DC Crime Data Analysis

May 29, 2021

*__Work Distribution:__
Hanbin Liu(11912410): Task 1, 2, 3, 5
Zhuoyuan Ma(11912412): Task 4 and integration

1 0 Backgroud Introduction

Crime is a common social problem in modern human society. It has to do with economics, culture, politics, technology and people's happiness.Based on crime data in Washington, D.C. from 2008 to 2021, we will use data mining techniques to uncover a lot of potential information.

On February 1 2020, the methodology of geography assignments of crime data was modified to increase accuracy. From January 1 2020 going forward, all crime data will have Ward, ANC, SMD, BID, Neighborhood Cluster, Voting Precinct, Block Group and Census Tract values calculated prior to, rather than after, anonymization to the block level. This change impacts approximately one percent of Ward assignments.

After data analysis, we find some interesting conclusions. The whole report includes data statistics

and visualization, classification and clustering, housing price prediction, model evaluation, additional data analysis and conclusions.

2 1 Other relevant data(Task 1)

@Task 1: Besides the datasets we provide you here, you can also find other relevant datasets by yourself, if you think these datasets are helpful for your analysis.

District map: https://mpdc.dc.gov/page/police-districts-and-police-service-areas

Ward map: https://planning.dc.gov/whatsmyward

ATM Banking ATM banking locations in the District of Columbia. The Chief Technology Office (OCTO) has captured the majority of ATM banking locations in the District of Columbia. https://opendata.dc.gov/datasets/DCGIS::atm-banking/about

3 2 Data statistics and visualization(Task 2)

3.0.1 2-1 Data preprocessing and visualization(Task 2-1)

@task:You are asked to first do the data preprocessing for all the data (include the relevant data you found). There may be redundancy in the data so that you have to process it at the beginning. Then visualize the data using python package 'matplotlib' and 'seaborn'. For instance, the histograms across the attributes 'offense', 'year', etc. You can also plot the histograms based on the geographic info. Based on these plots, can you find any interesting relations and draw any conclusions?

2-1-1 Basic information

[1]:
```
import os
import json
import pandas as pd
import numpy as np
import warnings
import matplotlib as mpl
import missingno as msno
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
warnings.filterwarnings("ignore")
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity='all'
```

[2]:
```
crime=pd.read_csv('DC_Crime.csv')
crime
```

[2]:
```
       NEIGHBORHOOD_CLUSTER  CENSUS_TRACT offensegroup  LONGITUDE  \
0                cluster 21        8702.0     property -77.003574
1                cluster 16        1600.0     property -77.026557
```

```
2                cluster 8        4702.0     property -77.020913
3               cluster 31        7808.0     property -76.919601
4               cluster 39       10900.0     property -77.003927
…                    …           …           …        …
449198          cluster 22       11100.0     property -76.977167
449199           cluster 6        5201.0     property -77.035546
449200          cluster 23        8802.0     property -76.982015
449201          cluster 23        8802.0     property -76.990857
449202           cluster 3        3400.0     property -77.017297

                     END_DATE         offense-text     SHIFT    YBLOCK  \
0       2017-04-29T08:00:23.000         theft f/auto       day  138139.0
1       2017-04-29T08:30:37.000         theft f/auto       day  146051.0
2       2017-04-29T11:10:57.000          theft/other       day  137185.0
3       2017-04-28T09:30:33.000          theft/other       day  135903.0
4       2017-04-29T13:42:11.000          theft/other       day  128340.0
…                           …                   …       …       …
449198  2021-03-17T20:20:07.000          theft/other   evening  139268.0
449199  2021-03-17T17:20:33.000  motor vehicle theft   evening  137976.0
449200  2021-03-17T23:09:20.000          theft/other  midnight  137150.0
449201  2021-03-17T22:40:24.000  motor vehicle theft  midnight  137569.0
449202  2021-03-25T13:00:28.000         theft f/auto       day  138891.0

        DISTRICT  WARD  …                                        BLOCK  \
0            5.0   5.0  …                150 - 299 block of q street ne
1            4.0   4.0  …       7600  - 7699 block of georgia avenue nw
2            1.0   6.0  …                600 - 699 block of k street nw
3            6.0   7.0  …           5715 5739  block of blaine street ne
4            7.0   8.0  …     4610 - 4659 block of south capitol street
…          …   …   …  …                                          …
449198       5.0   5.0  …         1815 - 1999 block of bryant street ne
449199       2.0   2.0  …              1500 - 1599 block of p street nw
449200       5.0   5.0  …        900 - 999 block of bladensburg road ne
449201       5.0   5.0  …  1300 - 1399 block of west virginia avenue ne
449202       3.0   1.0  …           300 - 399 block of oakdale place nw

                    START_DATE       CCN              OFFENSE  \
0       2017-04-29T01:30:14.000  17070672         theft f/auto
1       2017-04-29T02:30:10.000  17070675         theft f/auto
2       2017-04-29T10:43:33.000  17070714          theft/other
3       2017-04-28T09:15:27.000  17070736          theft/other
4       2017-04-29T13:03:40.000  17070780          theft/other
…                           …       …                   …
449198  2021-03-17T20:15:08.000  21034375          theft/other
449199  2021-03-16T21:00:41.000  21034386  motor vehicle theft
449200  2021-03-17T22:21:45.000  21034407          theft/other
449201  2021-03-17T22:38:11.000  21034425  motor vehicle theft
```

```
449202  2021-03-18T18:00:05.000  21037812         theft f/auto

        OCTO_RECORD_ID  ANC                REPORT_DAT  METHOD  \
0          17070672-01   5E  2017-04-29T13:49:31.000Z  others
1          17070675-01   4A  2017-04-29T14:38:59.000Z  others
2          17070714-01   6E  2017-04-29T15:19:02.000Z  others
3          17070736-01   7C  2017-04-29T16:11:44.000Z  others
4          17070780-01   8D  2017-04-29T18:17:15.000Z  others
…                  …  …                        …     …
449198     21034375-01   5C   2021-03-17T21:07:30.000  others
449199     21034386-01   2B   2021-03-17T21:38:51.000  others
449200     21034407-01   5D   2021-03-18T01:12:46.000  others
449201     21034425-01   5D   2021-03-18T00:57:07.000  others
449202     21037812-01   1B   2021-03-25T14:41:36.000  others

                                      location   LATITUDE
0       38.911121322949178,-77.003576581965632  38.911114
1       38.982391883146363,-77.026559339798794  38.982384
2       38.902525540064957,-77.020915170313728  38.902518
3       38.890951021927407,-76.919603310082607  38.890943
4       38.822847890448664,-77.003929146312586  38.822840
…                                          …       …
449198     38.9212817635688,-76.9771673661107  38.921282
449199     38.9096398205834,-77.0355462229667  38.909640
449200     38.9022029472732,-76.9820154636605  38.902203
449201      38.905978473251,-76.9908573768314  38.905978
449202     38.9178865750065,-77.0172966263538  38.917887

[449203 rows x 29 columns]
```

*NOTE: Carefully observing the data, we can infer that the END_DATE and START_DATE of adjacent samples should be in the same month of the same year, or even the same day. We browsed the DC_Crime.csv file and verified the conjecture. This conclusion is used in subsequent missing values dealing.

[3]:
```
# information of crime
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 449203 entries, 0 to 449202
Data columns (total 29 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   NEIGHBORHOOD_CLUSTER  443701 non-null  object
 1   CENSUS_TRACT          448011 non-null  float64
 2   offensegroup          449203 non-null  object
 3   LONGITUDE             449203 non-null  float64
 4   END_DATE              422407 non-null  object
```

```
 5   offense-text          449203 non-null  object
 6   SHIFT                 449203 non-null  object
 7   YBLOCK                449203 non-null  float64
 8   DISTRICT              448991 non-null  float64
 9   WARD                  449192 non-null  float64
 10  YEAR                  449203 non-null  int64
 11  offensekey            449203 non-null  object
 12  BID                   74761 non-null   object
 13  sector                448960 non-null  object
 14  PSA                   448960 non-null  float64
 15  ucr-rank              449203 non-null  int64
 16  BLOCK_GROUP           448011 non-null  object
 17  VOTING_PRECINCT       449135 non-null  object
 18  XBLOCK                449203 non-null  float64
 19  BLOCK                 449202 non-null  object
 20  START_DATE            449189 non-null  object
 21  CCN                   449203 non-null  int64
 22  OFFENSE               449203 non-null  object
 23  OCTO_RECORD_ID        449203 non-null  object
 24  ANC                   449203 non-null  object
 25  REPORT_DAT            449203 non-null  object
 26  METHOD                449203 non-null  object
 27  location              449203 non-null  object
 28  LATITUDE              449203 non-null  float64
dtypes: float64(8), int64(3), object(18)
memory usage: 99.4+ MB
```

Basic information of numerical features.

[4]:
```
crime.describe()
```

[4]:

	CENSUS_TRACT	LONGITUDE	YBLOCK	DISTRICT	\
count	448011.000000	449203.000000	449203.000000	448991.000000	
mean	6221.493905	-77.008255	137621.796787	3.717173	
std	3145.776703	0.036310	3463.677183	1.920224	
min	100.000000	-77.114142	127300.000000	1.000000	
25%	3500.000000	-77.031978	136006.000000	2.000000	
50%	7000.000000	-77.013177	137621.000000	3.000000	
75%	8904.000000	-76.985736	139707.000000	5.000000	
max	11100.000000	-76.910021	147441.000000	7.000000	

	WARD	YEAR	PSA	ucr-rank	\
count	449192.000000	449203.00000	448960.000000	449203.000000	
mean	4.432588	2014.06146	377.043440	5.946467	
std	2.358330	3.70223	191.595023	1.375372	
min	1.000000	2008.00000	101.000000	1.000000	
25%	2.000000	2011.00000	207.000000	6.000000	
50%	5.000000	2014.00000	308.000000	6.000000	

```
75%          6.000000    2017.00000     507.000000       7.000000
max          8.000000    2021.00000     708.000000       9.000000

                XBLOCK           CCN       LATITUDE
count   449203.000000  4.492030e+05  449203.000000
mean    399284.638732  1.417359e+07      38.906448
std       3149.181187  3.730647e+06       0.031201
min     390103.000000  5.370000e+03      38.813471
25%     397226.000000  1.109329e+07      38.891888
50%     398857.000000  1.413229e+07      38.906447
75%     401238.000000  1.715595e+07      38.925238
max     407806.000000  9.943899e+07      38.994901
```

Basic information of character features.

[5]:
```
crime['NEIGHBORHOOD_CLUSTER'].value_counts()
```

[5]:
```
cluster 2     35723
cluster 8     30045
cluster 25    25393
cluster 6     25020
cluster 26    20583
cluster 18    20579
cluster 39    19450
cluster 3     19265
cluster 23    17156
cluster 21    16221
cluster 7     15415
cluster 22    14869
cluster 4     12990
cluster 17    12374
cluster 31    12342
cluster 33    12205
cluster 1     11193
cluster 34    11078
cluster 32    10907
cluster 38     8425
cluster 11     7737
cluster 9      7686
cluster 5      7227
cluster 30     6396
cluster 19     6243
cluster 37     6076
cluster 24     6056
cluster 20     5334
cluster 28     5266
cluster 15     5119
cluster 36     4583
```

```
cluster 35     4522
cluster 14     3758
cluster 10     3714
cluster 27     3010
cluster 13     2977
cluster 12     2791
cluster 16     2580
cluster 29     1294
cluster 45       41
cluster 43       23
cluster 44       17
cluster 46        9
cluster 40        7
cluster 41        2
Name: NEIGHBORHOOD_CLUSTER, dtype: int64
```

[6]:
```
crime['offensegroup'].value_counts()
```

[6]:
```
property    373163
violent      76040
Name: offensegroup, dtype: int64
```

[7]:
```
crime['END_DATE'].value_counts()
```

[7]:
```
2008-05-16T00:00:00.000    39
2009-01-01T00:00:00.000    39
2008-10-17T00:00:00.000    37
2008-06-02T00:00:00.000    36
2008-08-22T00:00:00.000    36
                           ..
2020-05-19T19:21:56.000     1
2012-07-22T07:10:00.000     1
2019-10-19T16:45:29.000     1
2010-04-17T00:00:00.000     1
2014-02-14T08:30:00.000     1
Name: END_DATE, Length: 364178, dtype: int64
```

[8]:
```
crime['offense-text'].value_counts()
```

[8]:
```
theft/other                   175631
theft f/auto                  119119
robbery                        42215
motor vehicle theft            42159
burglary                       35898
assault w/dangerous weapon     28893
sex abuse                       3115
homicide                        1817
```

```
arson                              356
Name: offense-text, dtype: int64
```

[9]:
```
crime['SHIFT'].value_counts()
```

[9]:
```
evening     191978
day         169650
midnight     87575
Name: SHIFT, dtype: int64
```

[10]:
```
crime['offensekey'].value_counts()
```

[10]:
```
property|theft/other                 175631
property|theft f/auto                119119
violent|robbery                       42215
property|motor vehicle theft          42159
property|burglary                     35898
violent|assault w/dangerous weapon    28893
violent|sex abuse                      3115
violent|homicide                       1817
property|arson                          356
Name: offensekey, dtype: int64
```

