IDS Report

Data Preprocessing and Preliminary Analysis to get inferences from the dataset - ***Communities and Crime*** (Reference : http://archive.ics.uci.edu/ml/datasets/communities+and+crime)

Submitted to -

Dr. Sudheer Sharma
Dr. Sakthi Balan
Dr.Subrat Dash

Introduction

We have decided to work on a socioeconomic dataset, that describes crime in different communities.

We would be performing Data Preprocessing and Preliminary Analysis and will get inferences from the data using Libraries in **Python**.

Abstract:

Communities within the United States. The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR.

Data-set Description

The variables present in the dataset revolves around the community, such as the ratio of the population considered urban or rural, 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.

Limitation of the Data-set :

The survey for the Data-set was of the police departments with at least 100 officers, plus a random sample of smaller departments. So, communities not found in both census and crime datasets are not present in this data set. So many communities may be missing in the Data-set.

Normalization of the data

Data in the data set is based on original values. All numeric data was normalized into the decimal range 0.00-1.00 using an Unsupervised, equal-interval binning method.

The normalization maintains rough ratios of values of an attribute with the available precision except for extreme cases i.e. more than 3 Standard Deviation above the mean are normalized to 1.00 and all values more than 3 Standard Deviation below the mean are normalized to 0.00.

Goal

We will be performing Data Preprocessing and Preliminary Analysis on the dataset and will try to get inferences like crime rate through different states, which per capita income group is affected most by the crimes.

Data set Statistics:

Total number of :

- Attributes: 128
    - Predictive: 122
    - Non-Predictive: 5
    - Goal: 1
- Attribute Characteristics:Multivariate
    - Numeric: 127
    - String: 1
- Instances: 1994
- Missing Values: Yes
- Area: Socioeconomic

Data Preprocessing and Preliminary Analysis

Importing the Libraries

- **Numpy:** Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

- **Pandas:** Pandas provides high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

- **Matplotlib.pyplot :** Provides a MATLAB-like plotting framework.

- **Seaborn :** It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Label Encoder:** it is a utility class to help normalize labels such that they contain only values between 0 and n_classes-1.

- **Standard Scalar:** it standardizes features of the data set by scaling to unit variance and removing the mean (optionally) using column summary statistics on the samples in the training set.

- **Pylab:** It is a convenience module that bulk imports matplotlib.pyplot (for plotting)

- **Statsmodels.formula.api:** A convenience interface for specifying models using formula strings and DataFrames.

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import statsmodels.api as sm
     import statsmodels.formula.api as smf
     import pylab
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder
```

Mounting Drive to import Files

```
[ ]  from google.colab import drive
     drive.mount("/content/gdrive")
```

- We have mounted google drive with the colab, so we can get read/write access to files uploaded in the drive.

Importing the Dataset using Panda

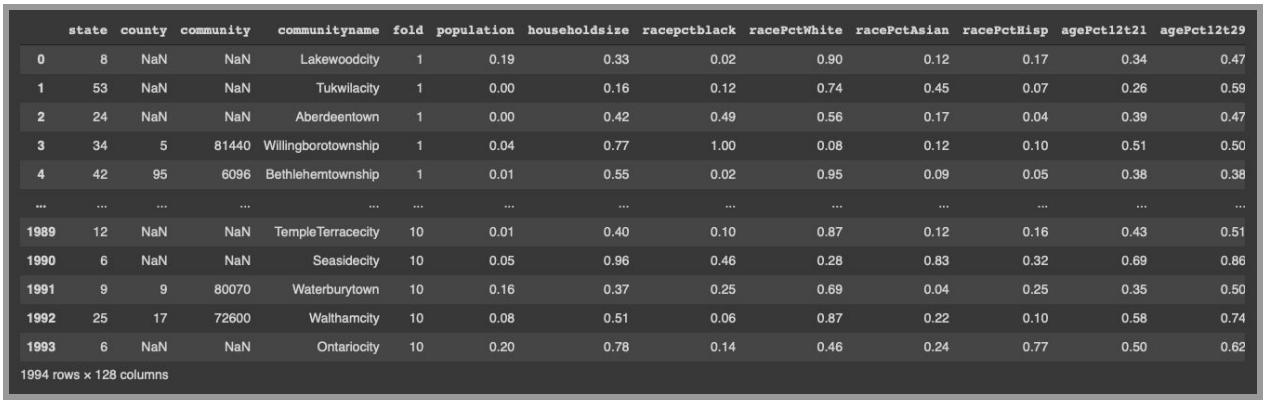
```
[ ]  pd.set_option('display.max_rows', 20)
     text=pd.read_csv('/content/gdrive/My Drive/IDS
     Report/communitiesFINAL.csv')
```

- Open the csv file using pd.read which contains all the information of our dataset and assign a variable text to it.

Replacing the Missing Values

```
[ ]  text=text.replace('?',float("NAN"))
     text
```

- By the above code, we have replaced the missing value (‘?’) with NaN (Not a Number) to process further.
- Printing the dataset after replacing the values.



	state	county	community	communityname	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29
0	8	NaN	NaN	Lakewoodcity	1	0.19	0.33	0.02	0.90	0.12	0.17	0.34	0.47
1	53	NaN	NaN	Tukwilacity	1	0.00	0.16	0.12	0.74	0.45	0.07	0.26	0.59
2	24	NaN	NaN	Aberdeentown	1	0.00	0.42	0.49	0.56	0.17	0.04	0.39	0.47
3	34	5	81440	Willingborotownship	1	0.04	0.77	1.00	0.08	0.12	0.10	0.51	0.50
4	42	95	6096	Bethlehemtownship	1	0.01	0.55	0.02	0.95	0.09	0.05	0.38	0.38
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1989	12	NaN	NaN	TempleTerracecity	10	0.01	0.40	0.10	0.87	0.12	0.16	0.43	0.51
1990	6	NaN	NaN	Seasidecity	10	0.05	0.96	0.46	0.28	0.83	0.32	0.69	0.86
1991	9	9	80070	Waterburytown	10	0.16	0.37	0.25	0.69	0.04	0.25	0.35	0.50
1992	25	17	72600	Walthamcity	10	0.08	0.51	0.06	0.87	0.22	0.10	0.58	0.74
1993	6	NaN	NaN	Ontariocity	10	0.20	0.78	0.14	0.46	0.24	0.77	0.50	0.62

1994 rows × 128 columns

Displaying all columns of the dataset

```
[6] text.columns

    Index(['state', 'county', 'community', 'communityname', 'fold', 'population',
           'householdsize', 'racepctblack', 'racePctWhite', 'racePctAsian',
           ...
           'LandArea', 'PopDens', 'PctUsePubTrans', 'PolicCars', 'PolicOperBudg',
           'LemasPctPolicOnPatr', 'LemasGangUnitDeploy', 'LemasPctOfficDrugUn',
           'PolicBudgPerPop', 'ViolentCrimesPerPop'],
          dtype='object', length=128)
```

- text.columns() prints all the attributes names.

Displaying the top 5 Rows of the dataset

```
[ ] text.head()
```

	state	county	community	communityname	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29	a
0	8	NaN	NaN	Lakewoodcity	1	0.19	0.33	0.02	0.90	0.12	0.17	0.34	0.47	
1	53	NaN	NaN	Tukwilacity	1	0.00	0.16	0.12	0.74	0.45	0.07	0.26	0.59	
2	24	NaN	NaN	Aberdeentown	1	0.00	0.42	0.49	0.56	0.17	0.04	0.39	0.47	
3	34	5	81440	Willingborotownship	1	0.04	0.77	1.00	0.08	0.12	0.10	0.51	0.50	
4	42	95	6096	Bethlehemtownship	1	0.01	0.55	0.02	0.95	0.09	0.05	0.38	0.38	

5 rows × 128 columns

- text.head() prints the top 5 rows of dataset.

Describing the Data Set

```
[ ]  text.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 1993 entries, 0 to 1992
     Columns: 128 entries, state to ViolentCrimesPerPop
     dtypes: float64(100), int64(2), object(26)
     memory usage: 1.9+ MB
```

- The above code shows the information like rows, columns, data type and the memory.

Determining the Size of the Data Set

```
[ ]  text.shape

     (1994, 128)
```

- Number of rows = 1994
- Number of columns = 128

Summarizing the statistics of Raw Data

