



Big Data in Law Enforcement 2

ENDTERM REPORT

Group: CS-2119

Member:

Aydar Amangeldy

## Introduction

The chosen dataset is US\_Crime\_Rates\_1960\_2014 from Kaggle website. This dataset is highly relevant to law enforcement, since it contains types of crimes occurred in USA from 1960 to 2014. By identifying patterns in the given dataset, we can project potential issues and improve law enforcement. The goal of the project is to identify key trends and patterns which will be helpful in decreasing the crime rate.

The link to Github: [https://github.com/P1zzApple/bdle2\\_endterm](https://github.com/P1zzApple/bdle2_endterm)

## Data Preparation

1. Download dataset from Kaggle (<https://www.kaggle.com/datasets/mahmoudshogaa/us-crime-rates-1960-2014?resource=download>)
2. Open Jupyter Notebook
3. Extract dataset table in Jupyter(photo below)

```
In [34]: import pandas as pd
import numpy as np

df = pd.read_csv('US_Crime_Rates_1960_2014.csv') # Load dataset
df.head(5) # Print table
```

Out[34]:

	Year	Population	Total	Violent	Property	Murder	Forcible_Rape	Robbery	Aggravated_assault	Burglary	Larceny_Theft	Vehicle_Theft
0	1960	17502175	356400	266400	3090700	8190	17190	137640	164320	512700	1889480	320200
1	1961	18298000	348800	260900	3186600	8780	17220	136670	156780	506600	1913000	336000
2	1962	18877900	3752200	301810	3430790	8830	17550	130860	164070	594300	2083600	360800
3	1963	188483000	410800	310970	3753000	9640	17900	136470	174210	1306400	2297000	408300
4	1964	191143000	458400	364230	4200430	10360	21420	130390	203000	1313200	2514480	475800

## Analysis

After the preparation, we will try to look to trends and changes in this dataset.

To start, we can use `df.describe()` function to get basic data out of table:

```
In [3]: df.describe()
```

Out[3]:

	Year	Population	Total	Violent	Property	Murder	Forcible_Rape	Robbery	Aggravated_assault	Burglary	Lz
count	55.00000	5.500000e+01	5.500000e+01	5.500000e+01	5.500000e+01	55.000000	55.000000	55.000000	5.500000e+01	5.500000e+01	5
mean	1987.00000	2.401500e+08	1.062151e+07	1.188805e+06	5.612429e+06	1.731723e+05	72714.800000	410449.800000	6.811810e+05	3.424017e+06	5
std	16.02082	6.166216e+07	3.175831e+06	4.587107e+05	2.738812e+06	4267.442854	26278.000000	130306.400000	2.993225e+05	1.259004e+06	1
min	1960.00000	1.793202e+06	3.384209e+05	2.664600e+05	3.095700e+06	8030.000000	17190.000000	136570.000000	1.543200e+05	9.121900e+05	1
25%	1973.00000	2.106215e+08	9.086888e+05	9.252150e+05	8.060014e+06	14819.000000	33400.000000	354911.000000	4.384300e+05	2.105338e+06	4
50%	1987.00000	2.422829e+08	1.140151e+07	1.322380e+06	1.018299e+07	17030.000000	84230.000000	423057.000000	7.412291e+05	2.323980e+06	5
75%	2000.00000	2.833697e+08	1.308449e+07	1.432762e+06	1.160299e+07	20561.000000	52930.000000	512157.000000	9.002180e+05	3.073600e+06	7
max	2014.00000	3.188871e+08	1.487230e+07	1.932275e+06	1.296110e+07	24700.000000	103060.000000	687731.000000	1.133610e+06	3.795330e+06	8

Try to get in which years there were minimum/maximum total crimes. We will try using NumPy to sort (despite this data already existing in previous line)

```
In [3]: import numpy as np

total_arr = df['Total'].values
print(total_arr) # total arr
min = np.min(total_arr)
max = np.max(total_arr)
print(min, max) # min and max total

[ 3384280  3488900  3752200  4189500  4564600  4735400  5223500  5503400
  6720200  7418900  8098000  8508200  8248800  8718100  10253400  11292400
  11349700  10984500  11209000  11249500  13408300  13423800  12974400  12189600
  11881800  12431400  13211800  13508700  13923100  14251400  14475200  14872000
  14430200  14144800  13909500  13052700  13433953  13154571  13475634  11634378
  11688972  11370609  11878954  11828538  11679474  11565499  11401311  11251828
  11149543  10762956  10363873  10258774  10218050  9850445  9473816]
3384280 14872900
```

Use SQLite3 to find years via sql commands

```
In [7]: import sqlite3
conn = sqlite3.connect('and.db')
cursor = conn.cursor()

#df.to_sql("ds", conn)
cursor.execute(f'''
    SELECT * from ds WHERE Total = {min}
''')
#print(df)
res = cursor.fetchone()
print(f'Year: {res[0]}, Population: {res[1]}, Total crimes: {min}')
cursor.execute(f'''
    SELECT * from ds WHERE Total = {max}
''')
res = cursor.fetchone()
print(f'Year: {res[0]}, Population: {res[1]}, Total crimes: {max}')

Year: 1960, Population: 179323175, Total crimes: 3384280
Year: 1991, Population: 252177000, Total crimes: 14872900
```