[11]:
```
crime['BID'].value_counts()
```

[11]:
```
downtown                    24999
golden triangle              9345
georgetown                   8014
capitol hill                 7472
noma                         6342
southwest                    5010
adams morgan                 4125
capitol riverfront           3279
mount vernon triangle cid    2927
anacostia                    2240
dupont circle                1008
Name: BID, dtype: int64
```

[12]:
```
crime['sector'].value_counts()
```

[12]:
```
2D3    41977
3D2    29394
3D3    28020
1D2    26652
3D1    24556
5D2    23422
2D2    22775
6D1    21940
```

```
1D1    21596
4D1    20670
6D3    20097
5D3    19130
4D3    19088
4D2    18024
6D2    17479
5D1    16795
2D1    16409
7D2    16324
1D3    16067
7D1    15848
7D3    12697
Name: sector, dtype: int64
```

[13]:
```
crime['BLOCK_GROUP'].value_counts()
```

[13]:
```
005800 1    11608
004400 2     7208
010700 1     7133
010600 2     6806
008803 1     5233
            …
000701 2       93
000701 3       88
000801 1       66
007301 1       35
001002 1       13
Name: BLOCK_GROUP, Length: 450, dtype: int64
```

[14]:
```
crime['VOTING_PRECINCT'].value_counts()
```

[14]:
```
precinct 129    20278
precinct 17     14224
precinct 83      9518
precinct 72      9397
precinct 39      8684
                …
precinct 12       578
precinct 52       543
precinct 144      543
precinct 138      472
precinct 136      364
Name: VOTING_PRECINCT, Length: 144, dtype: int64
```

[15]:
```
crime['BLOCK'].value_counts()
```

[15]:
```
3100 - 3299 block of 14th street nw           3152
900 - 999 block of rhode island avenue ne     1769
1300 - 1699 block of connecticut avenue nw    1579
3200 - 3275 block of m street nw              1426
5300 - 5399 block of wisconsin avenue nw      1384
                                              …
t street ne and brentwood road ne                1
5918  - 5999 block of 31st place nw              1
madison street nw and 14th street nw             1
34th street se and d street se                   1
1798 - 1799 block of lanier place nw             1
Name: BLOCK, Length: 17458, dtype: int64
```

[16]:
```
crime['START_DATE'].value_counts()
```

[16]:
```
2015-08-23T20:00:00.000    20
2013-09-16T08:23:00.000    12
2014-05-17T23:00:00.000    11
2008-06-23T17:00:00.000    10
2010-10-19T12:00:00.000    10
                           ..
2010-08-14T02:30:00.000     1
2014-10-14T08:15:00.000     1
2018-09-15T15:33:28.000     1
2020-12-01T10:17:33.000     1
2008-05-30T14:55:00.000     1
Name: START_DATE, Length: 373066, dtype: int64
```

[17]:
```
crime['OFFENSE'].value_counts()
# most important feature
```

[17]:
```
theft/other                  175631
theft f/auto                 119119
robbery                       42215
motor vehicle theft           42159
burglary                      35898
assault w/dangerous weapon    28893
sex abuse                      3115
homicide                       1817
arson                           356
Name: OFFENSE, dtype: int64
```

[18]:
```
crime['OCTO_RECORD_ID'].value_counts()
```

[18]:
```
19073083-01    2
19133889-01    2
20132555-01    1
11138589-01    1
```

```
12054005-01    1
               ..
17202958-01    1
09061971-01    1
16000647-01    1
17088751-01    1
20171820-01    1
Name: OCTO_RECORD_ID, Length: 449201, dtype: int64
```

[19]:
```
crime['ANC'].value_counts()
```

[19]:
```
1B    26078
2B    25099
1A    22704
2C    18520
6B    18322
5C    17223
6C    15371
2F    15285
5E    15277
5D    15143
6A    14317
2E    13024
4C    12835
6E    12725
7C    12259
7D    11520
7F    11375
8C    11242
4B    11106
8A    10764
7B    10742
6D    10418
1C    10082
8B    10063
7E     9672
4A     9668
8E     9588
8D     9209
2A     8486
5B     7772
3E     7341
3C     6177
4D     5595
5A     5464
1D     4547
3G     3540
```

```
3F     3484
3D     3431
3B     2111
2D     1624
Name: ANC, dtype: int64
```

[20]:
```
crime['REPORT_DAT'].value_counts()
```

[20]:
```
2013-09-16T04:00:00.000Z    12
2008-10-02T04:00:00.000Z     8
2009-04-15T04:00:00.000Z     7
2010-09-22T19:00:00.000Z     7
2008-05-31T04:00:00.000Z     7
                            ..
2019-12-02T15:19:02.000Z     1
2008-10-19T15:21:00.000Z     1
2018-06-05T22:34:04.000Z     1
2012-03-30T06:15:00.000Z     1
2014-07-26T05:00:00.000Z     1
Name: REPORT_DAT, Length: 418756, dtype: int64
```

[21]:
```
crime['METHOD'].value_counts()
```

[21]:
```
others    408728
gun        26957
knife      13518
Name: METHOD, dtype: int64
```

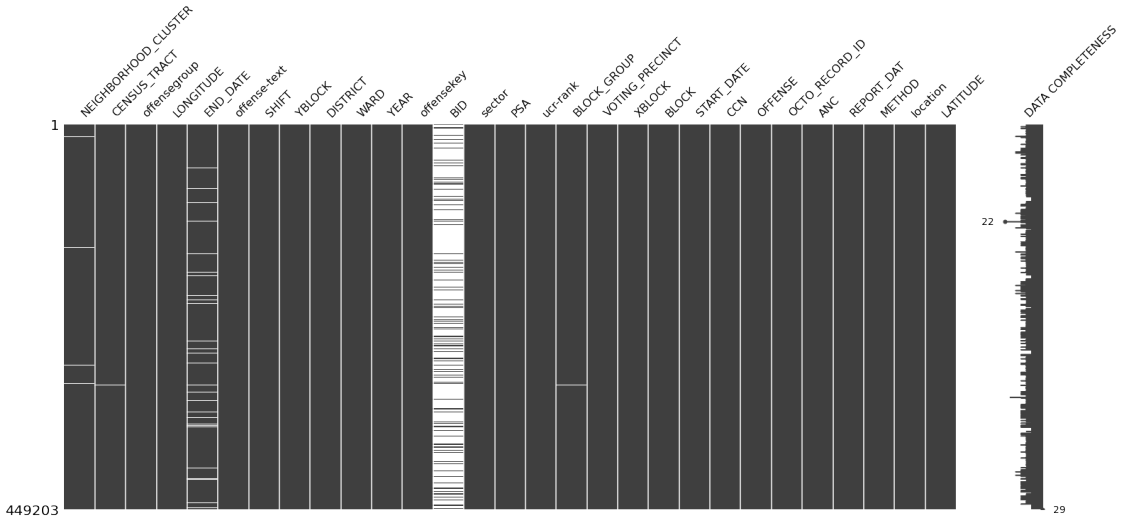
[22]:
```
crime['location'].value_counts()
```

[22]:
```
38.929520805242319,-77.03273284903338     2997
38.922219443509491,-76.993152403339593    1578
38.905158338230919,-77.063999928265474    1364
38.909933141882533,-77.043619521705949    1335
38.959915251272726,-77.085301891543978    1273
                                          …
38.9331089775265,-76.9738188502484           1
38.8296047498357,-77.0136244072696           1
38.9274185750513,-76.9955714717709           1
38.892806174510298,-77.048865377102388       1
38.8990211846567,-76.9295982853751           1
Name: location, Length: 23069, dtype: int64
```

2-1-2 Dealing With Missing Data Missing values visualization. The white line represents the missing value. The fringe width on the right represents the data integrity of a single sample.

[23]:
```
msno.matrix(crime, labels=True)
```

[23]: <AxesSubplot:>

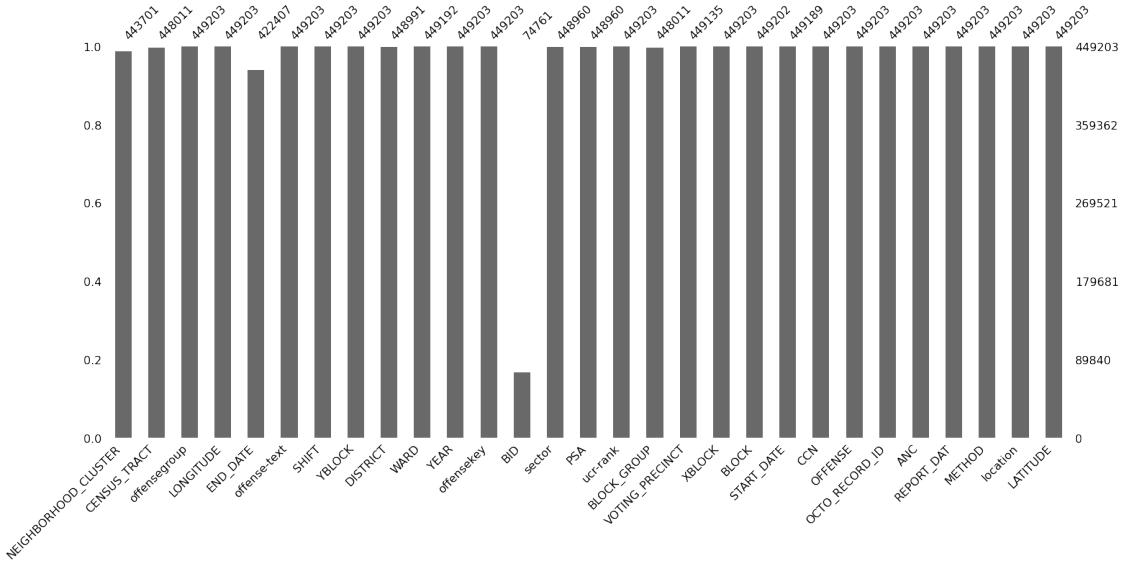


*NOTE: The BID column is almost white and is extremely missing.The missing of other features are negligible.

Missing proportion and quantity.

[24]:
```
msno.bar(crime)
```

[24]: <AxesSubplot:>



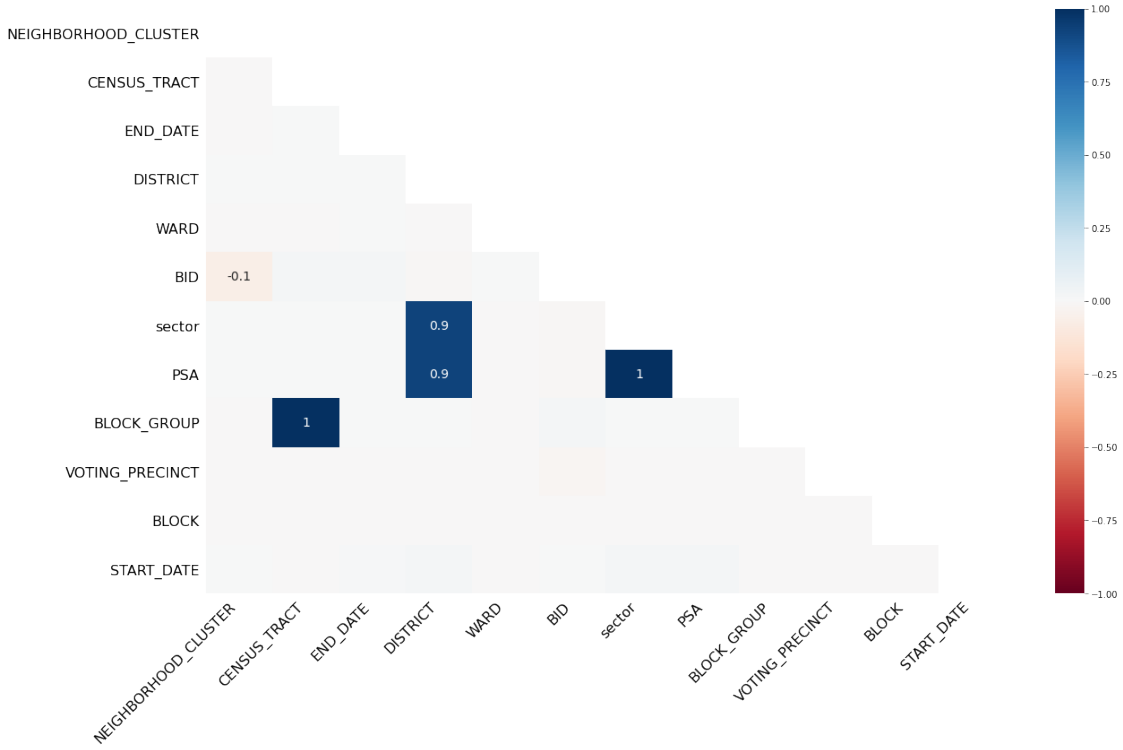
[25]:
```
crime.isnull().sum()
```

[25]:
```
NEIGHBORHOOD_CLUSTER      5502
CENSUS_TRACT              1192
offensegroup                 0
LONGITUDE                    0
END_DATE                 26796
offense-text                 0
SHIFT                        0
YBLOCK                       0
DISTRICT                   212
WARD                        11
YEAR                         0
offensekey                   0
BID                     374442
sector                     243
PSA                        243
ucr-rank                     0
BLOCK_GROUP               1192
VOTING_PRECINCT             68
XBLOCK                       0
BLOCK                        1
START_DATE                  14
CCN                          0
OFFENSE                      0
OCTO_RECORD_ID               0
ANC                          0
REPORT_DAT                   0
METHOD                       0
location                     0
LATITUDE                     0
dtype: int64
```

Heat map for missing values. 1 means both features must be missing simultaneously. -1 indicates that one feature is missing if and only if the other feature is not.

