370 0.45000000 0.44735589
## 34 MedOwnCostPctIncNoMtg   34 1994 0.40381645 0.19259349 0.37000000 0.38520050
##           mad min     max   range        skew     kurtosis          se
## 1  0.13546524   0 1.00000 1.00000  1.48321520  1.979757973 0.004625620
## 2  0.15629186   0 1.00000 1.00000  0.53024871  0.625662240 0.003985329
## 3  0.13581799   0 1.00000 1.00000 -0.10369042  1.556687200 0.003633951
## 4  0.13000836   0 1.00000 1.00000 -0.05136480  1.195855520 0.004205189
## 5  0.14136522   0 1.00000 1.00000 -0.04787307  1.213030989 0.003918214
## 6  0.12950746   0 1.00000 1.00000  0.33825610  0.972602528 0.003200322
## 7  0.14826000   0 1.00000 1.00000  2.46466217  9.907063695 0.003045740
## 8  0.20303959   0 1.00000 1.00000  0.63219804  0.302199324 0.004988003
## 9  0.18583062   0 1.00000 1.00000  0.12323957  0.305389032 0.004758581
## 10 0.00000000   0 1.00000 1.00000  2.89429479 10.954347790 0.003371289
## 11 0.00000000   0 1.00000 1.00000  3.80391327 16.949073982 0.003166963
## 12 0.08684870   0 1.00000 1.00000  2.29074269  9.170374514 0.003018547
## 13 0.15908992   0 1.00000 1.00000  0.96550026  0.860823902 0.004133365
## 14 0.20078091   0 1.00000 1.00000  1.03219922  0.621204642 0.005445878
## 15 0.21104716   0 1.00000 1.00000  0.67567480 -0.222373750 0.004967085
## 16 0.00000000   0 0.30103 0.30103  0.85633567 -1.201286862 0.002998218
## 17 0.05544252   0 0.30103 0.30103  0.36374225 -0.098976754 0.001279097
## 18 0.05515226   0 0.30103 0.30103  1.12602563  1.277068513 0.001354379
## 19 0.07112160   0 0.30103 0.30103  0.23991053 -0.634601866 0.001402551
## 20 0.06004710   0 0.30103 0.30103  0.65068272  0.151011747 0.001352174
## 21 0.05229984   0 0.30103 0.30103  0.93561206  0.538581438 0.001336056
## 22 0.16308600   0 1.00000 1.00000  0.45137993  0.346436199 0.003752472
## 23 0.19273800   0 1.00000 1.00000  0.93297144  0.727866856 0.004527484
## 24 0.19273800   0 1.00000 1.00000  0.65125254  0.072936862 0.004532291
## 25 0.14826000   0 1.00000 1.00000  0.92188633  1.124272998 0.003929235
## 26 0.16308600   0 1.00000 1.00000 -0.13022143 -0.003910067 0.003775937
## 27 0.19273800   0 1.00000 1.00000  0.97447833  1.054591312 0.004906332
## 28 0.00000000   0 1.00000 1.00000 -0.22757013 -1.239673044 0.005714612
## 29 0.19273800   0 1.00000 1.00000  0.37390418 -0.257035236 0.004232200
## 30 0.22239000   0 1.00000 1.00000 -0.41893256 -0.406940772 0.005205929
## 31 0.21497700   0 1.00000 1.00000  0.99213316  0.208717391 0.005438388
## 32 0.16308600   0 1.00000 1.00000  0.46420600  0.516144636 0.003795829
## 33 0.19273800   0 1.00000 1.00000  0.11383203 -0.393774287 0.004193863
## 34 0.17791200   0 1.00000 1.00000  0.88392723  0.622006589 0.004312996
##    pct_complete    variance
## 1             1 0.042664337
## 2             1 0.031670401
## 3             1 0.026331961
## 4             1 0.035261132
```