Next, let's try to use this on all types of crimes. The command outputs the crime type, year with the minimum/maximum amount of said crimes.

```
In [9]: crime_type = ['Violent', 'Property', 'Murder', 'Forcible_Rape', 'Robbery', 'Aggravated_assault', 'Burglary', 'Larceny_Theft', 'Vehicle_Theft']
for i in crime_type:
    crime = df[i].values
    min = np.min(crime)
    max = np.max(crime)
    cursor.execute(f'''SELECT * from ds WHERE {i} = {min}''')
    res1 = cursor.fetchone()
    cursor.execute(f'''SELECT * from ds WHERE {i} = {max}''')
    res2 = cursor.fetchone()
    print(f'{i}: min: {min}, {res1[0]} | max: {max}, {res2[0]}') # crime type, min and max (with years)

Violent: min: 288460, 1960 | max: 1052270, 1991
Property: min: 3095700, 1960 | max: 12961100, 1991
Murder: min: 8530, 1962 | max: 24700, 1991
Forcible_Rape: min: 17100, 1960 | max: 100000, 1992
Robbery: min: 100670, 1961 | max: 607730, 1991
Aggravated_assault: min: 154320, 1960 | max: 1135610, 1993
Burglary: min: 512100, 1960 | max: 3795200, 1980
Larceny_Theft: min: 105400, 1960 | max: 8142200, 1991
Vehicle_Theft: min: 328200, 1960 | max: 1661700, 1991
```

Furthermore, we will try to define top of committed crimes using Apache Spark.

```
In [32]: from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, avg, desc, split, explode

spark = SparkSession.builder.appName("Trutans").getOrCreate()

rdf = spark.read.option("header", "true").csv("US_Crime_Rates_1960_2014.csv")
rdf = rdf.withColumn("Crime coefficient", col("Total") / col("Population"))
rdf = rdf.orderBy(col("Crime coefficient").desc())
top = rdf.limit(5)
top.select("Year", "Crime coefficient").show()
```

```
-----
[Year]  Crime coefficient
-----
[1989]  0.05050006339055414
[1991]  0.05097891948790381
[1981]  0.053581064817205726
[1990]  0.05828275578686814
[1989]  0.05748996081130788
```

```
In [34]: rdf = rdf.orderBy("Crime coefficient")
antstop = rdf.limit(5)
antstop.select("Year", "Crime coefficient").show()

spark.stop()
```

```
-----
[Year]  Crime coefficient
-----
[1960]  0.018872872725688864
[1961]  0.019666942554865786
[1962]  0.020197085692061752
[1963]  0.021893827328724606
[1964]  0.02588880057668553
```

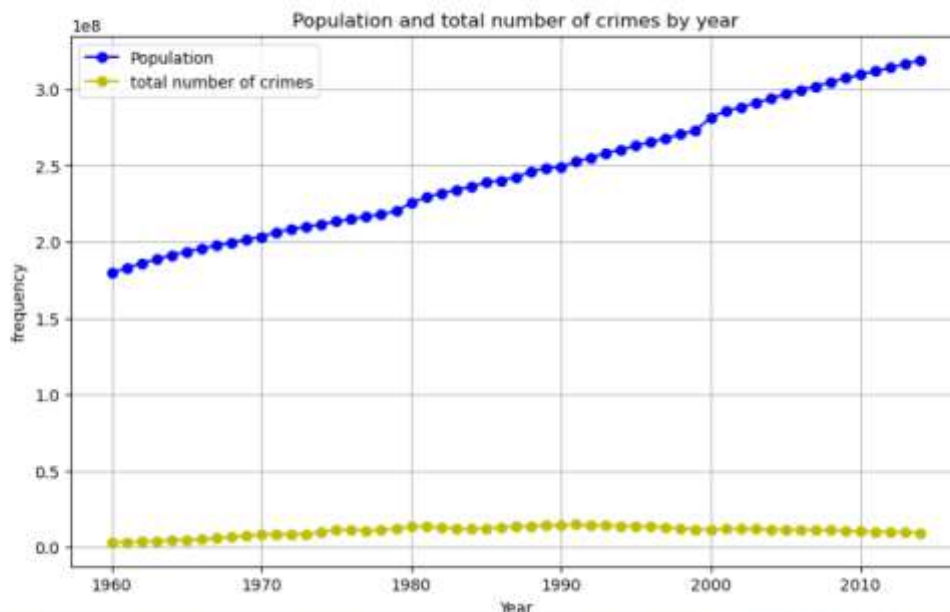
Lastly, to top it off, we can visualize data by using Matplotlib. Visualize population and total crimes.

```
In [14]: # Visualize the data
years = df['Year']
population = df['Population']
total = df['Total']

plt.figure(figsize=(18, 6))
plt.plot(years, population, color='b', marker='o')
plt.plot(years, total, label='total number of crimes', color='y', marker='o')

plt.xlabel('Year')
plt.ylabel('frequency')
plt.title('Population and total number of crimes by year')
plt.legend()
plt.grid(True)

plt.show()
```



Conclusion

To wrap up, the time of most crimes occurred were during the start of the 90s, the crimes coefficient is low compared to the population but this situation must be always cautiously monitored.