[26]:
```
msno.heatmap(crime)
```

[26]:
```
<AxesSubplot:>
```



*NOTE: We find that the sum of missing values of CENSUS_TRACT, BLOCK_GROUP, sector and PSA are equal respectively.In the heat map, the correlation between CENSUS_TRACT and BLOCK_GROUP, sector and PAS are 1. While the correlation between PSA and DISTRICT, sector and DISTRICT are 0.9.

So we can conclude that the features BLOCK_GROUP and CENSUS_TRACT; PSA, SECTOR, and DISTRICT may be missing for the same reason respectively. Or they come from the same data source respectively.

Deleting features. Based on the visualization, we choose to delete feature BID.

[27]:
```
del crime['BID']
```

Based on the conclusion of Out[2], the former item is chosen to fill in here.

[28]:
```
crime['START_DATE']=crime['START_DATE'].fillna(method='pad')
crime['END_DATE']=crime['END_DATE'].fillna(method='pad')
```

Deleting samples. Since the proportion of samples containing missing values are small enough, deleting is reasonable.

[29]:
```
crime_filtered=crime.dropna()
```

[30]:
```
crime_filtered.isnull().sum()
# show
```

[30]:
```
NEIGHBORHOOD_CLUSTER    0
CENSUS_TRACT            0
offensegroup            0
LONGITUDE               0
END_DATE                0
offense-text            0
SHIFT                   0
YBLOCK                  0
DISTRICT                0
WARD                    0
YEAR                    0
offensekey              0
sector                  0
PSA                     0
ucr-rank                0
BLOCK_GROUP             0
VOTING_PRECINCT         0
XBLOCK                  0
BLOCK                   0
START_DATE              0
CCN                     0
OFFENSE                 0
OCTO_RECORD_ID          0
ANC                     0
REPORT_DAT              0
METHOD                  0
location                0
LATITUDE                0
dtype: int64
```

2-1-3 Removing Redundant Data

[31]:
```
crime_filtered.drop_duplicates(keep='first',inplace=True)
```

2-1-4 Outlier Detection And Handling According to the feature description, we can do some outlier detection.

141 crimes started before 2007. 1 crime started in 1800, 2 crimes started in 1912 and 1 crime started in 1914. We have two explanations. One is that these are data in error, i.e., outliers. Another explanation is that these unsolved crimes recently solved through new technology advancements.

[32]:
```
print(crime_filtered['START_DATE'][crime_filtered['START_DATE']<'2007-01-01'].
 ↪count())
print(sorted(crime_filtered['START_DATE'][crime_filtered['START_DATE']<'2007-01-01'])[:
 ↪10])
```

```
141
['1800-01-16T09:51:02.000', '1912-08-22T21:00:00.000',
'1912-09-16T23:40:00.000', '1914-09-26T19:00:00.000', '1915-03-18T16:00:00.000',
```

```
'1915-08-30T06:00:35.000', '1915-09-17T18:00:52.000', '1915-10-10T22:30:45.000',
'1915-10-16T21:03:03.000', '1915-10-17T21:00:57.000']
```

Anyway, remove them for later analysis.

[33]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'START_DATE']<'2007-01-01':
        crime_filtered.drop(i,inplace=True)
```

55 crimes ended before 2007. 2 crimes ended in 1916, 1 crime ended in 1974 and 1 crime ended in 1980.
We view them as outliers and delete them.

[34]:
```
print(crime_filtered['END_DATE'][crime_filtered['END_DATE']<'2007-01-01'].
 ↪count())
print(sorted(crime_filtered['END_DATE'][crime_filtered['END_DATE']<'2007-01-01'])[:
 ↪10])
```

```
6
['1985-01-11T04:00:39.000', '2000-01-04T04:55:00.000',
'2000-08-28T20:10:00.000', '2003-07-23T06:00:00.000', '2005-04-17T15:30:31.000',
'2005-04-30T00:00:00.000']
```

[35]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'END_DATE']<'2007-01-01':
        crime_filtered.drop(i,inplace=True)
```

Three outliers, delete them.

[36]:
```
print(sorted(crime_filtered['END_DATE'][crime_filtered['END_DATE']>'2022-01-01'])[:
 ↪10])
for i in crime_filtered.index:
    if crime_filtered.loc[i,'END_DATE']>'2022-01-01':
        crime_filtered.drop(i,inplace=True)
```

```
['2102-01-16T13:42:00.000', '2110-01-02T09:40:00.000',
'2911-12-06T12:00:00.000']
```

Similarly, we detect difference between OCTO_RECORD_ID and CCN.

[37]:
```
crime_filtered['OCTO_RECORD_ID'].head(5)
```

[37]:
```
0    17070672-01
1    17070675-01
2    17070714-01
3    17070736-01
4    17070780-01
Name: OCTO_RECORD_ID, dtype: object
```

[38]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'CCN']!=int(crime_filtered.
 ↪loc[i,'OCTO_RECORD_ID'][0:8]):
        print(i)
# normal
```

Besides, we notice that most values of OCTO_RECORD_ID end with 1, so we infer that the sample would be a outlier if its OCTO_RECORD_ID does not end with 1.

[39]:
```
n=0
for i in crime_filtered.index:
    if int(crime_filtered.loc[i,'OCTO_RECORD_ID'].split('-')[1])!=1:
        n+=1
print(n)
```

```
132
```

Therefore, we can conclude that the values does not end with 1 are outliers. Then, we should change them to end with 1. But after that, all samples' OCTO_RECORD_ID have the form "'CCN'-01". Therefore, OCTO_RECORD_ID and CCN contain the same information, we can delete column OCTO_RECORD_ID here.

[40]:
```
del crime_filtered['OCTO_RECORD_ID']
```

Detection between location and LATITUDE, LONGITUDE.

[41]:
```
for i in crime_filtered.index:
     if abs(crime_filtered.loc[i,'LATITUDE']-float(crime_filtered.
 ↪loc[i,'location'].split(',')[0]))>0.001:
            print(i)
for i in crime_filtered.index:
     if abs(crime_filtered.loc[i,'LONGITUDE']-float(crime_filtered.
 ↪loc[i,'location'].split(',')[1]))>0.001:
            print(i)
# normal
```

Same reason, we delete column location.

[42]:
```
del crime_filtered['location']
```

Detection between CENSUS_TRACT and BLOCK_GROUP.

[43]:
```
for i in crime_filtered.index:
     if crime_filtered.loc[i,'CENSUS_TRACT']!=int(crime_filtered.
 ↪loc[i,'BLOCK_GROUP'].split(' ')[0]):
            print(i)
# normal
```

Same reason, we delete column BLOCK_GROUP.

[44]:
```
del crime_filtered['BLOCK_GROUP']
```

Detection between offense-text and OFFENSE.

[45]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'offense-text']!=crime_filtered.loc[i,'OFFENSE']:
        print(i)
# normal
```

Same reason, we delete column offense-text.

[46]:
```
del crime_filtered['offense-text']
```

Detection between offensegroup, offensekey.