```
[ ] stat=pd.read_csv('/content/gdrive/My Drive/IDS Report/Summary_Stat.txt',header=None)
    stat
```

	0	1	2	3	4	5	6	7	8
0	Attribute	Min	Max	Mean	SD	Correl	Median	Mode	Missing
1	population	0	1	0.06	0.13	0.37	0.02	0.01	0
2	householdsize	0	1	0.46	0.16	-0.03	0.44	0.41	0
3	racepctblack	0	1	0.18	0.25	0.63	0.06	0.01	0
4	racePctWhite	0	1	0.75	0.24	-0.68	0.85	0.98	0
...	...	...	...	...	...	...	...	...	...
119	LemasPctPolicOnPatr	0	1	0.70	0.21	-0.08	0.75	0.74	1675
120	LemasGangUnitDeploy	0	1	0.44	0.41	0.12	0.5	0	1675
121	LemasPctOfficDrugUn	0	1	0.09	0.24	0.35	0	0	0
122	PolicBudgPerPop	0	1	0.20	0.16	0.10	0.15	0.12	1675
123	ViolentCrimesPerPop	0	1	0.24	0.23	1.00	0.15	0.03	0

124 rows × 9 columns

- This the Summary Statistics which contains following information of each attribute :
    - Minimum
    - Maximum
    - Mean
    - Standard deviation
    - Correl
    - Median
    - Mode
    - Missing values

Data Preprocessing

> Missing Values :

In Data Preprocessing, Identifying and handling Missing Values is crucial, as failing this may lead us to draw inaccurate and faulty conclusions and inferences from the data.

Handling Missing Values

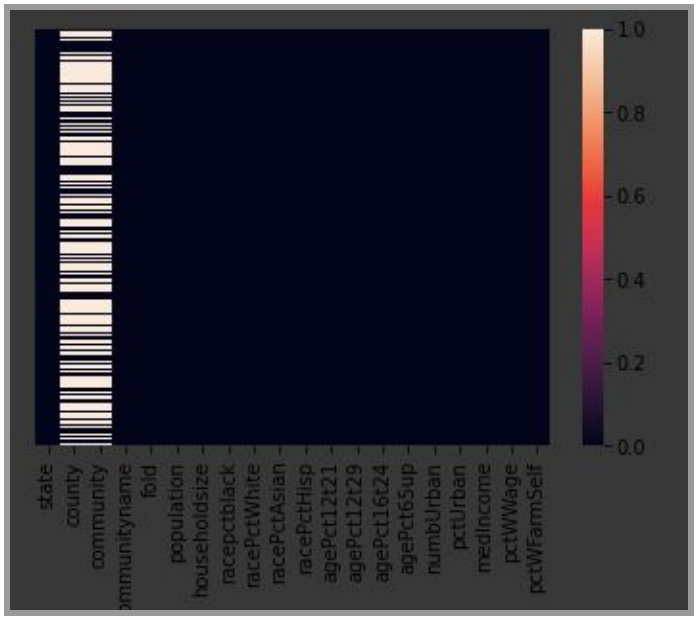
We can handle the missing values in two ways,

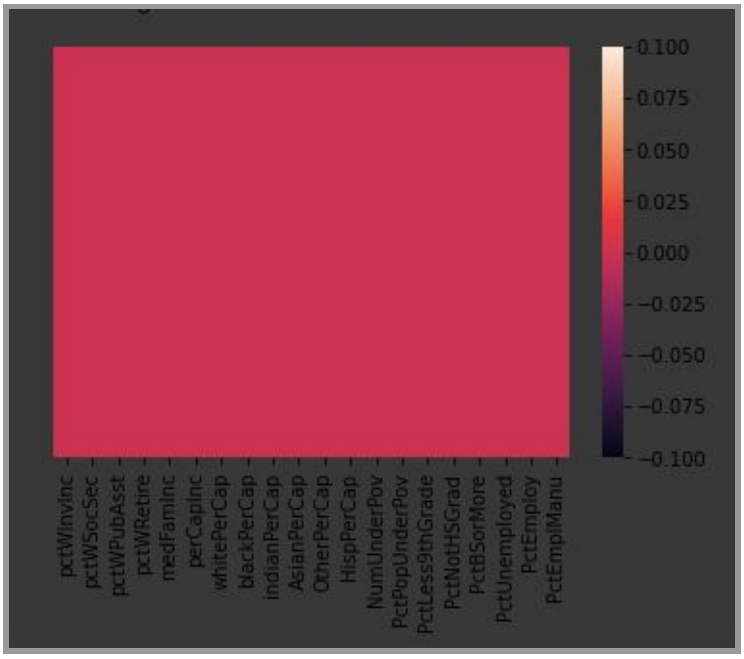
1. Removing the entire row that contains missing value.
2. Replacing the Missing value with an appropriate measure of central tendency (Mean, Median or Mode).

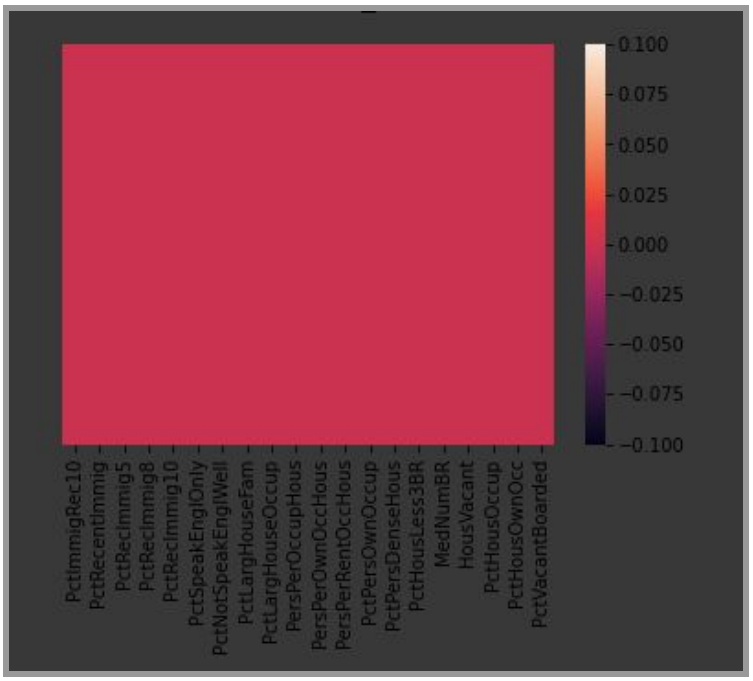
Finding the Missing Values using Heat Map

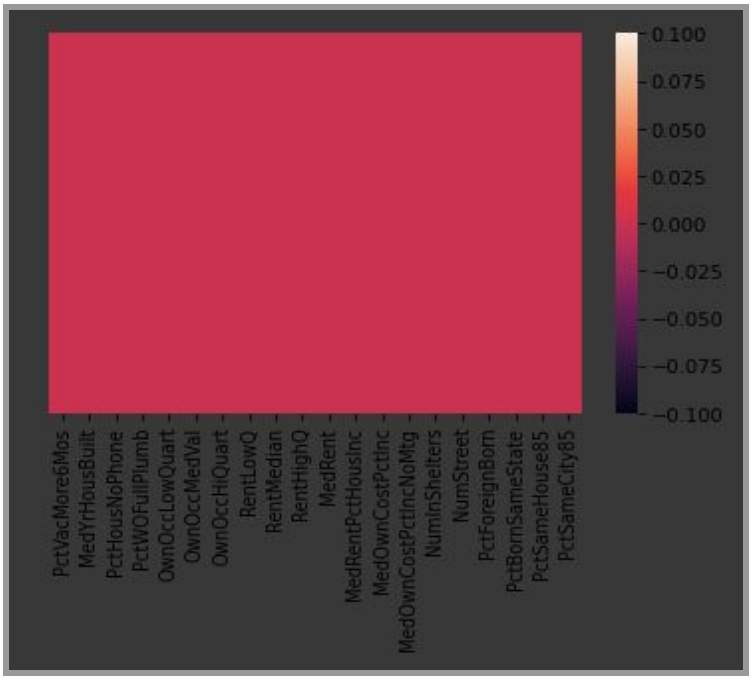
Here, in this dataset, we first find the missing values using Heat Map so as to get a visual representation of the Missing Values for the various attributes in the dataset.

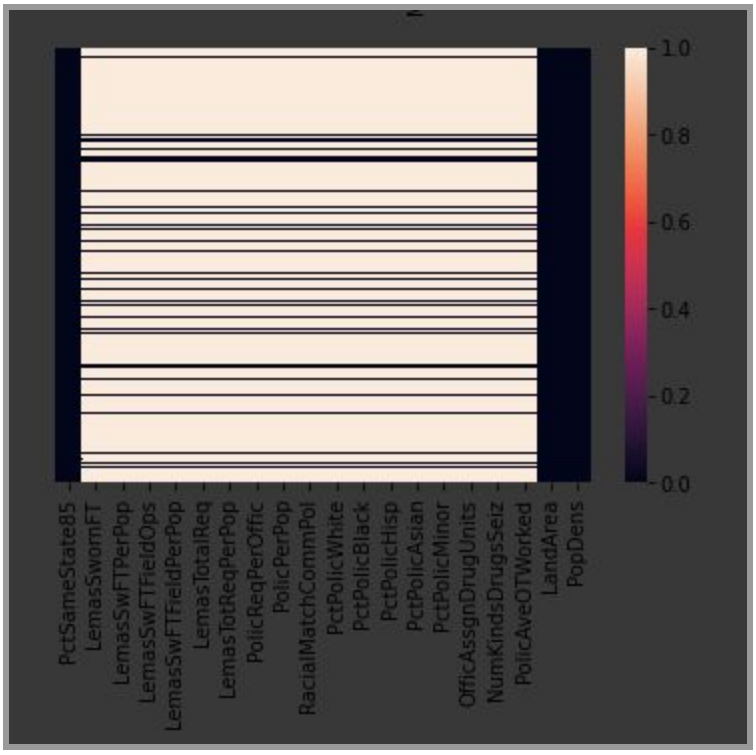
```
[ ]  df1 = text.iloc[:,0:20]
     df2 = text.iloc[:,20:40]
     df3 = text.iloc[:,40:60]
     df4 = text.iloc[:,60:80]
     df5 = text.iloc[:,80:100]
     df6 = text.iloc[:,100:120]
     df7 = text.iloc[:,120:]
     sns.heatmap(df1.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df2.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df3.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df4.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df5.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df6.sample(250).isnull(),yticklabels=False)
     plt.show()
     sns.heatmap(df7.sample(250).isnull(),yticklabels=False)
     plt.show()
```

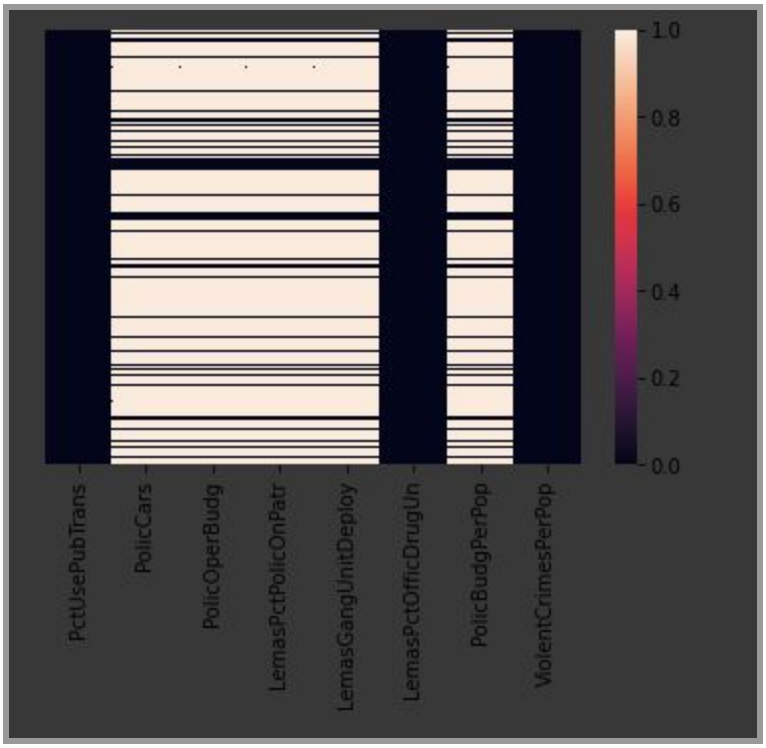



0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
pctWInvInc
pctWSocSec
pctWPubAsst
pctWRetire
medFamInc
perCapInc
whitePerCap
blackPerCap
indianPerCap
AsianPerCap
OtherPerCap
HispPerCap
NumUnderPov
PctPopUnderPov
PctLess9thGrade
PctNotHSGrad
PctBSorMore
PctUnemployed
PctEmploy
PctEmplManu


0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
PctImmigRec10
PctRecentImmig
PctRecImmig5
PctRecImmig8
PctRecImmig10
PctSpeakEnglOnly
PctNotSpeakEnglWell
PctLargHouseFam
PctLargHouseOccup
PersPerOccupHous
PersPerOwnOccHous
PersPerRentOccHous
PctPersOwnOccup
PctPersDenseHous
PctHousLess3BR
MedNumBR
HousVacant
PctHousOccup
PctHousOwnOcc
PctVacantBoarded


0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
PctVacMore6Mos
MedYrHousBuilt
PctHousNoPhone
PctWOFullPlumb
OwnOccLowQuart
OwnOccMedVal
OwnOccHiQuart
RentLowQ
RentMedian
RentHighQ
MedRent
MedRentPctHousInc
MedOwnCostPctInc
MedOwnCostPctIncNoMtg
NumInShelters
NumStreet
PctForeignBorn
PctBornSameState
PctSameHouse85
PctSameCity85


1.0
0.8
0.6
0.4
0.2
0.0
PctSameState85
LemasSwornFT
LemasSwFTPerPop
LemasSwFTFieldOps
LemasSwFTFieldPerPop
LemasTotalReq
LemasTotReqPerPop
PolicReqPerOffic
PolicPerPop
RacialMatchCommPol
PctPolicWhite
PctPolicBlack
PctPolicHisp
PctPolicAsian
PctPolicMinor
OfficAssgnDrugUnits
NumKindsDrugsSeiz
PolicAveOTWorked
LandArea
PopDens


1.0
0.8
0.6
0.4
0.2
0.0
PctUsePubTrans
PolicCars
PolicOperBudg
LemasPctPolicOnPatr
LemasGangUnitDeploy
LemasPctOfficDrugUn
PolicBudgPerPop
ViolentCrimesPerPop

> Null Values

Removing or Eradicating Null Values are as important as dealing with Missing Values.

Evaluating the total number of Null values

For our dataset, we initially calculate the total number of Null Values.