```
## 5              1 0.030612685
## 6              1 0.020422675
## 7              1 0.018497401
## 8              1 0.049611057
## 9              1 0.045152313
## 10             1 0.022662987
## 11             1 0.019999132
## 12             1 0.018168579
## 13             1 0.034066897
## 14             1 0.059137222
## 15             1 0.049195838
## 16             1 0.017924689
## 17             1 0.003262364
## 18             1 0.003657681
## 19             1 0.003922494
## 20             1 0.003645779
## 21             1 0.003559382
## 22             1 0.028077607
## 23             1 0.040873229
## 24             1 0.040960075
## 25             1 0.030785143
## 26             1 0.028429860
## 27             1 0.047999755
## 28             1 0.065117643
## 29             1 0.035715566
## 30             1 0.054040793
## 31             1 0.058974665
## 32             1 0.028730190
## 33             1 0.035071440
## 34             1 0.037092251
```

```
missing_vaues <- metastats[metastats$pct_complete < 1,]
missing_vaues[order(missing_vaues$pct_complete),]
```

```
##  [1] attributes   vars         n            mean         sd
##  [6] median       trimmed      mad          min          max
## [11] range        skew         kurtosis     se           pct_complete
## [16] variance
## <0 rows> (or 0-length row.names)
```

```
p4<-ggplot(Communities_Crime_Train_Cleaned_Normalized, aes(x=ViolentCrimesPerPop)) +
  geom_histogram(color="yellow", fill="red", bins=40) +
  ggtitle("Non Violent per Population")

p4
```



E. Split the Communities_Crime_Train dataset into Train and Test sets for Non Violent and Violent Crime

Violent Crime Split

```
set.seed(1234)
train <- createDataPartition(y = Communities_Crime_Train_Cleaned_Normalized$ViolentCrimesPerPop, p = 0.8
VC_train <- na.omit(Communities_Crime_Train_Cleaned_Normalized[train,])
VC_test  <- na.omit(Communities_Crime_Train_Cleaned_Normalized[-train,])
```

```
dim(VC_train)
```

```
## [1] 1597   34
```

Build Models

In this section, we built models to predict Violent Crimes per Population and used three different approaches for each prediction:

1. Linear REGRESSION model using stepwise coefficient selection.
2. LASSO REGRESSION model
3. RIDGE REGRESSION model
4. ELASTIC NET REGRESSION model

Violent Crime Models

We repeated the same steps as above using the Violent crimes per population as the response variable. Starting with Linear Regression

LINEAR REGRESSION MODEL

```r
summary(VC_Step_Model )
```

```
##
## Call:
## lm(formula = ViolentCrimesPerPop ~ racePctAsian + pctWFarmSelf +
##     AsianPerCap + PctVacantBoarded + NumInShelters + NumStreet +
##     PopDens + pctUrban + PctTeen2Par + PctHousOccup + PctSameCity85 +
##     PctUnemployed + PctEmplProfServ + PctWorkMomYoungKids + PctVacMore6Mos +
##     MedYrHousBuilt + PctHousNoPhone + MedRentPctHousInc + MedOwnCostPctInc +
##     MedOwnCostPctIncNoMtg, data = VC_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47626 -0.08945 -0.01263  0.07848  0.66051
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.003355   0.036109   0.093 0.925992
## racePctAsian           0.094106   0.024237   3.883 0.000108 ***
## pctWFarmSelf          -0.048175   0.022840  -2.109 0.035080 *
## AsianPerCap            0.043963   0.021967   2.001 0.045532 *
## PctVacantBoarded       0.141455   0.021402   6.609 5.27e-11 ***
## NumInShelters          0.139597   0.035333   3.951 8.13e-05 ***
## NumStreet              0.139245   0.034744   4.008 6.42e-05 ***
## PopDens                0.066259   0.027459   2.413 0.015934 *
## pctUrban              -0.161488   0.033791  -4.779 1.93e-06 ***
## PctTeen2Par            1.167354   0.098506  11.851  < 2e-16 ***
## PctHousOccup           0.433307   0.076847   5.639 2.03e-08 ***
## PctSameCity85         -0.348426   0.079587  -4.378 1.28e-05 ***
## PctUnemployed          0.074472   0.029219   2.549 0.010904 *
## PctEmplProfServ       -0.066924   0.024208  -2.765 0.005767 **
## PctWorkMomYoungKids    0.075972   0.023058   3.295 0.001007 **
## PctVacMore6Mos        -0.090918   0.026472  -3.435 0.000609 ***
## MedYrHousBuilt         0.065726   0.023858   2.755 0.005939 **
## PctHousNoPhone         0.243157   0.026617   9.135  < 2e-16 ***
## MedRentPctHousInc      0.097871   0.026992   3.626 0.000297 ***
## MedOwnCostPctInc       0.058836   0.026695   2.204 0.027668 *
## MedOwnCostPctIncNoMtg -0.063442   0.022320  -2.842 0.004535 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1366 on 1576 degrees of freedom
## Multiple R-squared:  0.6284, Adjusted R-squared:  0.6237
## F-statistic: 133.3 on 20 and 1576 DF,  p-value: < 2.2e-16
```