[47]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'offensekey'].split('|')[0]!=crime_filtered.
 ↪loc[i,'offensegroup']:
        print(i)
# normal
```

Detection between offensekey, OFFENSE.

[48]:
```
for i in crime_filtered.index:
    if crime_filtered.loc[i,'offensekey'].split('|')[1]!=crime_filtered.
 ↪loc[i,'OFFENSE']:
        print(i)
# normal
```

Same reason, we delete column offensekey.

[49]:
```
del crime_filtered['offensekey']
```

Next, let's visualize the geographic distribution to visually determine whether there are outliers.(Use DBSCAN for clustering)

[50]:
```
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
```

[51]:
```
crime_filtered.shape[0]
# number of rows
```

[51]:
```
442043
```

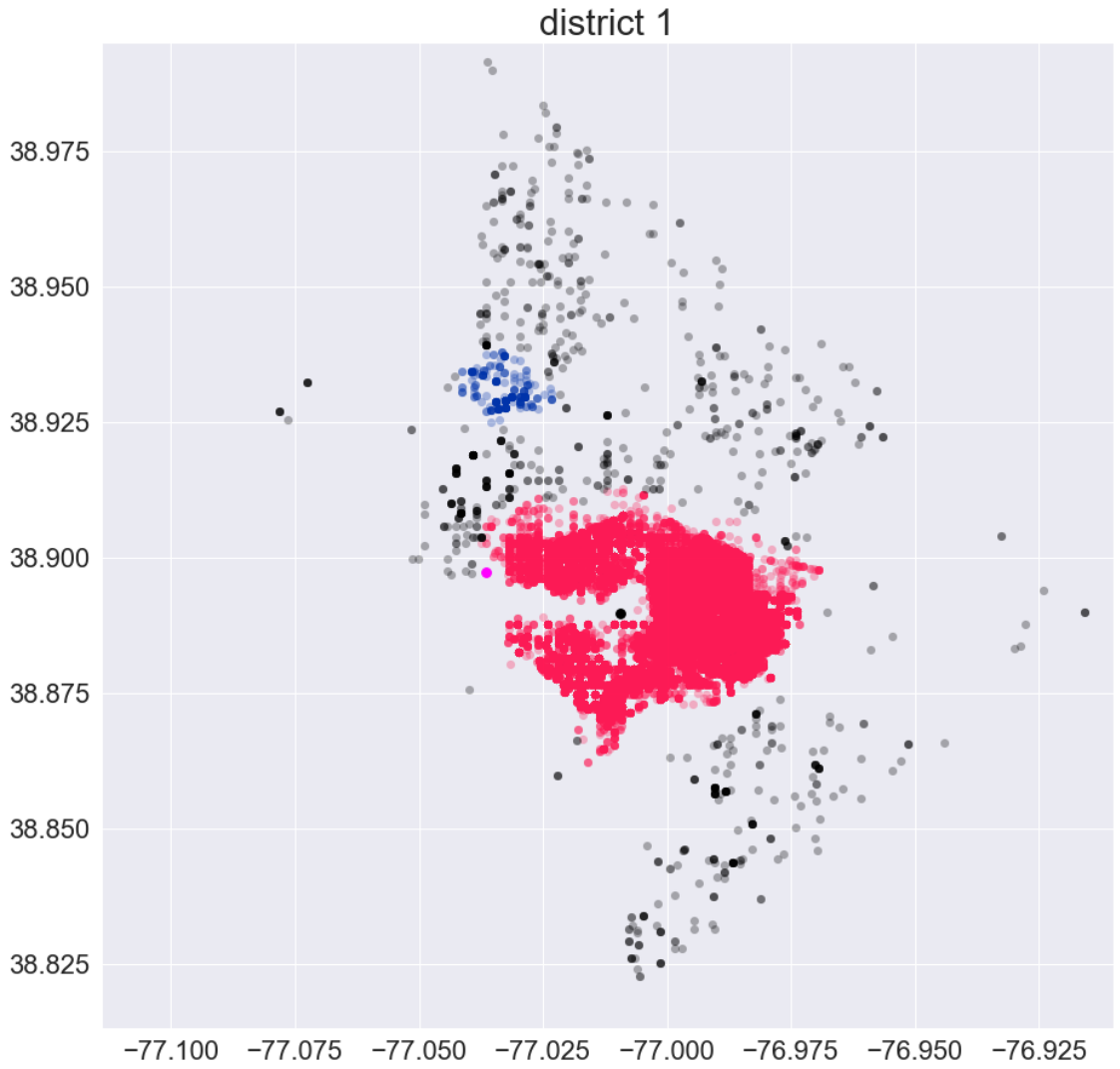
[52]:
```
# define a function to visualize clusters
# this function is also used for clustering in Task 3
def see_cluster(data,num,type,T):
    plt.figure(figsize=(15,15))
    plt.xlim(-77.114,-76.91)
    plt.ylim(38.813,38.995)
```

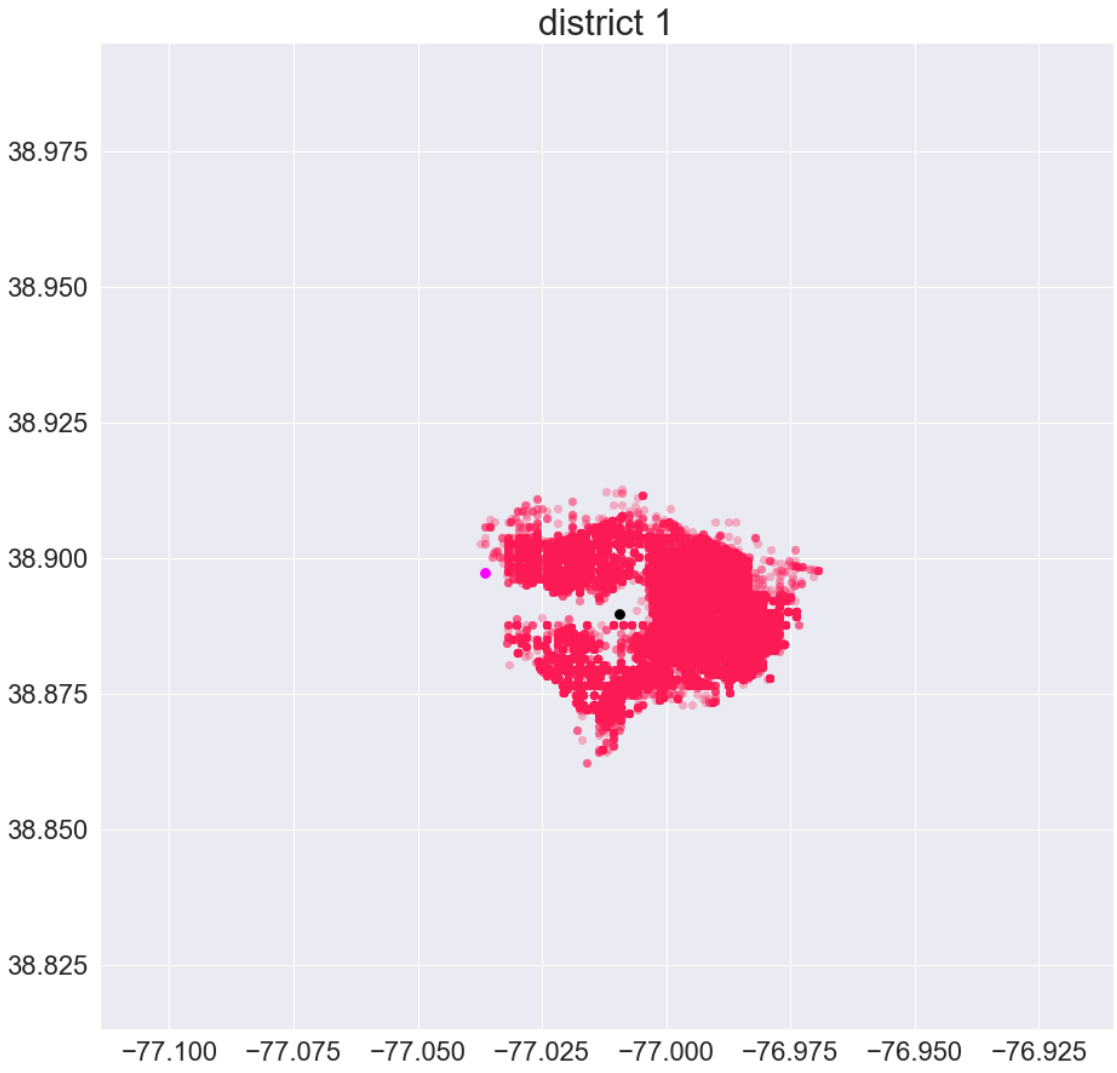
```
    sns.set_theme(style="darkgrid")
    sns.set(font_scale=2)
    for i in range(0,num+1):
        plt.scatter(data['LONGITUDE'][data['cluster']==i],␣
↪data['LATITUDE'][data['cluster']==i], s=50, alpha=0.3,␣
↪color=[(i+1)%2,((i+1)%10)/10,((i+1)%3)/3], lw=0, label = 'cluster '+ str(i))
    plt.scatter(data['LONGITUDE'][data['cluster']==-1],
    data['LATITUDE'][data['cluster']==-1], s=50, alpha=0.3,
    color=[0,0,0], lw=0, label = 'noise')
    plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White␣
↪House')
    plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S.␣
↪Capitol')
    if type == 0:
        plt.legend(loc='upper right')
    else:
        if T == 1:
            plt.legend(loc='upper right')
        plt.title(type,size=30)
    plt.show()
```

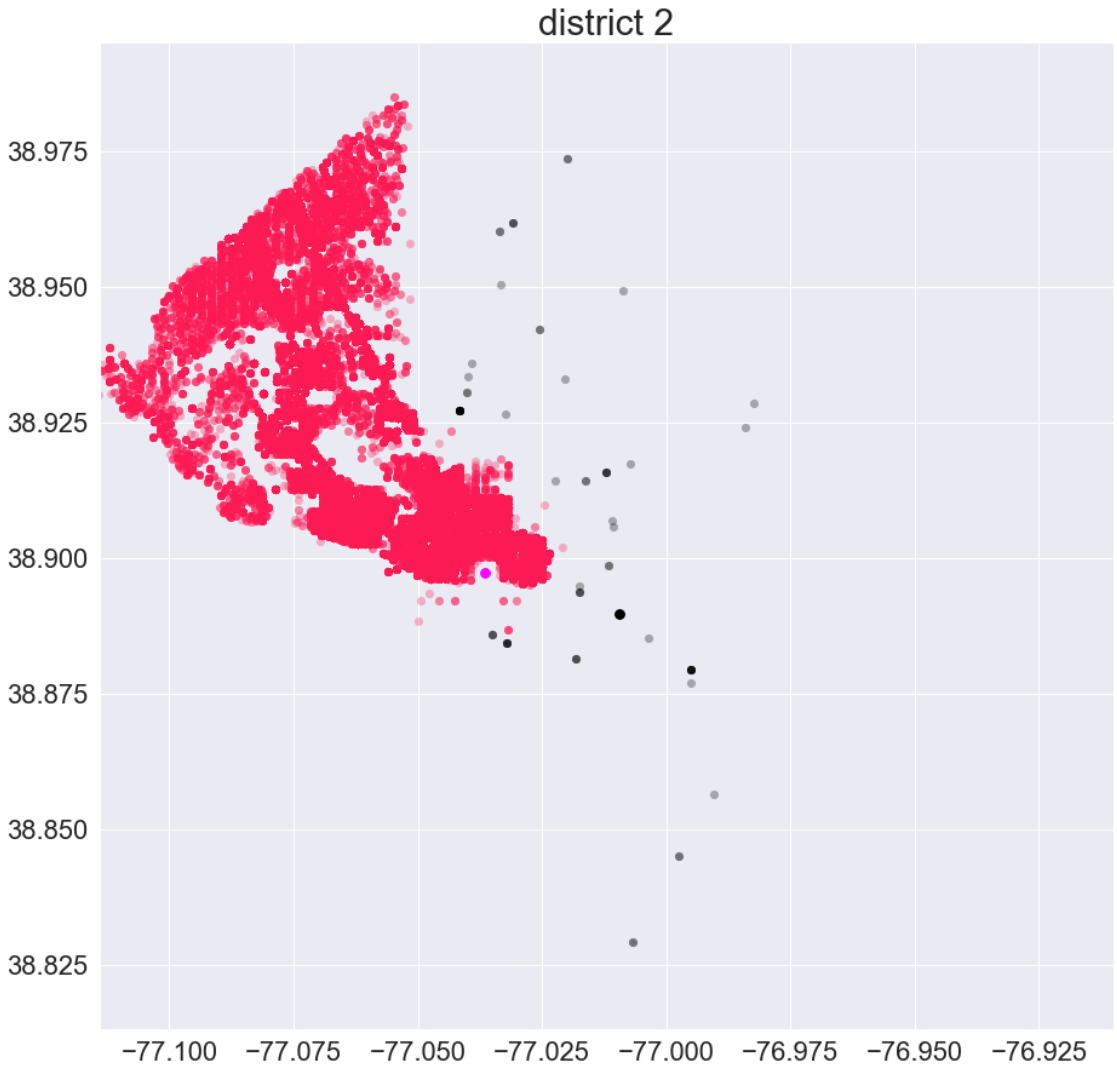
Here we used the extra data(District map: https://mpdc.dc.gov/page/police-districts-and-police-service-areas). We compare the following plots to the district map and find that there are a lot of outliers. So we delete these outliers.
For each district, the first plot is the map with outliers and the second plot is the map after deleting outliers.

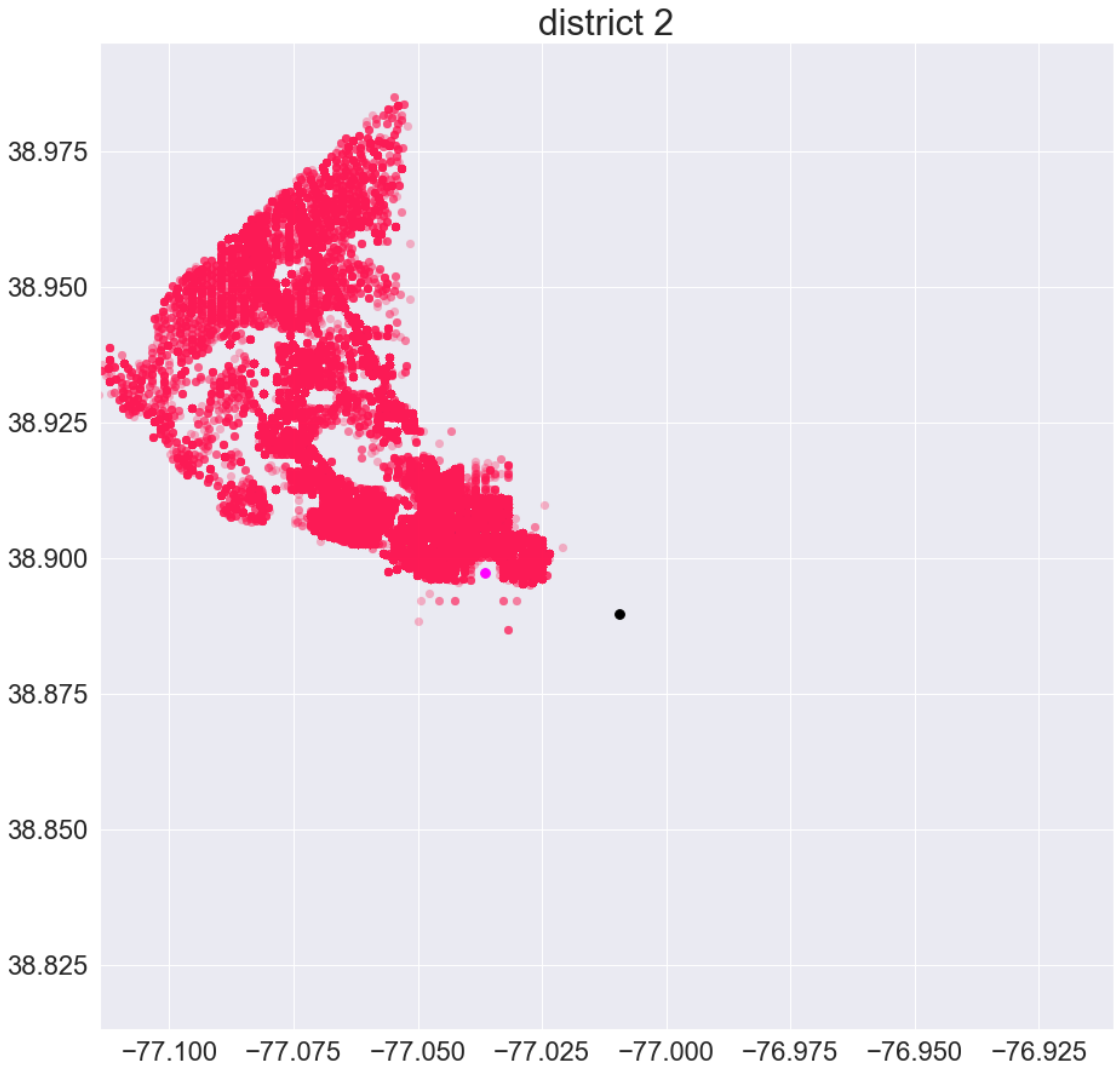
[56]:
```
tempdata=pd.DataFrame()
#
for i in range(1,8):
    data = crime_filtered[crime_filtered.DISTRICT==i]
    x = data[['LONGITUDE','LATITUDE']]
    x = preprocessing.scale(x)
    db = DBSCAN(eps=0.3, min_samples=50).fit(x)  # DBSCAN
    data['cluster']=db.labels_
    max_index = data['cluster'].max()
    see_cluster(data,max_index,'district '+str(i),0) # visualization