```
pd.set_option('display.max_rows',None)
text.isnull().sum()
```

```
state                     0
county                 1174
community              1177
communityname             0
fold                      0
population                0
householdsize             0
racepctblack              0
racePctWhite              0
racePctAsian              0
racePctHisp               0
agePct12t21               0
agePct12t29               0
agePct16t24               0
agePct65up                0
numbUrban                 0
pctUrban                  0
medIncome                 0
pctWWage                  0
pctWFarmSelf              0
pctWInvInc                0
pctWSocSec                0
pctWPubAsst               0
pctWRetire                0
medFamInc                 0
perCapInc                 0
whitePerCap               0
blackPerCap               0
```

```
LemasSwornFT              1675
LemasSwFTPerPop           1675
LemasSwFTFieldOps         1675
LemasSwFTFieldPerPop      1675
LemasTotalReq             1675
LemasTotReqPerPop         1675
PolicReqPerOffic          1675
PolicPerPop               1675
RacialMatchCommPol        1675
PctPolicWhite             1675
PctPolicBlack             1675
PctPolicHisp              1675
PctPolicAsian             1675
PctPolicMinor             1675
OfficAssgnDrugUnits       1675
NumKindsDrugsSeiz         1675
PolicAveOTWorked          1675
LandArea                     0
PopDens                      0
PctUsePubTrans               0
PolicCars                 1675
PolicOperBudg             1675
LemasPctPolicOnPatr       1675
LemasGangUnitDeploy       1675
LemasPctOfficDrugUn          0
PolicBudgPerPop           1675
ViolentCrimesPerPop          0
dtype: int64
```

What we infer from this outcome is that the Null values are mostly populated in a few attributes that are, County, Community, etc.

Dealing with Null Values - Eradicating them

So, to deal with these Null values, we plan to remove the attributes that contain Null Values -

```
[ ]  del text['LemasTotReqPerPop']
     del text['county']
     del text['community']
     del text['LemasSwornFT']
     del text['LemasSwFTPerPop']
     del text['PolicReqPerOffic']
     del text['PolicBudgPerPop']
     del text['PolicCars']
     del text['PolicPerPop']
     del text['PctPolicBlack']
     del text['PolicAveOTWorked']
     del text['PolicOperBudg']
     del text['PctPolicMinor']
     del text['LemasTotalReq']
     del text['PctPolicAsian']
     del text['NumKindsDrugsSeiz']
     del text['LemasPctPolicOnPatr']
     del text['LemasGangUnitDeploy']
     del text['LemasPctOfficDrugUn']
     del text['RacialMatchCommPol']
     del text['PctPolicHisp']
     del text['PctPolicWhite']
     del text['LemasTotalReq']
     del text['PctPolicAsian']
```

Re-Evaluating the total number of Null Values

We re-evaluate the total number of Null values after removing the columns that majorly contained the Null Values, and as expected, we see a major drop in the total number of Null Values.

```
[ ]
    text.isnull().sum()

    state                   0
    communityname           0
    fold                    0
    population              0
    householdsize           0
    racepctblack            0
    racePctWhite            0
    racePctAsian            0
    racePctHisp             0
    agePct12t21             0
    agePct12t29             0
    agePct16t24             0
    agePct65up              0
    numbUrban               0
    pctUrban                0
    medIncome               0
    pctWWage                0
    pctWFarmSelf            0
    pctWInvInc              0
    pctWSocSec              0
    pctWPubAsst             0
    pctWRetire              0
    medFamInc               0
    perCapInc               0
    whitePerCap             0
    blackPerCap             0
    indianPerCap            0
    AsianPerCap             0
```

```
PctHousOccup            0
PctHousOwnOcc           0
PctVacantBoarded        0
PctVacMore6Mos          0
MedYrHousBuilt          0
PctHousNoPhone          0
PctWOFullPlumb          0
OwnOccLowQuart          0
OwnOccMedVal            0
OwnOccHiQuart           0
RentLowQ                0
RentMedian              0
RentHighQ               0
MedRent                 0
MedRentPctHousInc       0
MedOwnCostPctInc        0
MedOwnCostPctIncNoMtg   0
NumInShelters           0
NumStreet               0
PctForeignBorn          0
PctBornSameState        0
PctSameHouse85          0
PctSameCity85           0
PctSameState85          0
LandArea                0
PopDens                 0
PctUsePubTrans          0
ViolentCrimesPerPop     0
dtype: int64
```

Re-Assessing the Size of the Dataset

```
[ ]  text.shape

     (1994, 103)
```

Describing the Data after Cleaning

```
[ ] display(text.describe())
```

	state	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29	agePct16t24	agePct65u
count	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.000000	1994.00000
mean	28.683551	5.493982	0.057593	0.463395	0.179629	0.753716	0.153681	0.144022	0.424218	0.493867	0.336264	0.42316
std	16.397553	2.873694	0.126906	0.163717	0.253442	0.244039	0.208877	0.232492	0.155196	0.143564	0.166505	0.17918
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	12.000000	3.000000	0.010000	0.350000	0.020000	0.630000	0.040000	0.010000	0.340000	0.410000	0.250000	0.30000
50%	34.000000	5.000000	0.020000	0.440000	0.060000	0.850000	0.070000	0.040000	0.400000	0.480000	0.290000	0.42000
75%	42.000000	8.000000	0.050000	0.540000	0.230000	0.940000	0.170000	0.160000	0.470000	0.540000	0.360000	0.53000
max	56.000000	10.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 101 columns

After cleaning the data, we compare the summary of the clean data with the dirty data and find that the statistics are very much similar to that of the dirty data. The values of the central tendencies being almost the same.

Hence, we conclude that the Cleaning (or Data Preprocessing) of the dataset does not change the features of the original dataset.

Retrieving the Categorical Values from the Dataset

After cleaning the dataset we move on to the next step of Data Preprocessing, that is Encoding the categorical data so as to get rid of issues caused by them later. Listing the Categorical Values:

```
[ ]  categorical=list(text.select_dtypes(include=['object']).columns.values)
     categorical

     ['communityname', 'OtherPerCap']
```

Encoding the Categorical Values using LabelEncoder

To encode the Categorical Values in the dataset, we use LabelEncoder from the sklearn's preprocessing library.

```
[ ]  text["OtherPerCap"] = text["OtherPerCap"].replace(np.nan, 'none', regex=True)

     le=LabelEncoder()
     for i in range(0,len(categorical)):
       text[categorical[i]]=le.fit_transform(text[categorical[i]])
       x=1
     text.head()
```

	state	communityname	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29	agePct16t24
0	8	804	1	0.19	0.33	0.02	0.90	0.12	0.17	0.34	0.47	0.29
1	53	1625	1	0.00	0.16	0.12	0.74	0.45	0.07	0.26	0.59	0.35
2	24	1	1	0.00	0.42	0.49	0.56	0.17	0.04	0.39	0.47	0.28
3	34	1787	1	0.04	0.77	1.00	0.08	0.12	0.10	0.51	0.50	0.34
4	42	141	1	0.01	0.55	0.02	0.95	0.09	0.05	0.38	0.38	0.23

5 rows × 103 columns

Dropping the Duplicate Values:

Next up we get rid of the duplicate values in the dataset using the pre-existing 'drop_duplicates' function of the Pandas library and display the complete Dataset.