```r
predicted <- predict(VC_Step_Model, newx = VC_test)# predict on test data
predicted_values <- cbind (actual=VC_test$ViolentCrimesPerPop, predicted)  # combine
```

```
## Warning in base::cbind(...): number of rows of result is not a multiple of
## vector length (arg 1)
```

```r
mean (apply(predicted_values, 1, min)/apply(predicted_values, 1, max))
```

```
## [1] 0.6153482
```

```
calc_RMSE <- function(model){
   RMSE <- sqrt(mean(model$residuals^2))
   return(RMSE)
}
```

```
df1<-data.frame(
  RMSE = calc_RMSE(VC_Step_Model),
  Rsquare = .6284)
```

Split the data again

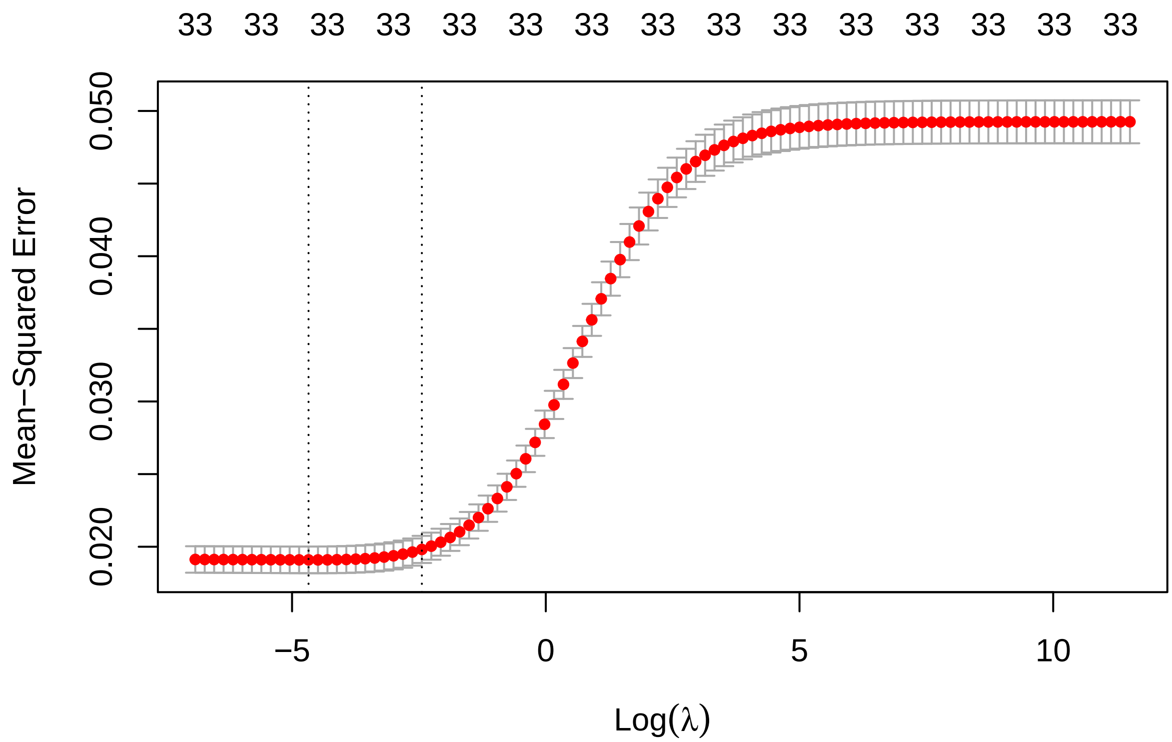
```
#Split the data into Training and Test Set
set.seed(123)
train <- Communities_Crime_Train_Cleaned_Normalized$ViolentCrimesPerPop %>%
         createDataPartition(p=0.8, list = F)
train_data <- Communities_Crime_Train_Cleaned_Normalized[train, ]
test_data <- Communities_Crime_Train_Cleaned_Normalized[-train, ]
```