    # delete outliers
    temp = data[data['cluster']==-1].shape[0]
    index = -1
    for j in range(0,max_index+1):
        if temp < data[data['cluster']==j].shape[0]:
            temp = data[data['cluster']==j].shape[0]
            index = j
    newdata = data[data['cluster'] == index]
    see_cluster(newdata,1,'district '+str(i),0) #visualization
```

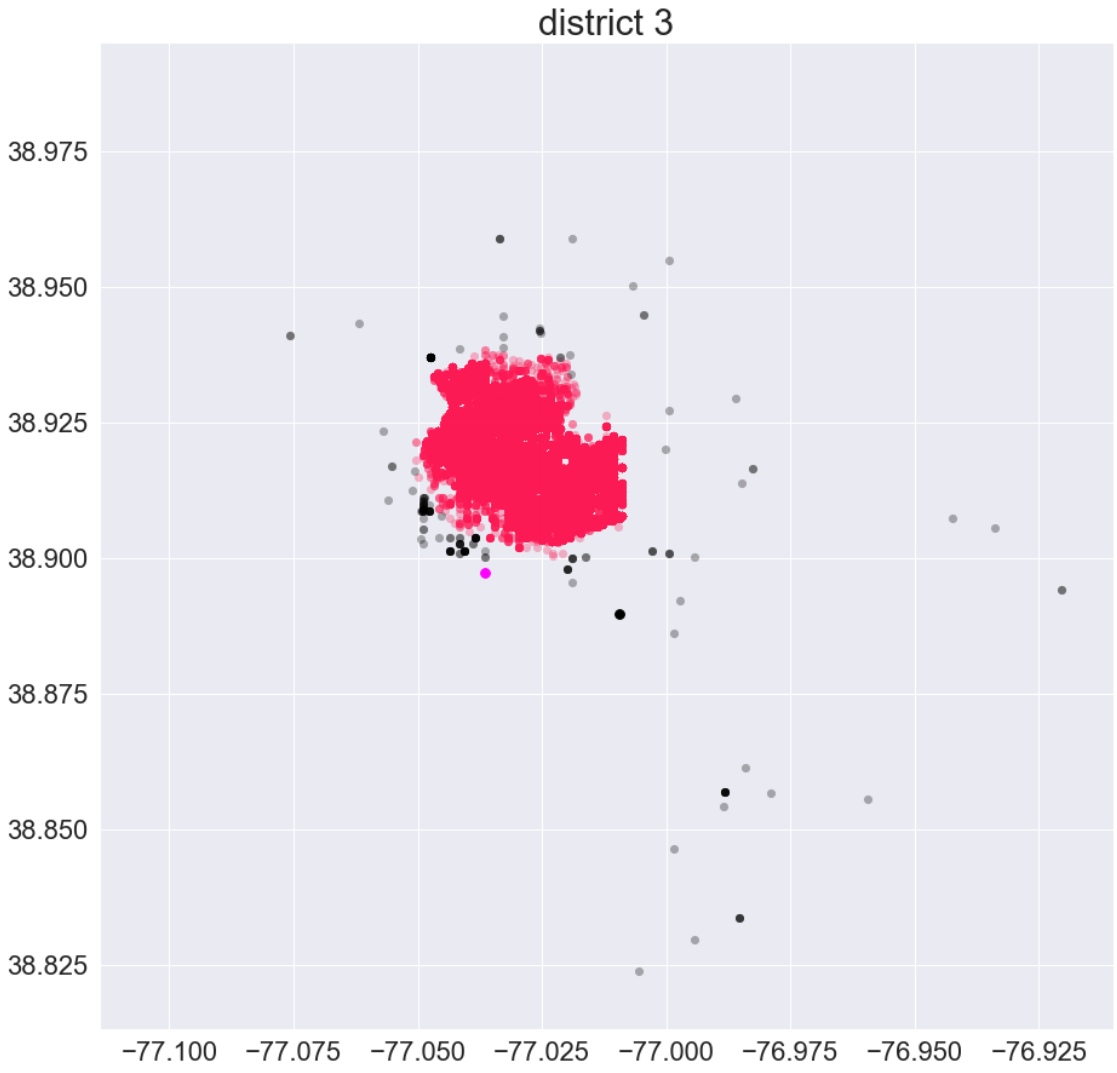
```
    # update data
    tempdata = tempdata.append(newdata)