```
[ ] pd.reset_option('display.max_rows', None)
    com=text['communityname']
    text.sort_values("communityname", inplace = True)
    text.drop_duplicates(subset ="communityname", keep = False, inplace = True)
    text
```

	state	communityname	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29	agePct16t24
2	24	1	1	0.00	0.42	0.49	0.56	0.17	0.04	0.39	0.47	0.28
1401	34	2	8	0.01	0.56	0.22	0.75	0.21	0.09	0.36	0.48	0.28
1671	40	3	9	0.01	0.29	0.08	0.72	0.03	0.02	0.61	0.58	0.55
1209	6	4	7	0.02	0.66	0.02	0.85	0.42	0.11	0.49	0.42	0.25
1541	45	5	8	0.02	0.39	0.60	0.51	0.04	0.01	0.34	0.38	0.23
...	...	...	...	...	...	...	...	...	...	...	...	...
675	39	1823	4	0.14	0.40	0.74	0.38	0.02	0.07	0.42	0.43	0.29
670	6	1824	4	0.03	0.41	0.05	0.64	0.47	0.33	0.40	0.54	0.34
89	6	1825	1	0.04	0.34	0.01	0.89	0.06	0.20	0.32	0.32	0.18
1657	40	1826	9	0.02	0.55	0.01	0.91	0.10	0.04	0.48	0.46	0.24
307	39	1827	2	0.03	0.35	0.21	0.83	0.01	0.01	0.39	0.45	0.30

1707 rows × 103 columns

Dimensionality Reduction using Correlation Matrix

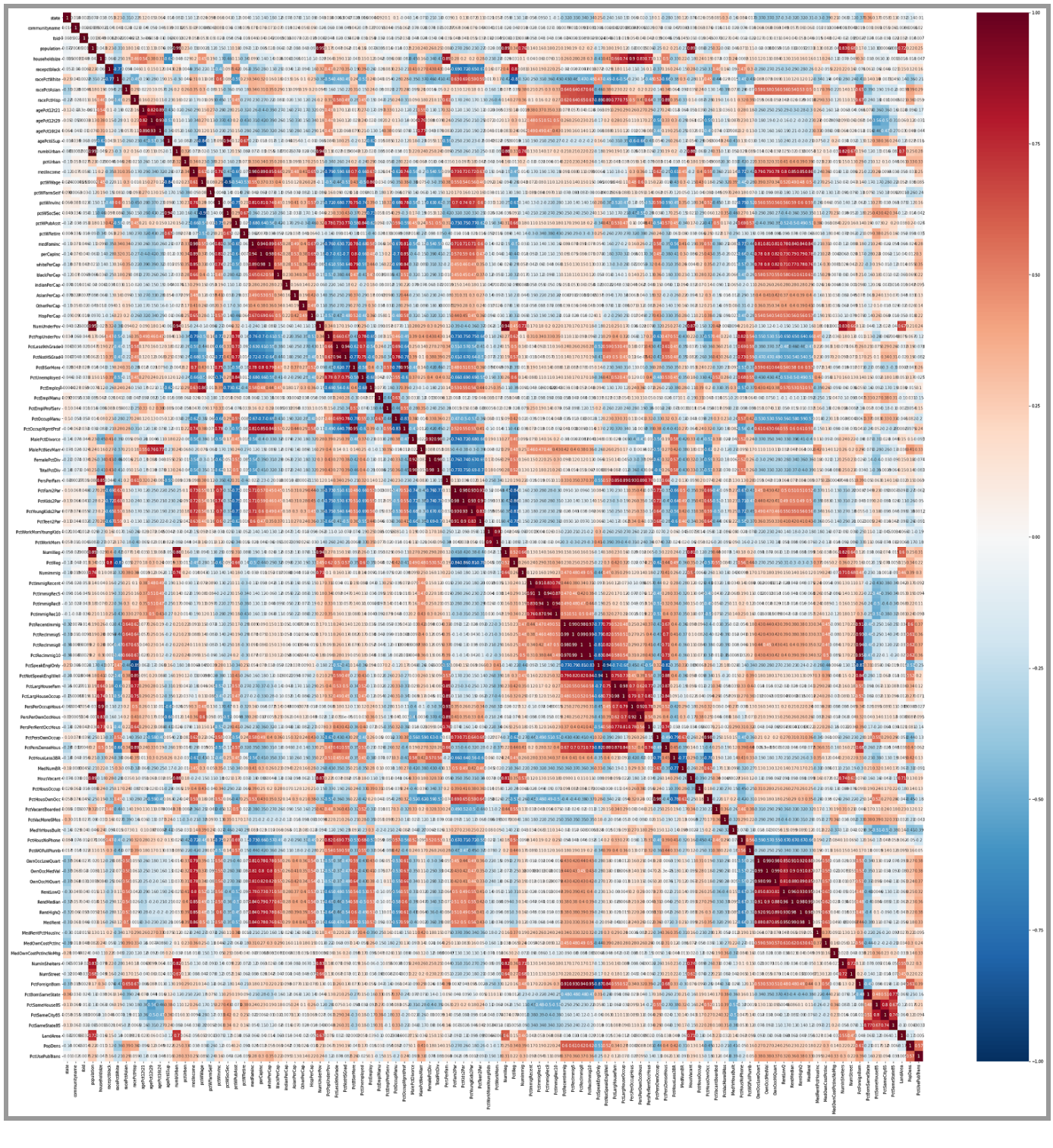
The last stage of Data Preprocessing is the Dimensionality Reduction wherein we reduce the number of input variables in the dataset. More input features often make a predictive modeling task more challenging to model, more generally referred to as the curse of dimensionality.

Correlation Matrix before cleaning:

We print the Correlation matrix before the deletion of inconsequential attributes.

```
[ ]  c=text.iloc[:,:]
     features = list(c)
     features.remove("ViolentCrimesPerPop")
     X=text[features]
     correlation = X.corr()
     plt.figure(figsize=(50, 50))
     sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu_r")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2fbc7104e0>
```



Deleting the inconsequential Attributes:

To clean the data and obtain good data, we delete the attributes that are least correlated. We do this on the basis of redundancy and whether the data of the attribute is relevant or not.

We delete the following attributes:

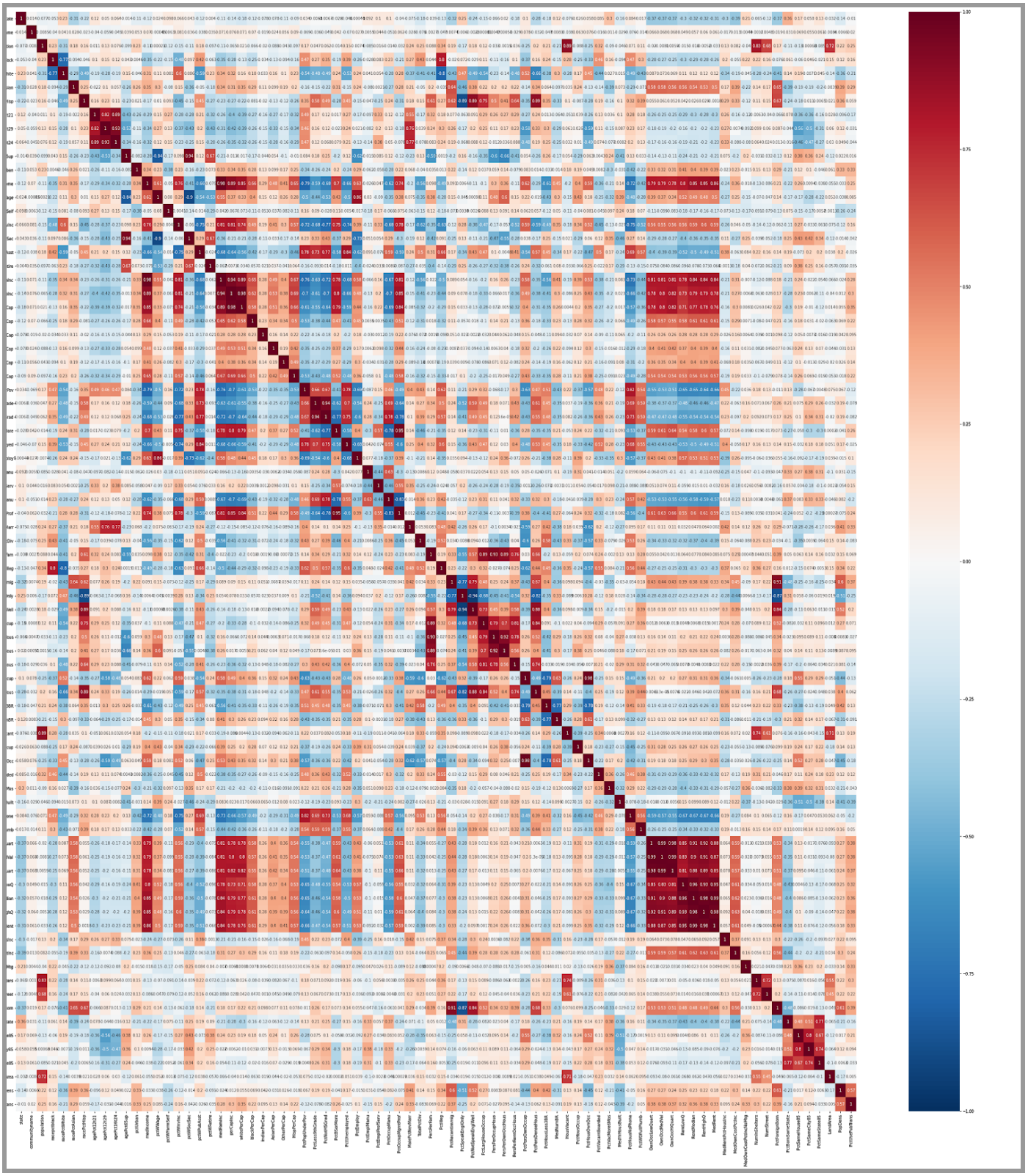
```
[ ]  del text['PctRecImmig8']
     del text['NumUnderPov']
     del text['MalePctDivorce']
     del text['FemalePctDiv']
     del text['PctFam2Par']
     del text['PctKids2Par']
     del text['PctYoungKids2Par']
     del text['PctTeen2Par']
     del text['householdsize']
     del text['fold']
     del text['numbUrban']
     del text['PctWorkMomYoungKids']
     del text['PctImmigRecent']
     del text['PctImmigRec5']
     del text['PctWorkMom']
     del text['NumIlleg']
     del text['NumImmig']
     del text['PctImmigRec8']
     del text['PctImmigRec10']
     del text['PctRecImmig5']
     del text['PctRecImmig10']
     del text['PctLargHouseFam']
```

Correlation matrix after the deletion (cleaning):

We compute the correlation matrix after cleaning the data and observe that the matrix contains less white colored areas which suggests that we successfully eradicated the columns that were less correlated or less relevant.