```
x <- model.matrix(ViolentCrimesPerPop ~.,train_data)[,-1]
y <- train_data$ViolentCrimesPerPop
```

RIDGE REGRESSION MODEL

```
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)
```

```
cv <- cv.glmnet(x, y, alpha = 0, lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)

plot(cv)
```




```r
#Find the best lambda using cross-validation
set.seed(123)
VC_Ridge_Model <- glmnet(x,y, alpha = 0, lambda = cv$lambda.min)
coef(VC_Ridge_Model)
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)           0.025692959
## racePctAsian          0.077469412
## pctWFarmSelf         -0.045845268
## blackPerCap          -0.047659248
## indianPerCap          0.003473235
## AsianPerCap           0.046776958
## HispPerCap            0.038326513
## NumImmig             -0.002133070
## PctVacantBoarded      0.112086270
## PctWOFullPlumb        0.003112270
## NumInShelters         0.088316555
## NumStreet             0.127663119
## LandArea              0.076182681
## PopDens               0.093064419
## PctUsePubTrans        0.000490278
## pctUrban             -0.163946132
## PctTeen2Par           1.253892181
## PctHousOccup          0.341664321
## PctBornSameState      0.157546154
```

```
## PctSameCity85         -0.236128306
## PctSameState85        -0.192288625
## pctWRetire            -0.035391659
## PctUnemployed          0.091076898
## PctEmplManu           -0.028584812
## PctEmplProfServ       -0.092899055
## PctWorkMomYoungKids    0.069922164
## PctImmigRecent         0.015100573
## MedNumBR               0.004970020
## PctVacMore6Mos        -0.070458300
## MedYrHousBuilt         0.040352958
## PctHousNoPhone         0.224146090
## MedRentPctHousInc      0.084180821
## MedOwnCostPctInc       0.053807829
## MedOwnCostPctIncNoMtg -0.065220898
```

```r
summary(VC_Ridge_Model)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      33     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

```r
print(VC_Ridge_Model, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"))
```