crime_filtered = tempdata

# black and blue points are outliers
```

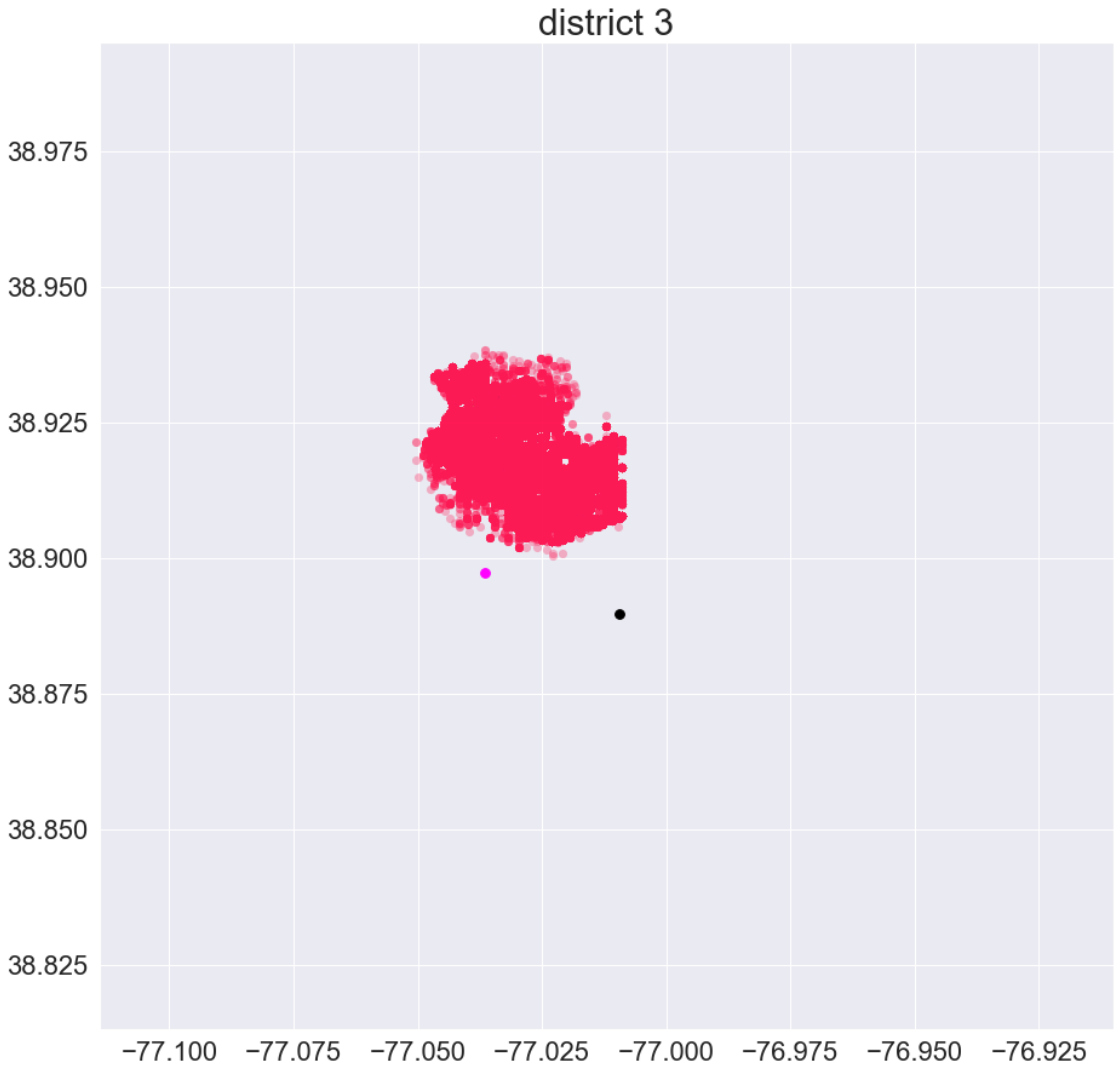


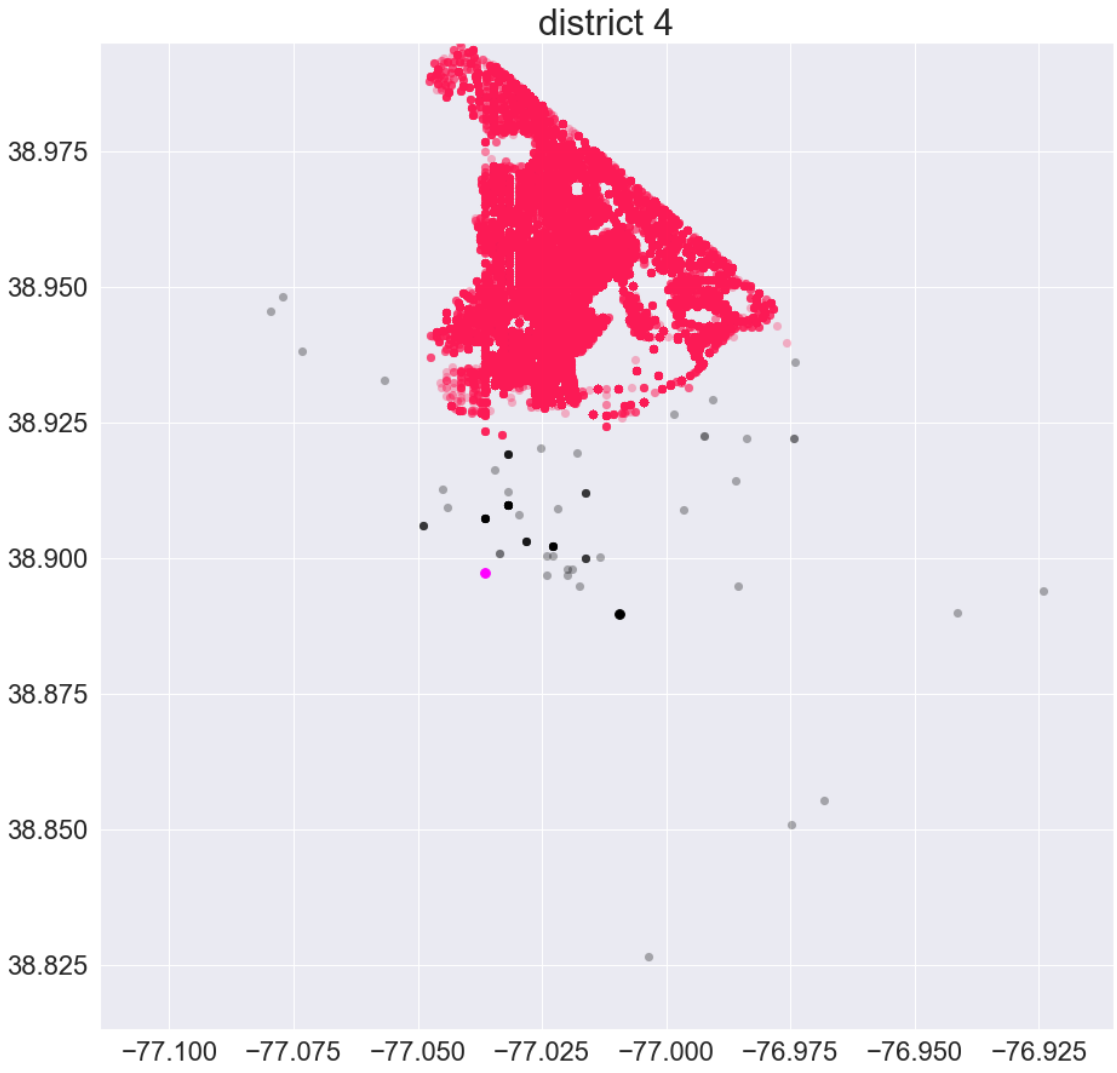

district 1
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

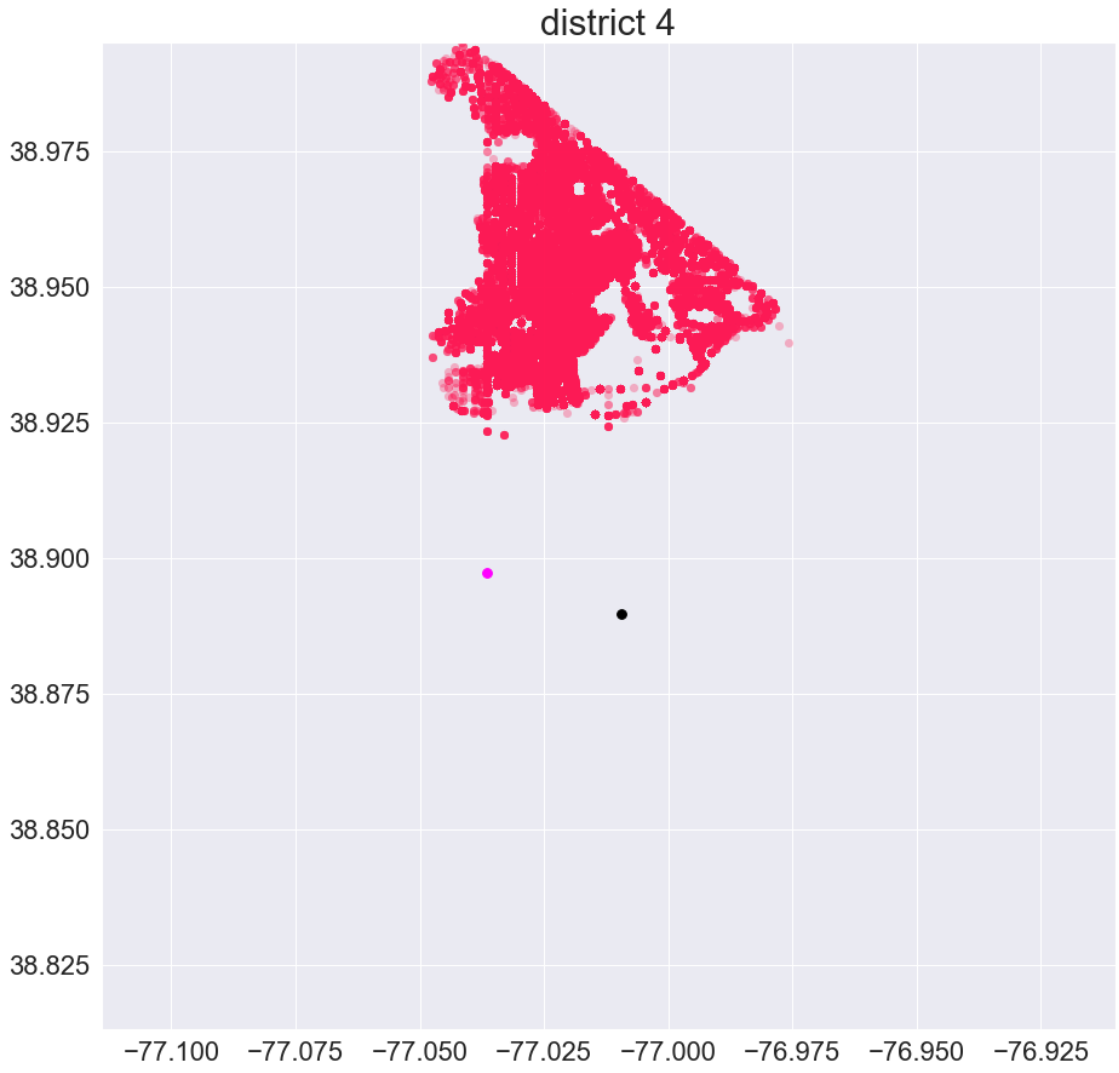

district 2
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

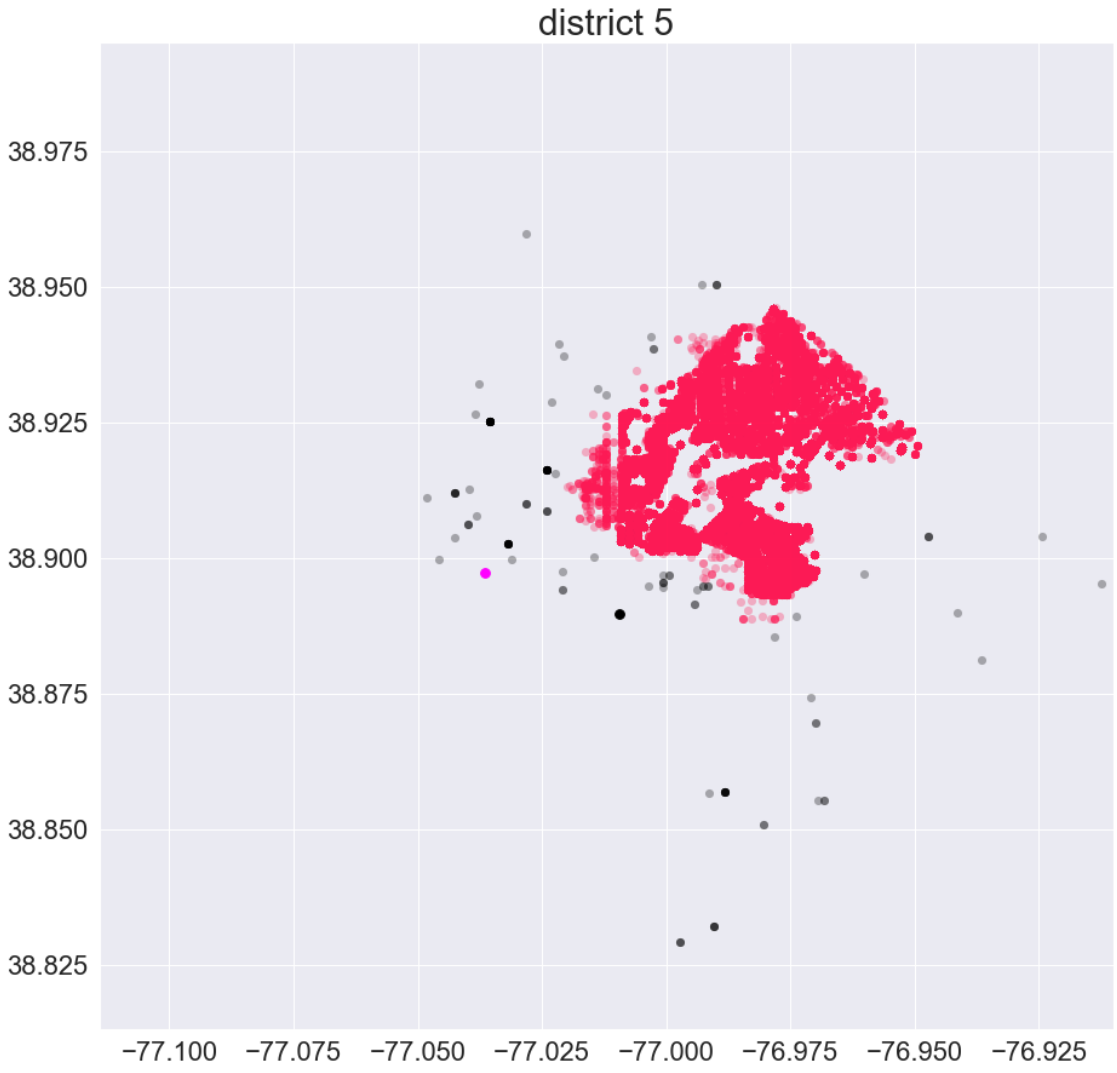

district 2
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

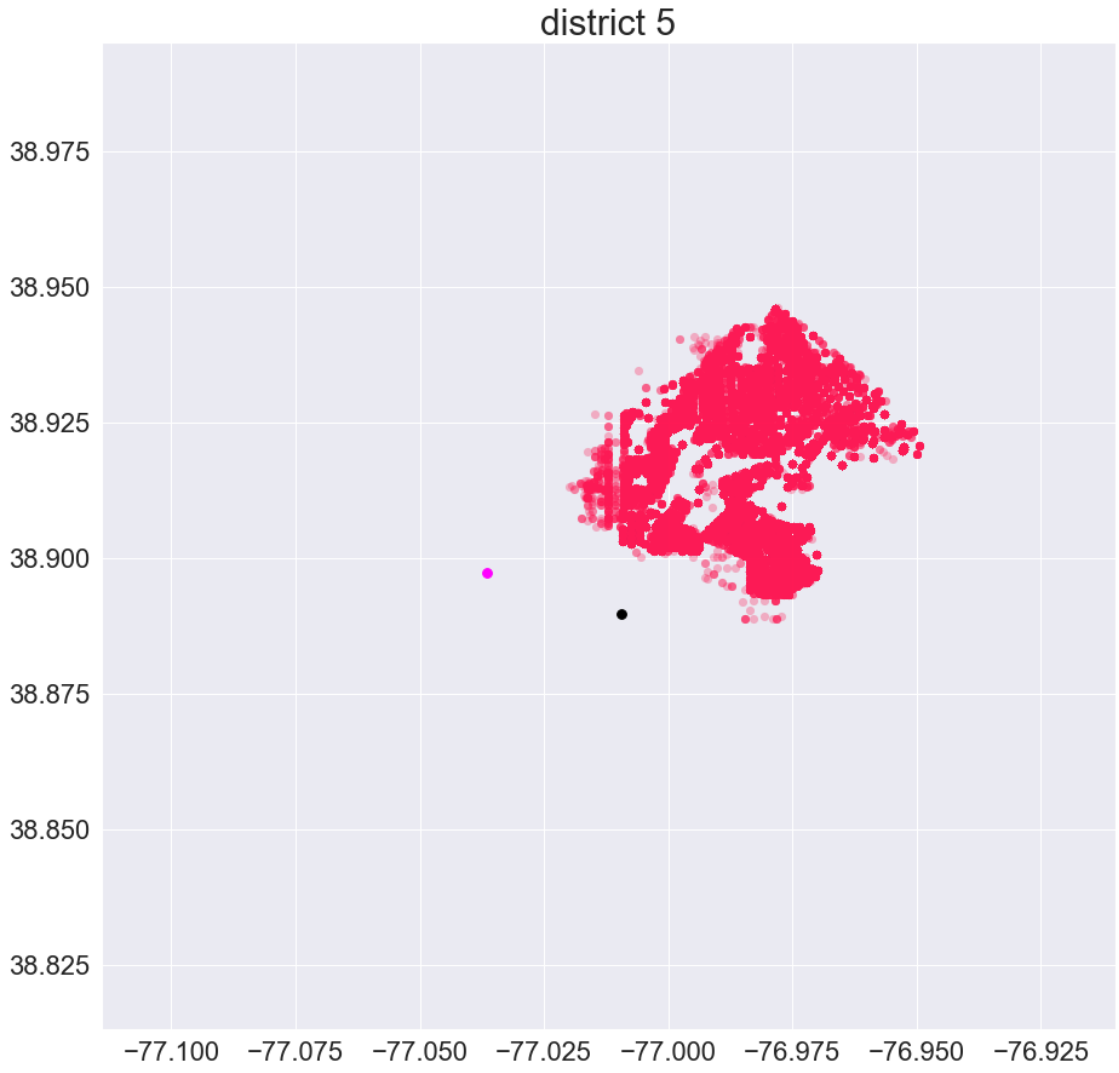

district 3
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

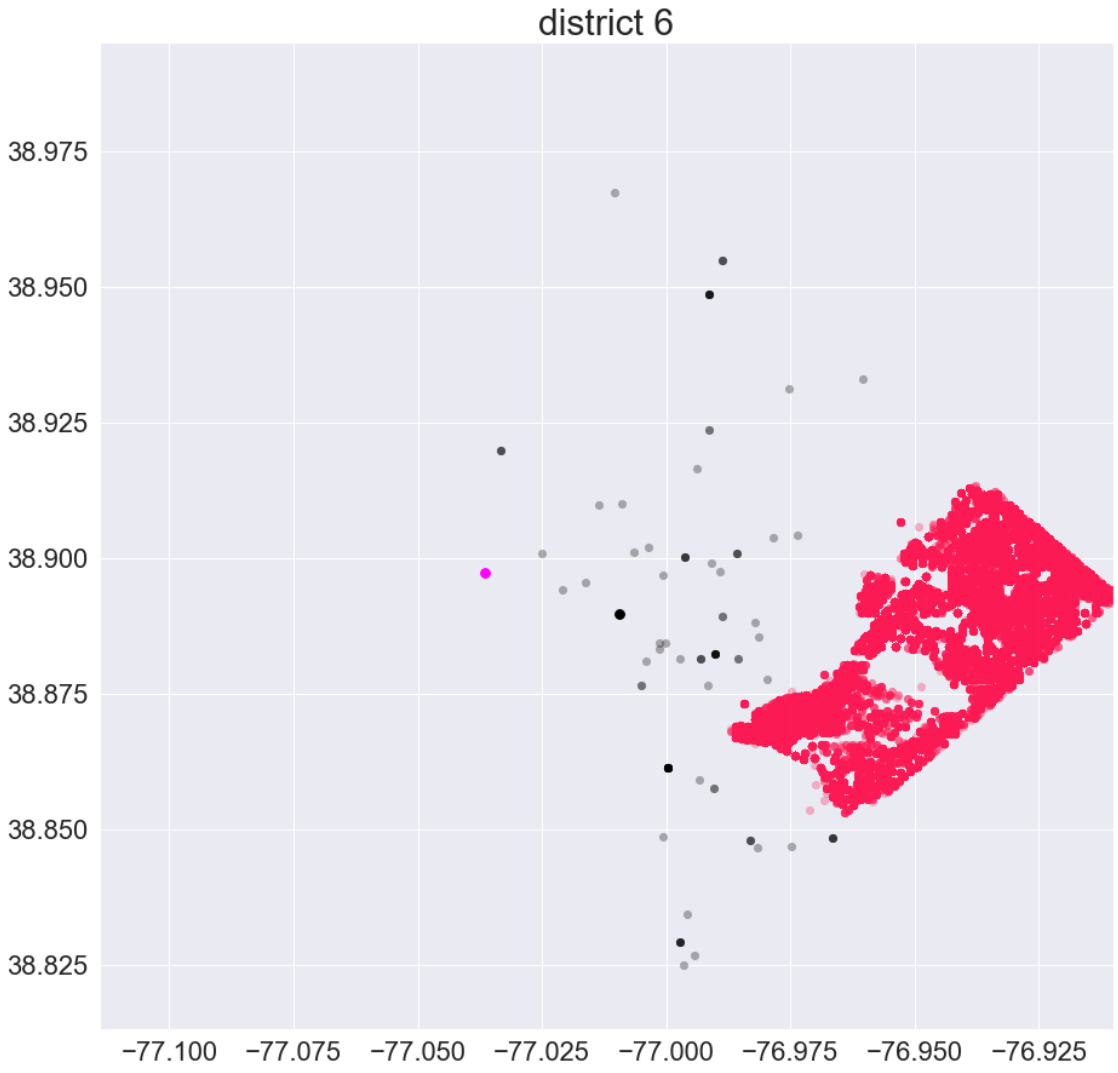

district 3
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