```
[ ]  c=text.iloc[:,:]
     features = list(c)
     features.remove("ViolentCrimesPerPop")
     X=text[features]
     correlation = X.corr()
     plt.figure(figsize=(50, 50))
     sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu_r")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2fbc7104e0>
```

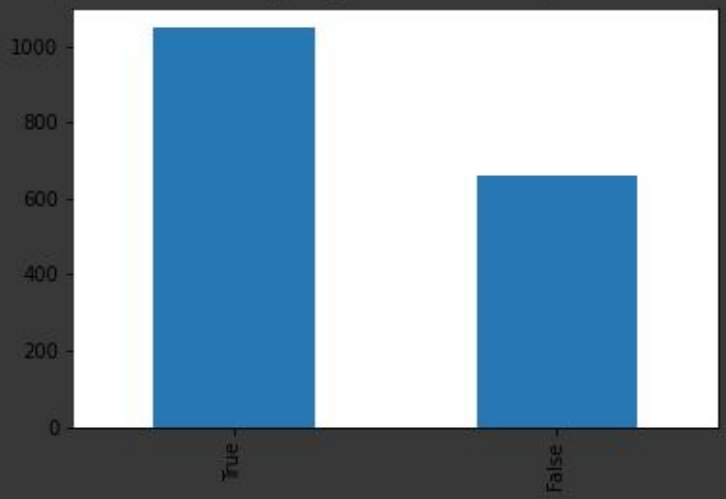


Feature Aggregation

We execute Feature Accumulation so as to structure the data in a better perspective. This feature aggregation provides us with an abstract view of the data as behaviour of groups is more stable than individual data objects.

```
[ ]  text['HighCrime']=(text['ViolentCrimesPerPop']>0.1)
     text.HighCrime.value_counts().plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe3ce346a20>
```




```
[ ]  x=text.HighCrime.value_counts()
     x
```

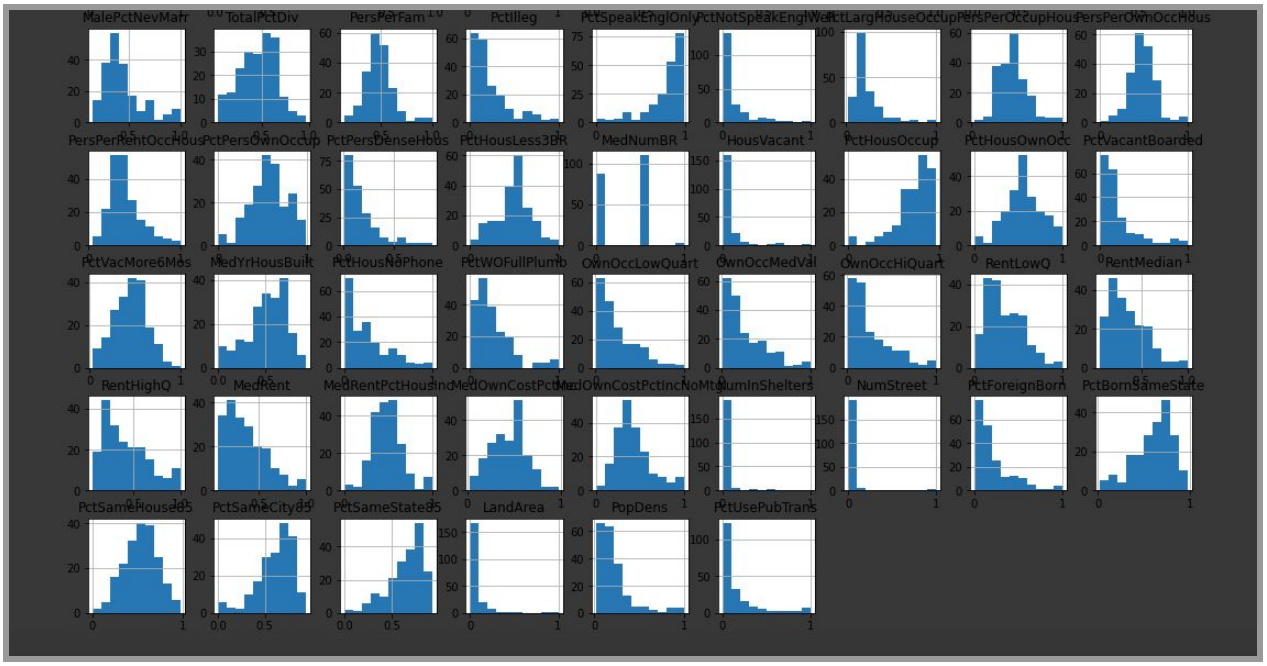
```
True     1046
False     661
Name: HighCrime, dtype: int64
```

Sample Data of 200 records Represented in terms of Histogram -

Now we sample the dataset and represent 200 records in terms of histogram.

```
[ ]  text.sample(200).hist(column=['state' , 'population' , 'racepctblack' , 'racePctWhite' , 'racePctAsian' ,
     'racePctHisp' , 'agePct12t21' , 'agePct12t29' , 'agePct16t24' , 'agePct65up' , 'pctUrban' , 'medIncome' ,
      'pctWWage' , 'pctWFarmSelf' , 'pctWInvInc' , 'pctWSocSec' , 'pctWPubAsst' , 'pctWRetire' ,
     'medFamInc' , 'perCapInc' , 'whitePerCap' , 'blackPerCap' , 'indianPerCap' , 'AsianPerCap' , 'OtherPerCap' ,
      'HispPerCap' , 'PctPopUnderPov' , 'PctLess9thGrade' , 'PctNotHSGrad' , 'PctBSorMore' , 'PctUnemployed' ,
       'PctEmploy' , 'PctEmplManu' , 'PctEmplProfServ' , 'PctOccupManu' ,
     'PctOccupMgmtProf' , 'MalePctNevMarr' , 'TotalPctDiv' , 'PersPerFam' , 'PctIlleg' , 'PctSpeakEnglOnly' ,
      'PctNotSpeakEnglWell' , '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' ],figsize=(18,18))
    pylab.suptitle("Analyzing distribution for the series", fontsize="xx-large")
```

And analyse the distribution of the sample for the frequency of various feature values in the dataset.

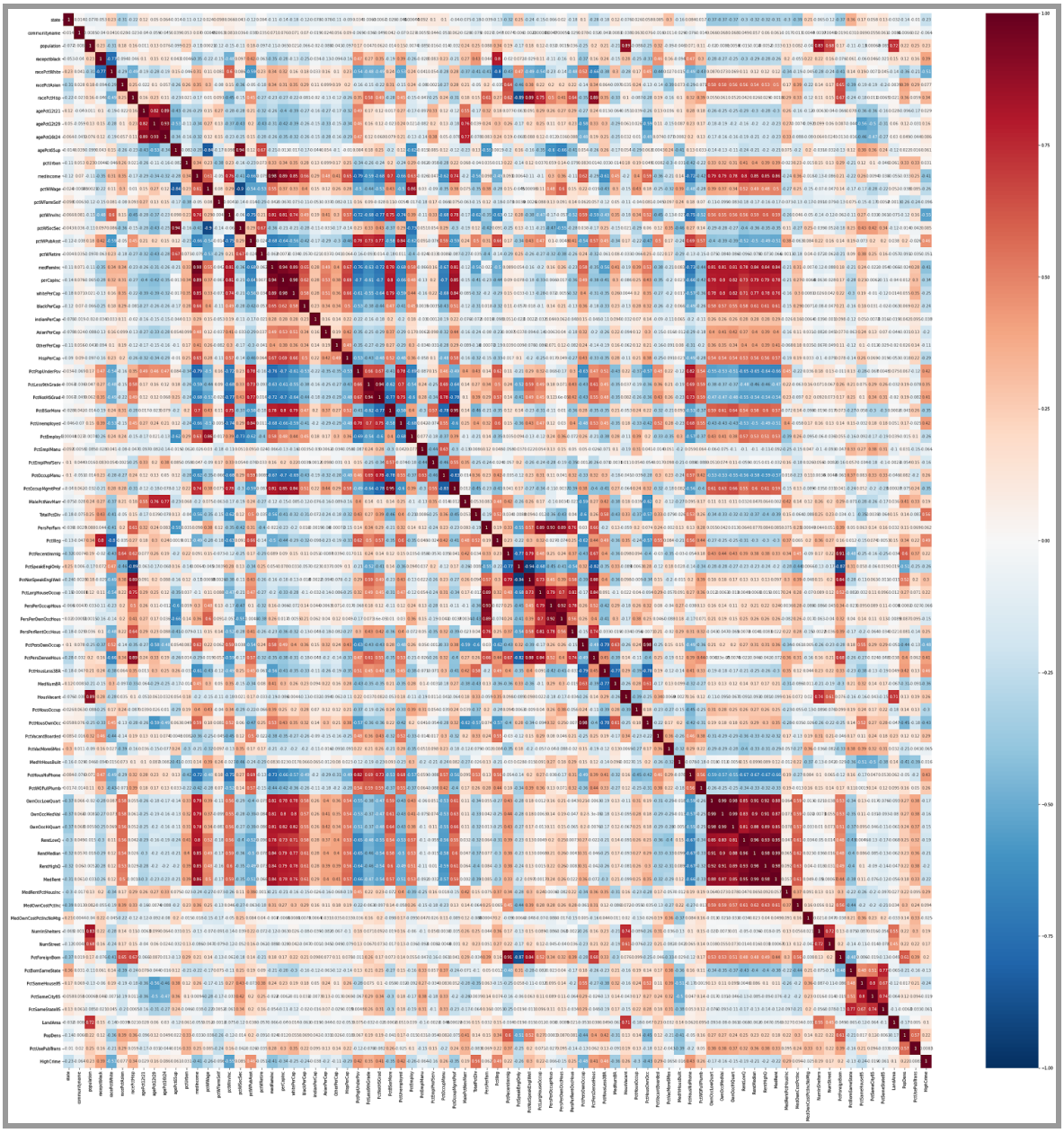


Correlation Matrix

With the help of the Correlation matrix we identify the attributes which show high dependency with Violent Crimes per Population.