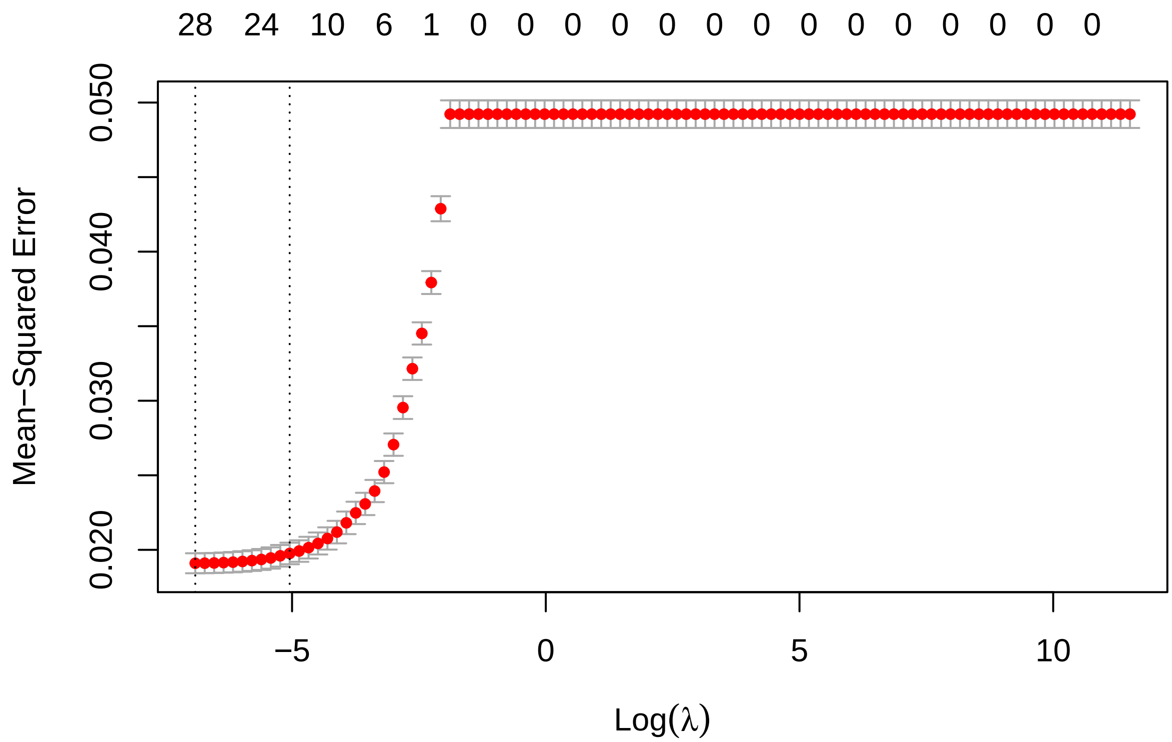
```
##
## Call:  glmnet(x = x, y = y, alpha = 0, lambda = cv$lambda.min)
##
##   Df   %Dev   Lambda
## 1 33 0.6272 0.009326
```

```r
x.test <- model.matrix(ViolentCrimesPerPop ~., test_data)[,-1]
predictions <- VC_Ridge_Model %>% predict(x.test) %>% as.vector()
df2<- data.frame(
  RMSE = RMSE(predictions, test_data$ViolentCrimesPerPop),
  Rsquare = R2(predictions, test_data$ViolentCrimesPerPop))
```

LASSO

```r
set.seed(123)
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)
```

```r
cv <- cv.glmnet(x, y, alpha = 1, lambda = lambdas_to_try,
                      standardize = TRUE, nfolds = 10)

plot(cv)
```




```r
# Best cross-validated lambda
lambda_cv <- cv$lambda.min
VC_Lasso_Model <- glmnet(x,y, alpha=1, lambda = cv$lambda.min )
coef(VC_Lasso_Model)
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)          0.015532344
## racePctAsian         0.081987509
## pctWFarmSelf        -0.033536282
## blackPerCap         -0.036746744
## indianPerCap         .
## AsianPerCap          0.043013316
## HispPerCap           0.031848195
## NumImmig             .
## PctVacantBoarded     0.110960929
## PctWOFullPlumb       .
## NumInShelters        0.077727891
## NumStreet            0.131362637
## LandArea             0.074417999
## PopDens              0.089254102
## PctUsePubTrans       .
## pctUrban            -0.172382108
## PctTeen2Par          1.334368621
## PctHousOccup         0.317021120
## PctBornSameState     0.121253410
```

```
## PctSameCity85         -0.220455061
## PctSameState85        -0.150428698
## pctWRetire            -0.026130454
## PctUnemployed          0.075893644
## PctEmplManu           -0.022099186
## PctEmplProfServ       -0.089687051
## PctWorkMomYoungKids    0.059152216
## PctImmigRecent         0.004221059
## MedNumBR               .
## PctVacMore6Mos        -0.065641489
## MedYrHousBuilt         0.039019231
## PctHousNoPhone         0.233367388
## MedRentPctHousInc      0.081625699
## MedOwnCostPctInc       0.056861918
## MedOwnCostPctIncNoMtg -0.061705199
```

```r
x.test <- model.matrix(ViolentCrimesPerPop ~., test_data)[,-1]
predictions <- VC_Lasso_Model %>%  predict(x.test) %>% as.vector()
df3<- data.frame(
  RMSE = RMSE(predictions, test_data$ViolentCrimesPerPop),
  Rsquare = R2(predictions, test_data$ViolentCrimesPerPop))
```