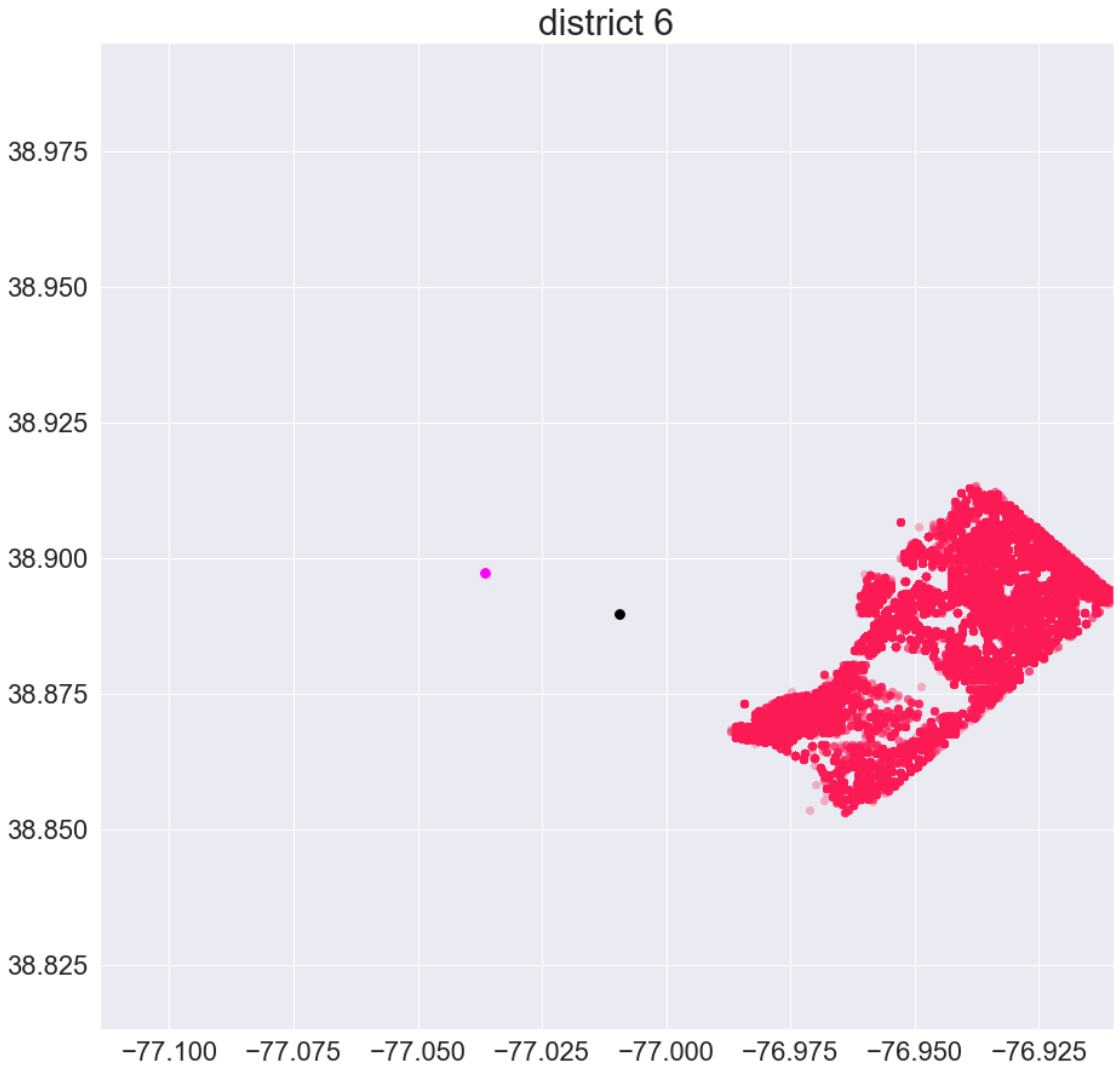

district 4
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

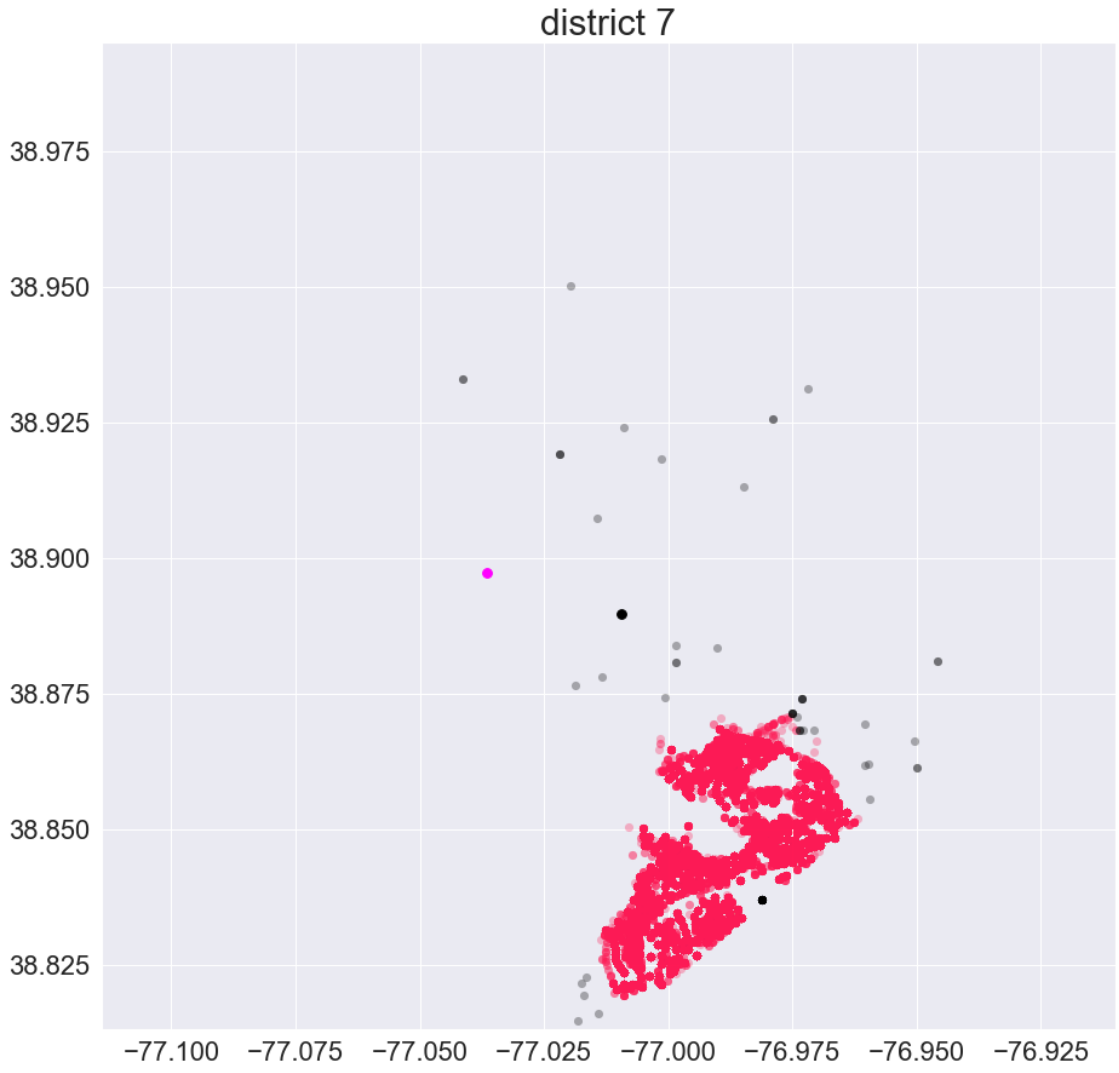

district 4
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

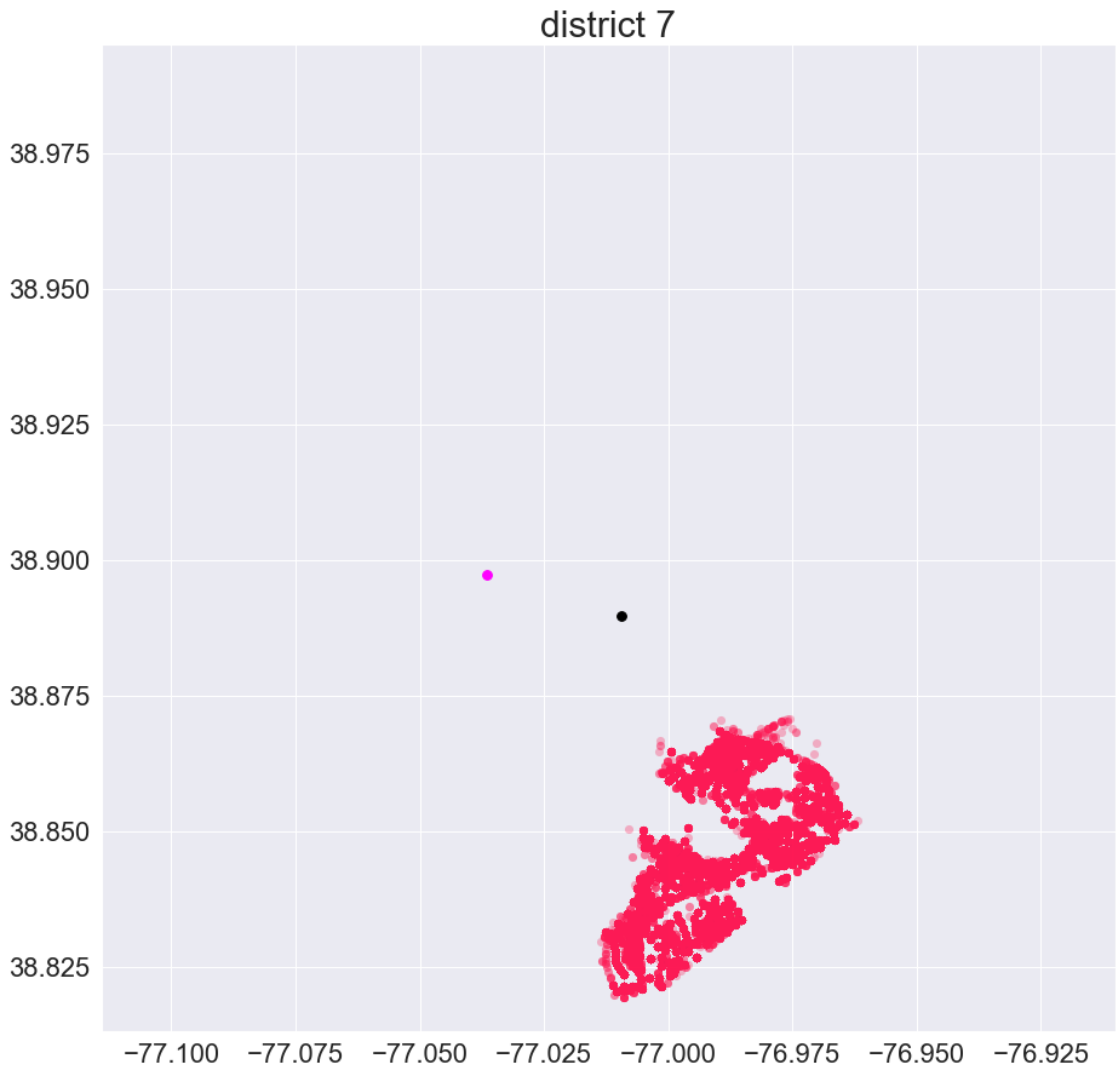

district 5
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


district 5
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


district 6
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


district 6
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


district 7
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925