```
[ ]  c=text.iloc[:,:]
     features = list(c)
     features.remove("ViolentCrimesPerPop")
     X=text[features]
     correlation = X.corr()
     plt.figure(figsize=(50, 50))
     sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu_r")
```

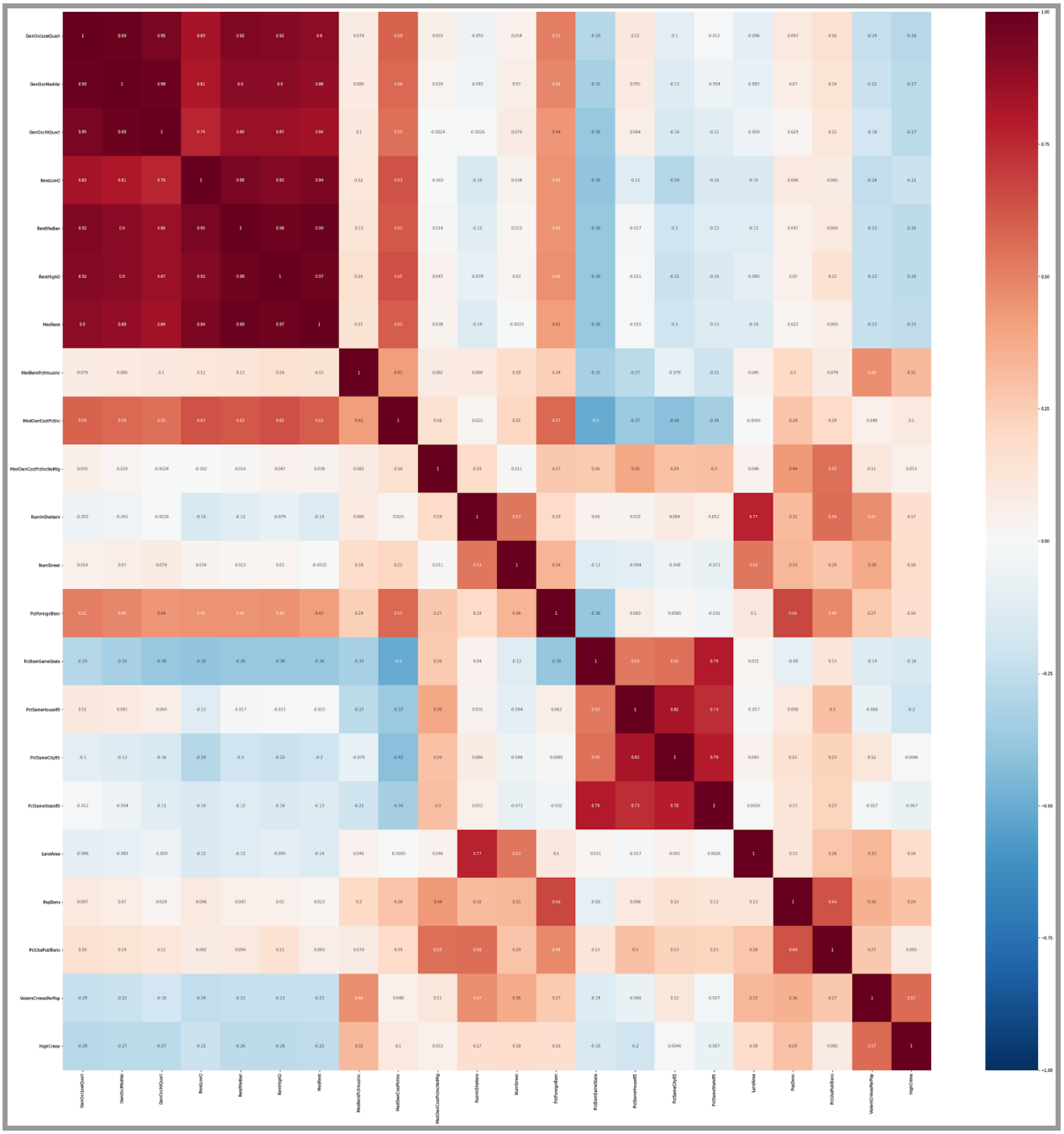
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2faa961b38>
```



Looking for the highly correlated attributes:

```
[ ]  c=text.iloc[:,60:]
     correlation = c.sample(100).corr()
     plt.figure(figsize=(300, 200))
     sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu_r")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2fb0d7a278>
```



```
[ ] y=text.sort_values('ViolentCrimesPerPop')
    y
```

	state	communityname	population	racepctblack	racePctWhite	racePctAsian	racePctHisp	agePct12t21	agePct12t29	agePct16t24	agePct65up
1230	36	1155	0.01	0.16	0.82	0.04	0.10	0.42	0.56	0.37	0.47
342	25	654	0.00	0.24	0.73	0.16	0.10	0.76	0.81	0.72	0.03
773	36	1179	0.01	0.01	0.97	0.03	0.03	0.53	0.62	0.55	0.52
529	44	1699	0.00	0.01	0.99	0.02	0.02	0.30	0.40	0.23	0.58
426	42	344	0.01	0.01	0.99	0.02	0.01	0.33	0.43	0.19	0.20
...	...	...	...	...	...	...	...	...	...	...	...
1208	34	280	0.03	1.00	0.00	0.08	0.20	0.40	0.51	0.33	0.39
82	48	1667	0.00	0.21	0.65	0.04	0.28	0.44	0.42	0.30	0.65
1119	1	136	0.04	1.00	0.10	0.00	0.00	0.43	0.45	0.30	0.58
998	6	888	0.08	0.46	0.00	0.13	1.00	0.65	0.72	0.50	0.12
362	1	148	0.41	1.00	0.02	0.03	0.01	0.41	0.50	0.33	0.48

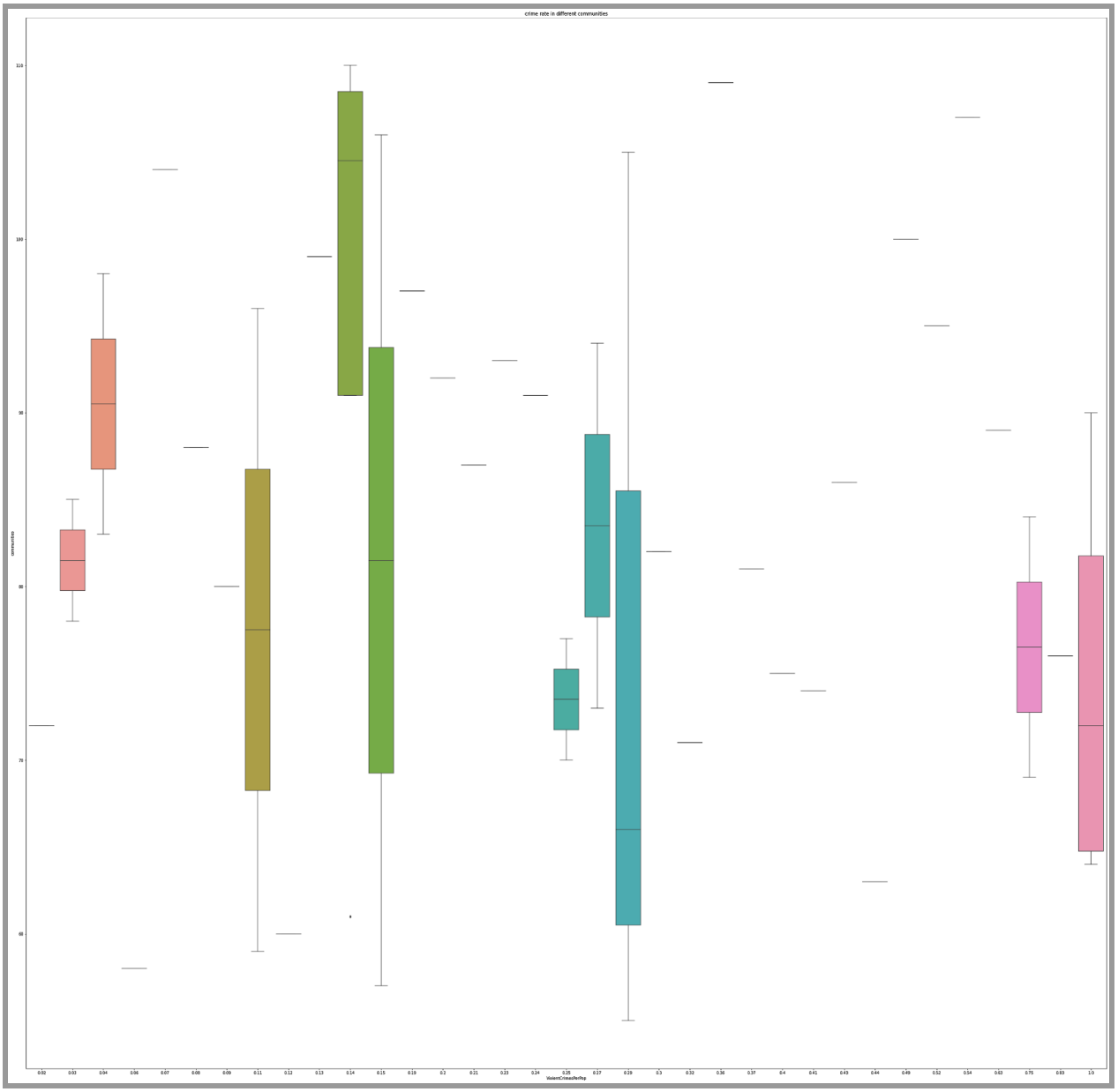
1707 rows x 82 columns

Box and Whiskers Plot Representing Crime Rates in different Communities

```
[ ] c=text.iloc[50:100,:]
```