ELASTIC NET REGRESSION

```r
# Build the model using the training set
set.seed(123)
VC_ENR_model <- train(ViolentCrimesPerPop ~., test_data, method = "glmnet", trControl = trainControl("c
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
# Best tuning parameter
VC_ENR_model$bestTune
```

```
##     alpha     lambda
## 37 0.1375 0.01783083
```

```r
coef(VC_ENR_model$finalModel, VC_ENR_model$bestTune$lambda)
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)            0.157680636
## racePctAsian           0.088754092
## pctWFarmSelf          -0.053725218
## blackPerCap           -0.028161126
## indianPerCap          -0.002883816
## AsianPerCap            .
## HispPerCap            -0.041325116
## NumImmig               0.027737207
## PctVacantBoarded       0.185972176
## PctWOFullPlumb         .
## NumInShelters          0.097926956
## NumStreet              0.172466010
## LandArea               .
## PopDens                .
## PctUsePubTrans         .
## pctUrban              -0.056955146
```

```
## PctTeen2Par           1.244844210
## PctHousOccup          0.536922258
## PctBornSameState      0.157099376
## PctSameCity85        -0.445948912
## PctSameState85        0.029187107
## pctWRetire           -0.024791641
## PctUnemployed         0.096527635
## PctEmplManu           .
## PctEmplProfServ      -0.056410183
## PctWorkMomYoungKids   .
## PctImmigRecent        0.025632954
## MedNumBR             -0.016154825
## PctVacMore6Mos       -0.065540034
## MedYrHousBuilt        0.022356667
## PctHousNoPhone        0.109971624
## MedRentPctHousInc     0.037474487
## MedOwnCostPctInc      0.024442327
## MedOwnCostPctIncNoMtg -0.054749612
```

```r
x.test <- model.matrix(ViolentCrimesPerPop ~., test_data)[,-1]
predictions <- VC_ENR_model %>% predict(x.test)
# Model performance metrics
df4<- data.frame(
  RMSE = RMSE(predictions, test_data$ViolentCrimesPerPop),
  Rsquare = R2(predictions, test_data$ViolentCrimesPerPop)
)
```

```r
final <- rbind( df1,df2,df3,df4)
rownames(final) <-(c("VC_Step_Model","VC_Ridge_Model","VC_Lasso_Model","VC_ENR_model"))
#colnames(final) <- c("Model Name", "RMSE", "Rsquare")
final <- as.data.frame(final)
final <- cbind(Model= rownames(final), final)
rownames(final) <- 1:nrow(final)

final
```

```
##            Model      RMSE   Rsquare
## 1  VC_Step_Model 0.1356970 0.6284000
## 2 VC_Ridge_Model 0.1330519 0.6398185
## 3 VC_Lasso_Model 0.1330618 0.6397077
## 4   VC_ENR_model 0.1288544 0.6634376
```

Model	RMSE	Rsquare
VC_Step_Model	0.1356970	0.6284000
VC_Ridge_Model	0.1330519	0.6398185
VC_Lasso_Model	0.1330618	0.6397077
VC_ENR_model	0.1288544	0.6634376

References

[1] UCI Machine Learning Repository. Center for Machine Learning and Intelligent Systems https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

[2] "USING MACHINE LEARNING ALGORITHMS TO ANALYZE CRIME DATA" by Lawrence McClendon and Natarajan Meghanathan (Jackson State University, 1400 Lynch St, Jackson, MS, USA) (Machine Learning

and Applications: An International Journal (MLAIJ) Vol.2, No.1, March 2015) Link

[3] “A Convex Framework for Fair Regression” by Richard Berk, Hoda Heidari , Shahin Jabbari, Matthew Joseph, Michael Kearns , Jamie Morgenstern, Seth Neel , and Aaron Roth (Department of Statistics, Department of Criminology, Department of Computer and Information Science of the University of Pennsylvania)(June 9, 2017) Citation: arXiv:1706.02409 [cs.LG] Link

[4] “A Comparative Study to Evaluate Filtering Methods for Crime Data Feature Selection”, Masila Abdul Jalil, Fatihah Mohd, and Noor Maizura Mohamad Noor (School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu, Malaysia) (October 2017) Link

[5] Variance Inflation Factor

[6] Data Camp Tutorial