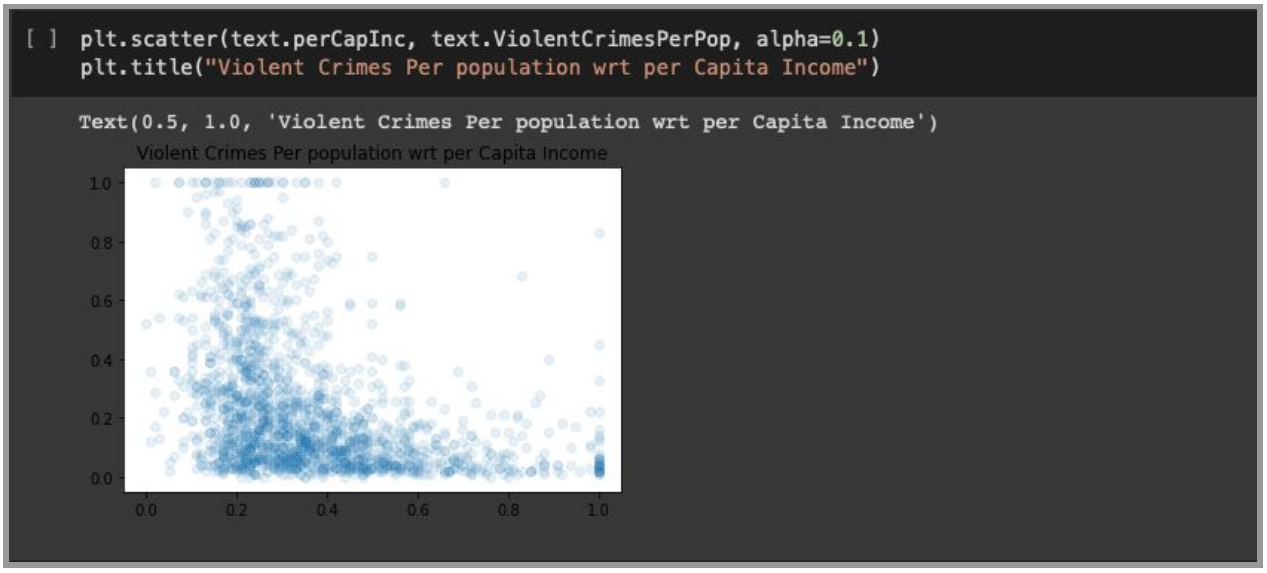
```
[ ] plt.figure(figsize=(300, 200))
    kx = sns.boxplot(x='ViolentCrimesPerPop', y="communityname", data = c)
    kx.set(xlabel='ViolentCrimesPerPop', ylabel='communities', title='crime rate in different communities')
```

```
[Text(0, 0.5, 'communities'),
 Text(0.5, 0, 'ViolentCrimesPerPop'),
 Text(0.5, 1.0, 'crime rate in different communities')]
```



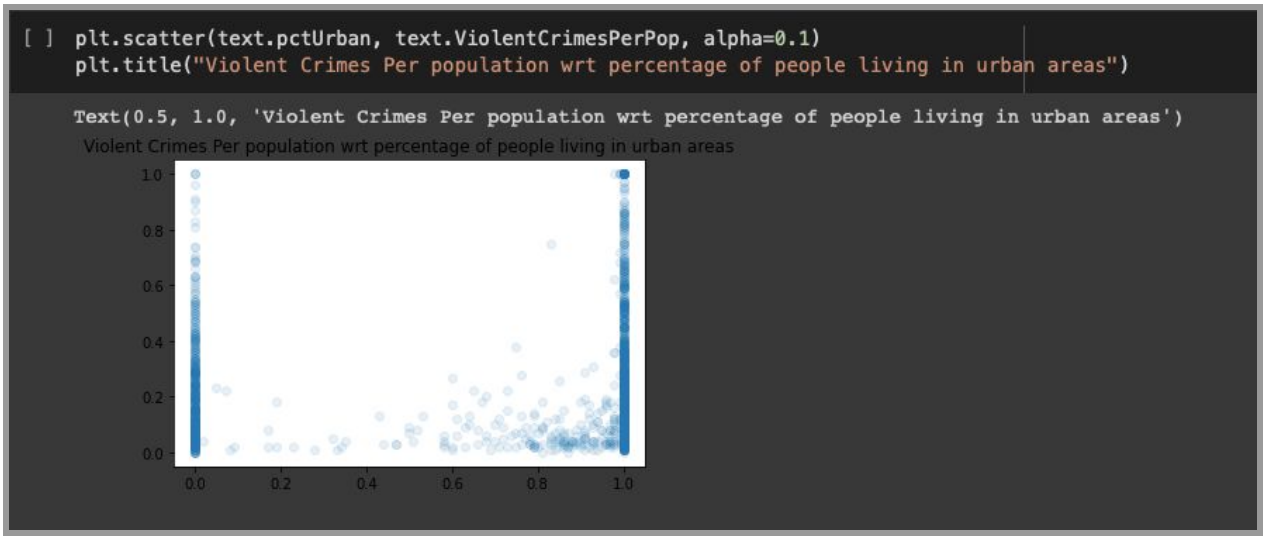

```
[ ]  len(text.communityname.unique())

     1707
```

Violent Crimes per population v/s per Capita Income (Scatter Plot)



This shows that the lower class and lower-middle class population are more involved in violent crime

Violent Crimes Per population v/s percentage of people living in urban areas (Scatter Plot)



This shows that urban areas with very low and very high populations have significantly high violent crime rates.

On the Basis of Age:

Violent Crimes Per population v/s population of 12-21 yr old (Scatter Plot)

This shows that communities having populations of 12-21yr old between 20-60% have very low violent crime rates.

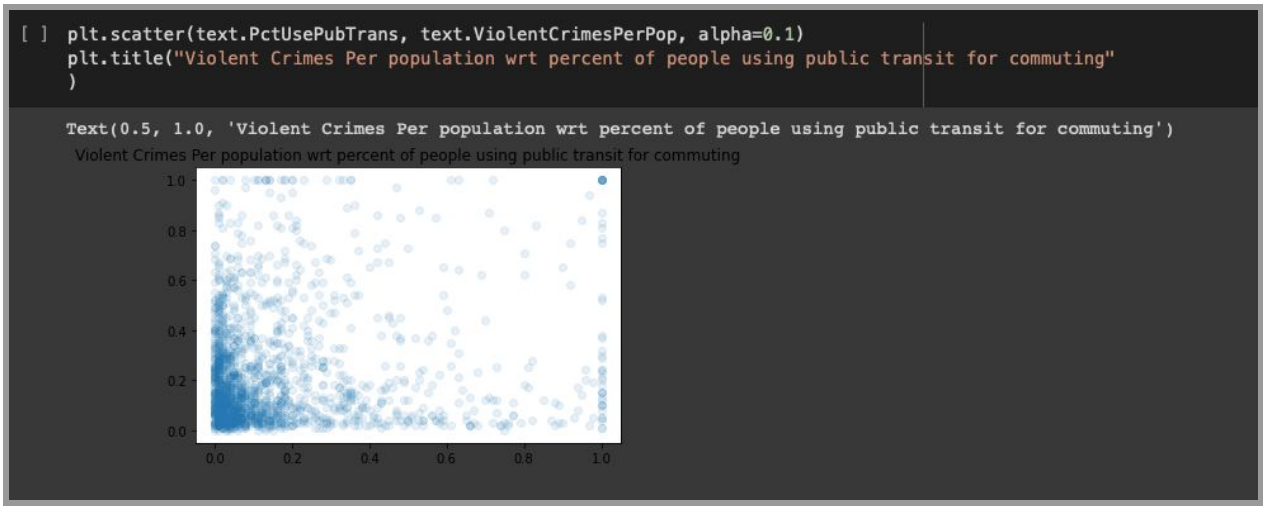
Violent Crimes Per population v/s population of 16-24 yr old (Scatter Plot)

```
[ ]  plt.scatter(text.agePct16t24, text.ViolentCrimesPerPop, alpha=0.1)
     plt.title("Violent Crimes Per population wrt percentage of population that is 16-24 in age")

     Text(0.5, 1.0, 'Violent Crimes Per population wrt percentage of population that is 16-24 in age')
```

This shows that communities having populations of 12-21yr old between 10-50% have very low violent crime rates.

Violent Crimes per population v/s population using public transit for commuting (Scatter Plot)

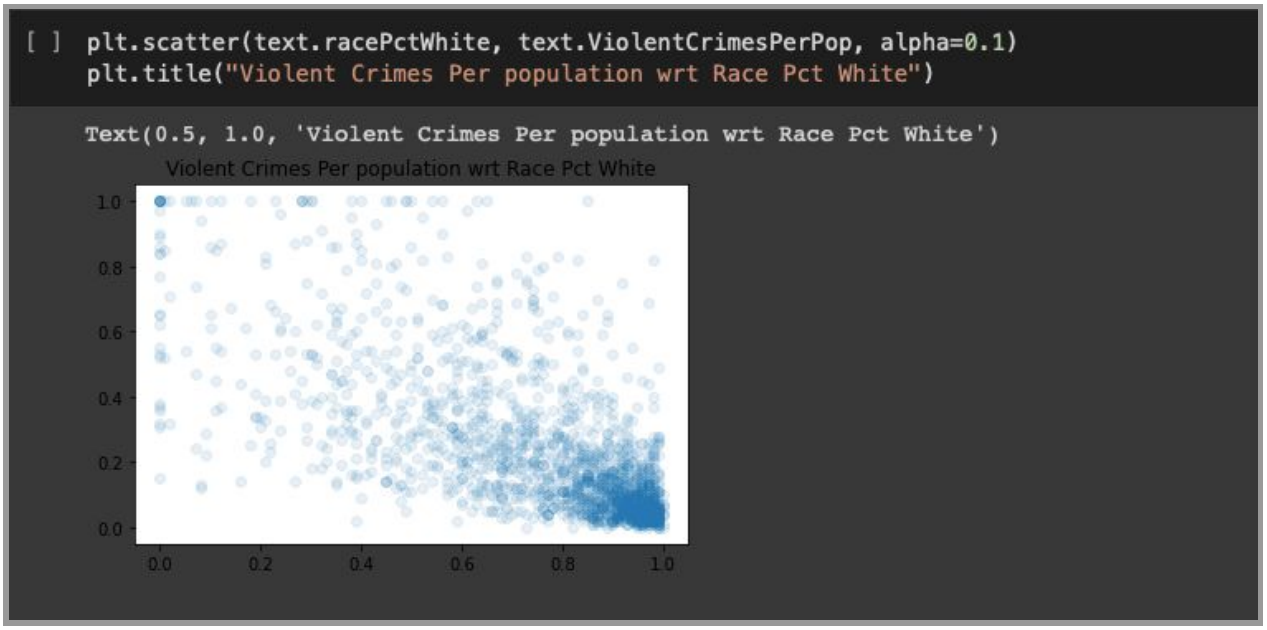
```
[ ]  plt.scatter(text.PctUsePubTrans, text.ViolentCrimesPerPop, alpha=0.1)
     plt.title("Violent Crimes Per population wrt percent of people using public transit for commuting"
     )

     Text(0.5, 1.0, 'Violent Crimes Per population wrt percent of people using public transit for commuting')
```



This shows that when there is a low percentage of population using public transit for commuting in the community, violent crime decreases very significantly.

On the Basis of Race:

Violent Crimes per population v/s percentage of white Race(Scatter Plot)

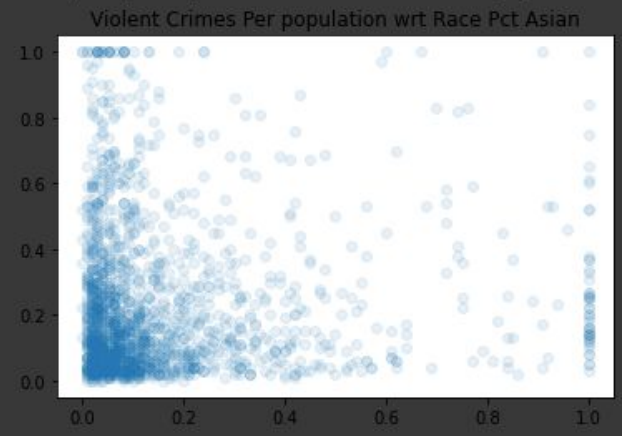


This shows that when there is a high percentage of white people in the community, violent crime decreases very significantly.

Violent Crimes per population v/s percentage of Asian Race(Scatter Plot)

```
[ ]  plt.scatter(text.racePctAsian, text.ViolentCrimesPerPop, alpha=0.1)
     plt.title("Violent Crimes Per population wrt Race Pct Asian")
```

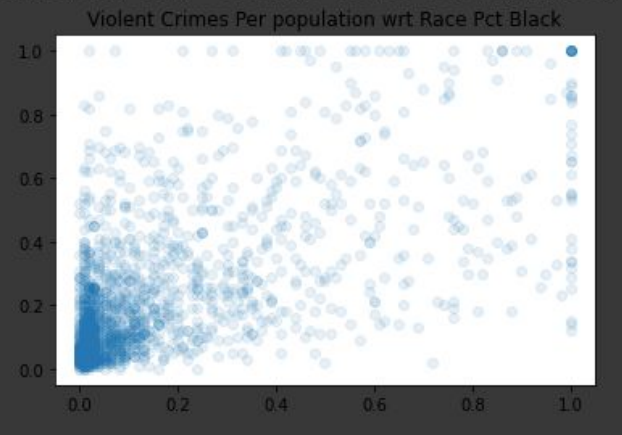
```
Text(0.5, 1.0, 'Violent Crimes Per population wrt Race Pct Asian')
```



Violent Crimes per population v/s percentage of African American(Scatter Plot)

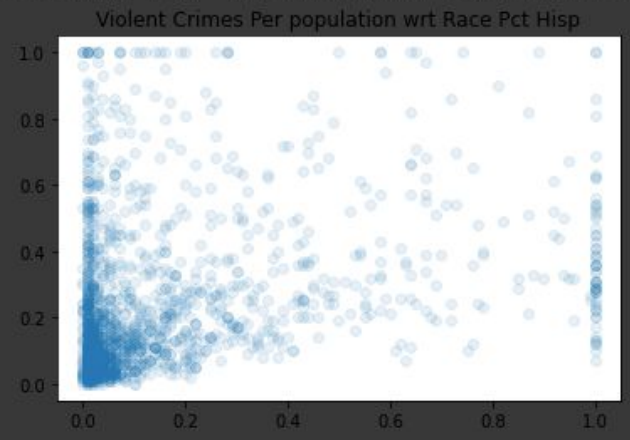
```
[ ]  plt.scatter(text.racepctblack, text.ViolentCrimesPerPop, alpha=0.1)
     plt.title("Violent Crimes Per population wrt Race Pct Black")
```

```
Text(0.5, 1.0, 'Violent Crimes Per population wrt Race Pct Black')
```



Violent Crimes per population v/s percentage of Hispanics(Scatter Plot)

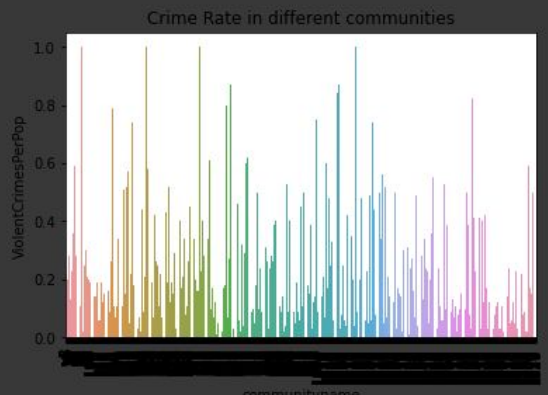
```
[ ]  plt.scatter(text.racePctHisp, text.ViolentCrimesPerPop, alpha=0.1)
     plt.title("Violent Crimes Per population wrt Race Pct Hisp")

     Text(0.5, 1.0, 'Violent Crimes Per population wrt Race Pct Hisp')
```



This shows that when there is a low percentage of Asians, African American, Hispanics in the community, violent crime decreases very significantly.

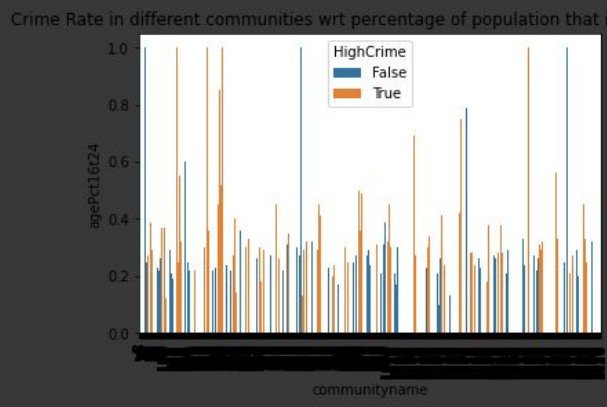
Bar Graph Representing Violent Crime in Different Communities:

```
[ ] sns.barplot(x='communityname', y='ViolentCrimesPerPop', data=text)
    plt.xticks(None,None,rotation=90)
    t=plt.title('Crime Rate in different communities')
```

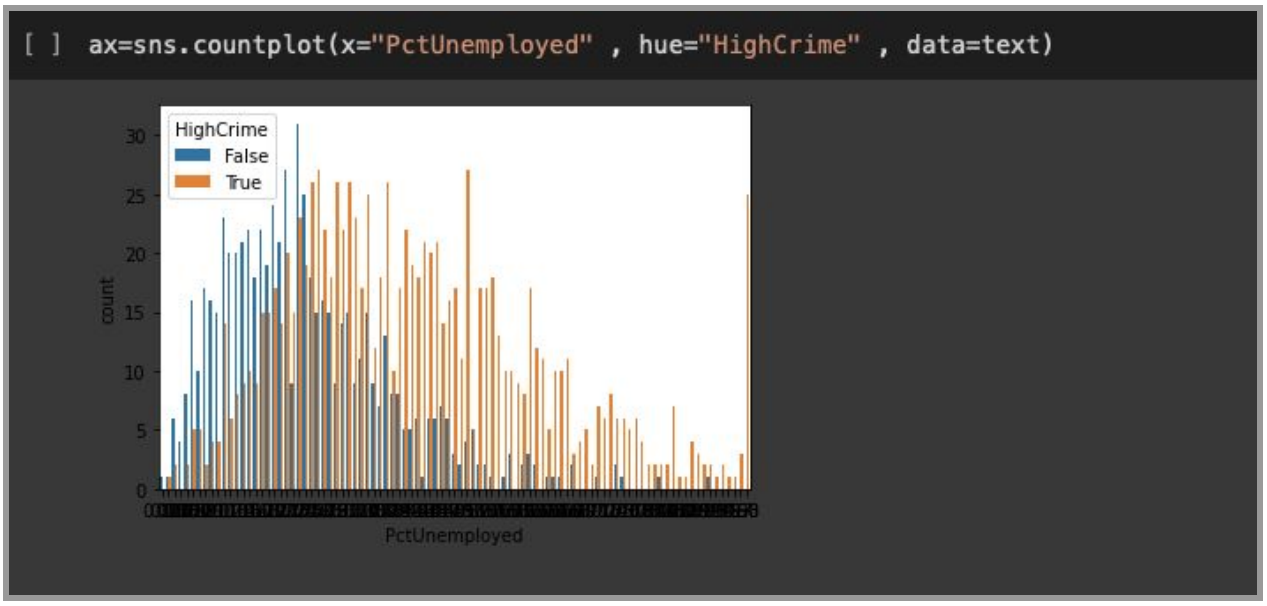


Bar Graph Representing Crime Rate in different Communities with respect to Percentage of Population in Age Group 16-24:

```
[ ] sns.barplot(x='communityname', y='agePct16t24', hue='HighCrime',data=text)
    plt.xticks(None,None,rotation=90)
    t=plt.title('Crime Rate in different communities wrt percentage of population that is 16-24 in age')
```



Bar Graph Representing Crime Rate vs Unemployment:



This shows that when there is a low percentage of Unemployed in the community, crime tends to be on the lower side as the bar graph shows the HighCrime attribute is False .

Conclusion:

We have chosen the Dataset regarding Crimes in different communities in the US. To draw inferences from this Dataset, we preprocessed the data - that is, **we eliminated the missing values, dealt with the null values, handled the Categorical attributes, dropped duplicate values and the values that were less correlated, and performed feature aggregation** and eventually drew inferences (or conclusions) regarding - **violent crimes and income, how crimes take place in rural and urban areas, the relation between age groups and how crime is prominent in a specific age group, crimes in different races and communities and how employment affects crime rate.**

References

- Stack Overflow : https://stackoverflow.com
- UpGrad : https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/
- Medium - TowardsDataScience : https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825
- ScienceDirect : https://www.sciencedirect.com/topics/engineering/data-preprocessing