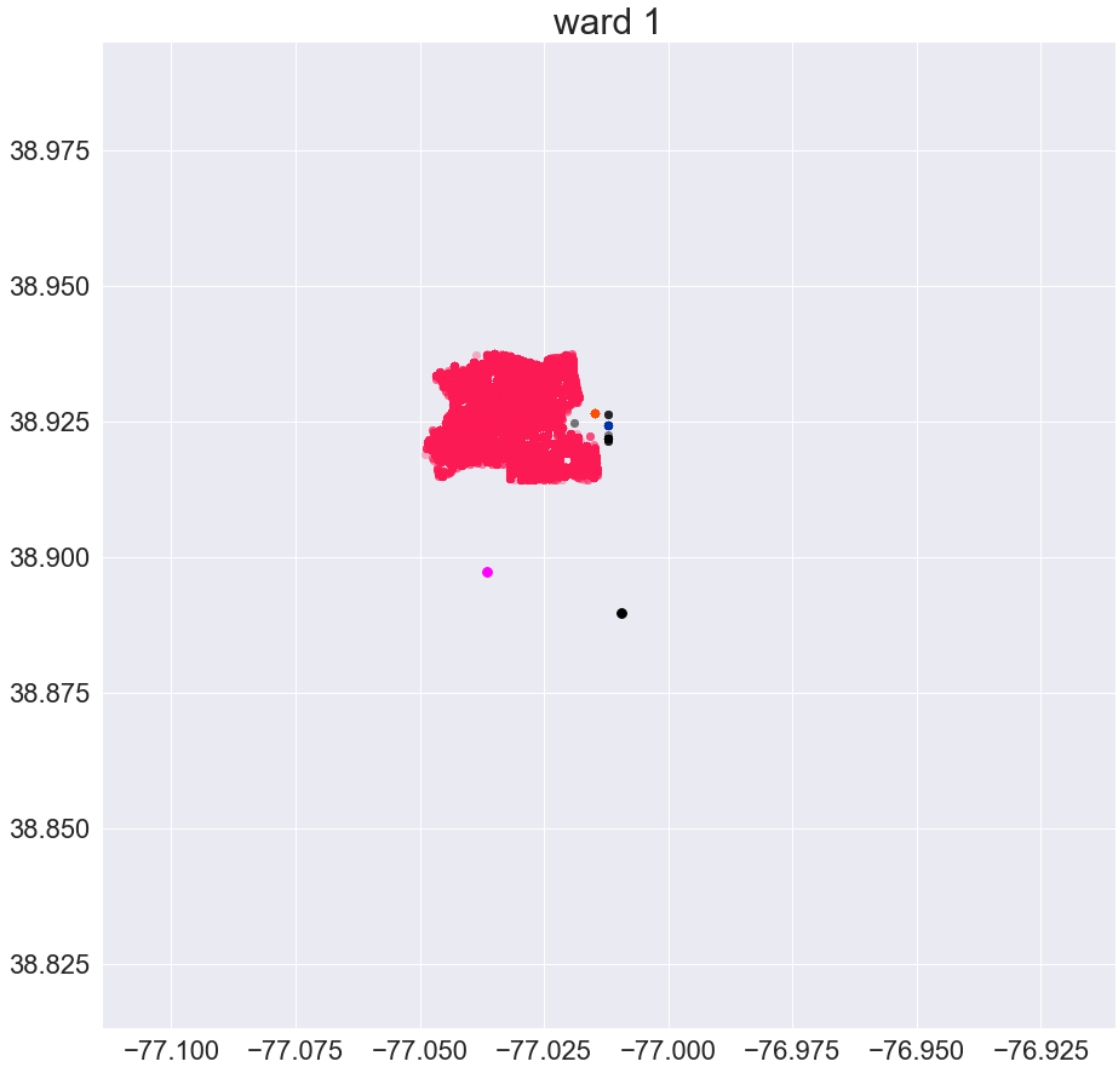
[57]:
```
crime_filtered.shape[0]
# number of rows
```

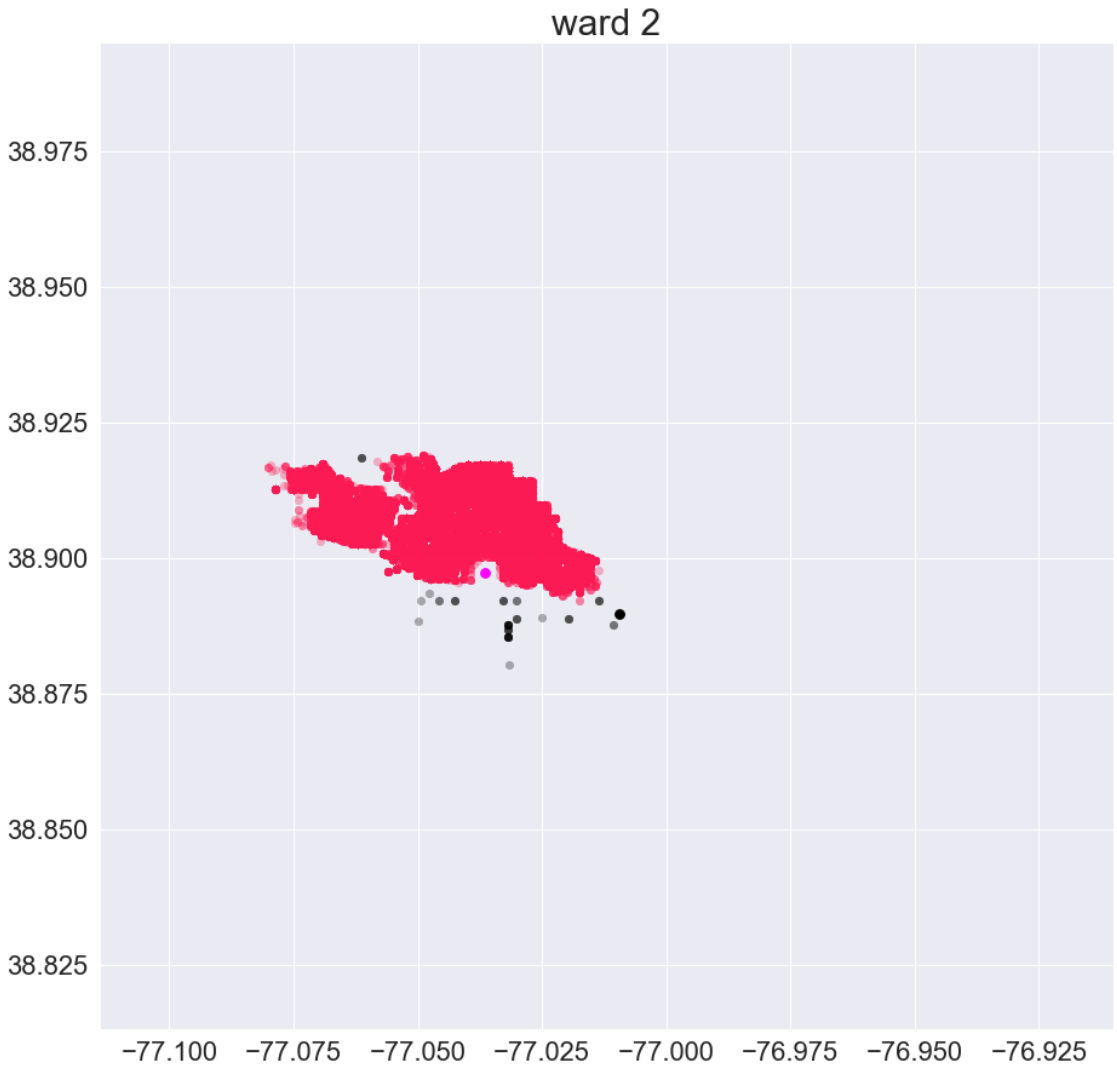
[57]: 440236

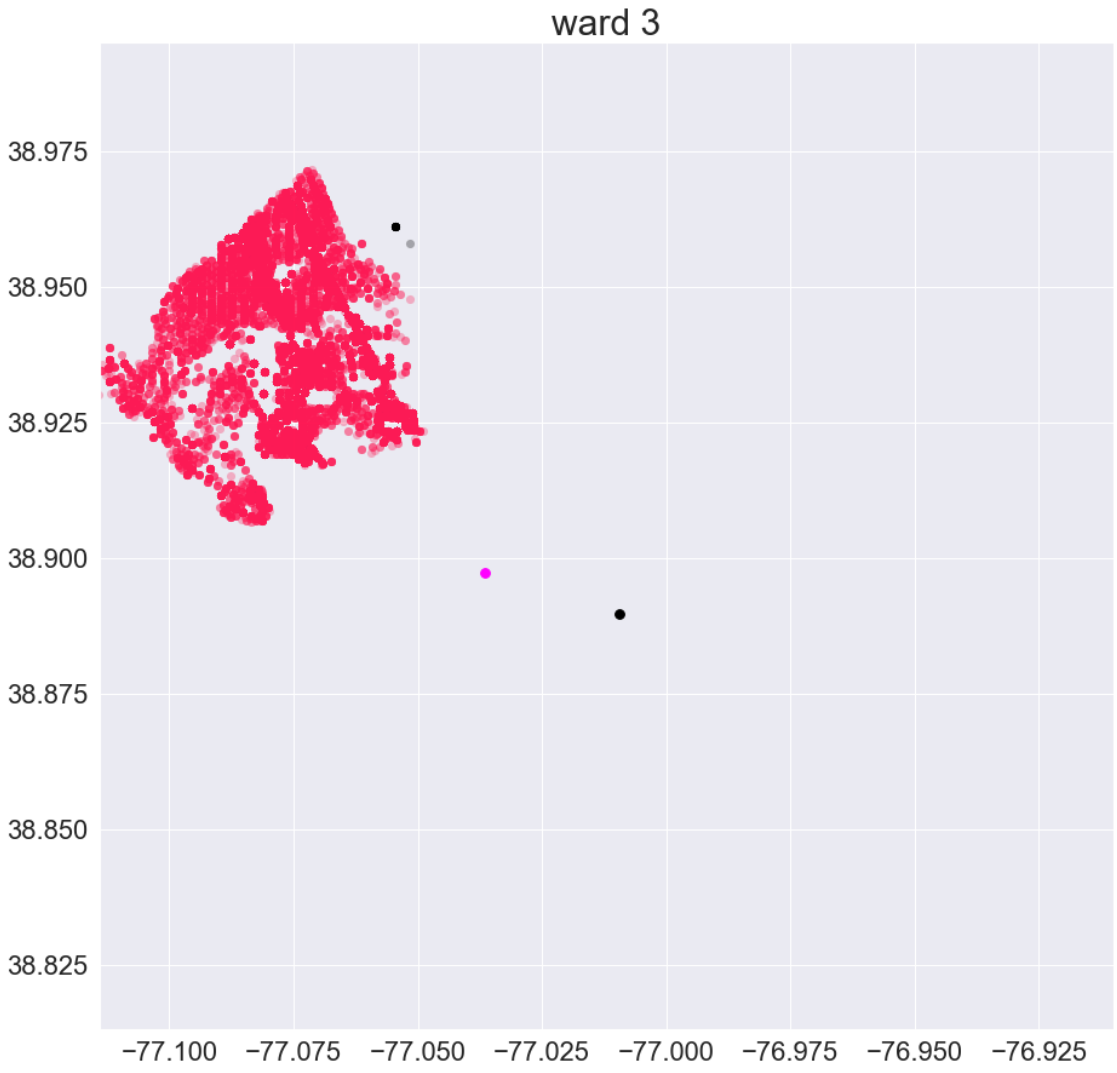
Similarly, we then consider 'WARD'. Fortunately, there are no outliers compared to the extra data(Ward map: https://planning.dc.gov/whatsmyward).

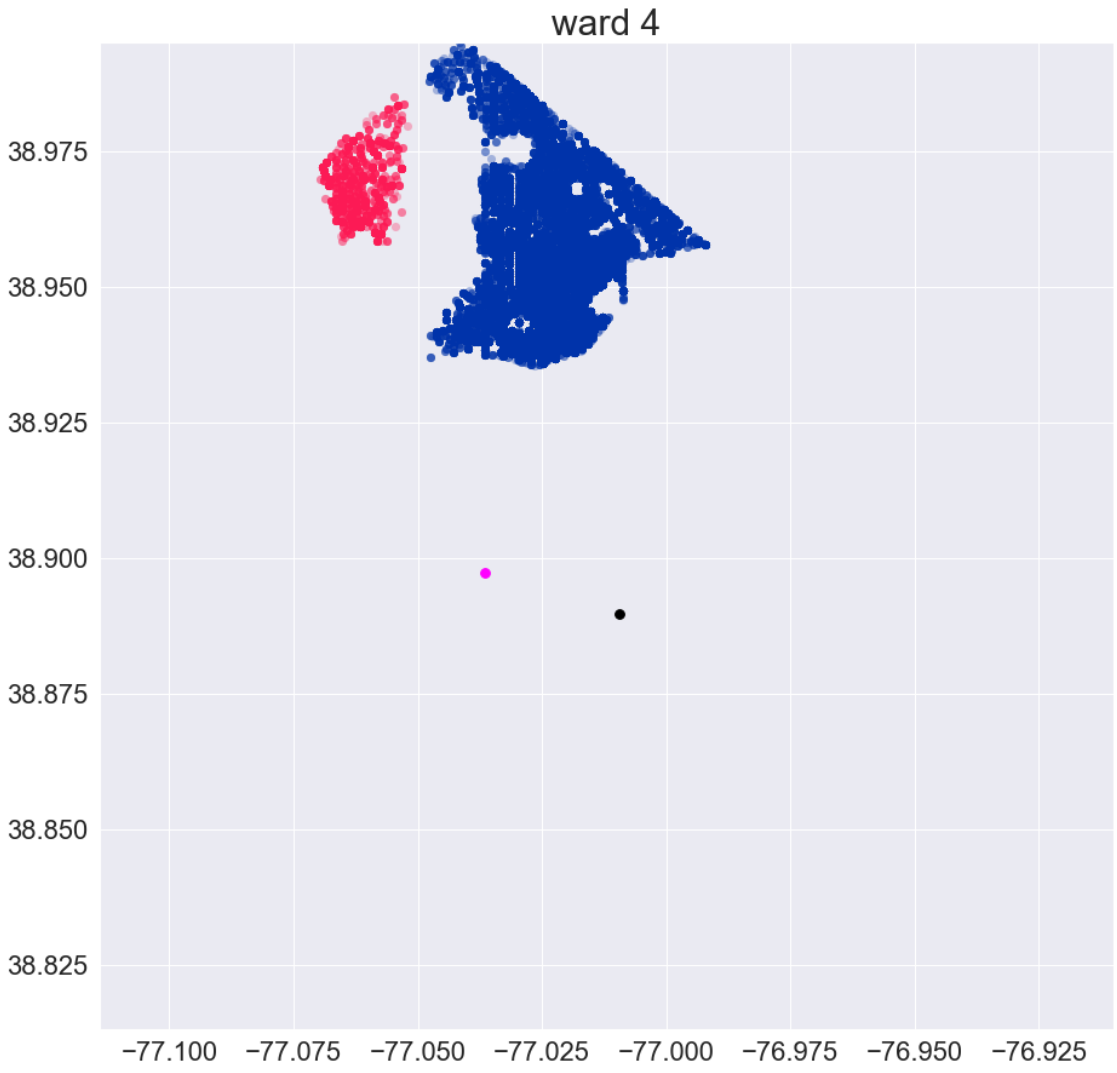
[58]:
```
for i in range(1,9):
    data = crime_filtered[crime_filtered.WARD==i]
    x = data[['LONGITUDE','LATITUDE']]
    x = preprocessing.scale(x)
    db = DBSCAN(eps=0.3, min_samples=50).fit(x)
    data['cluster']=db.labels_
    max_index = data['cluster'].max()
```

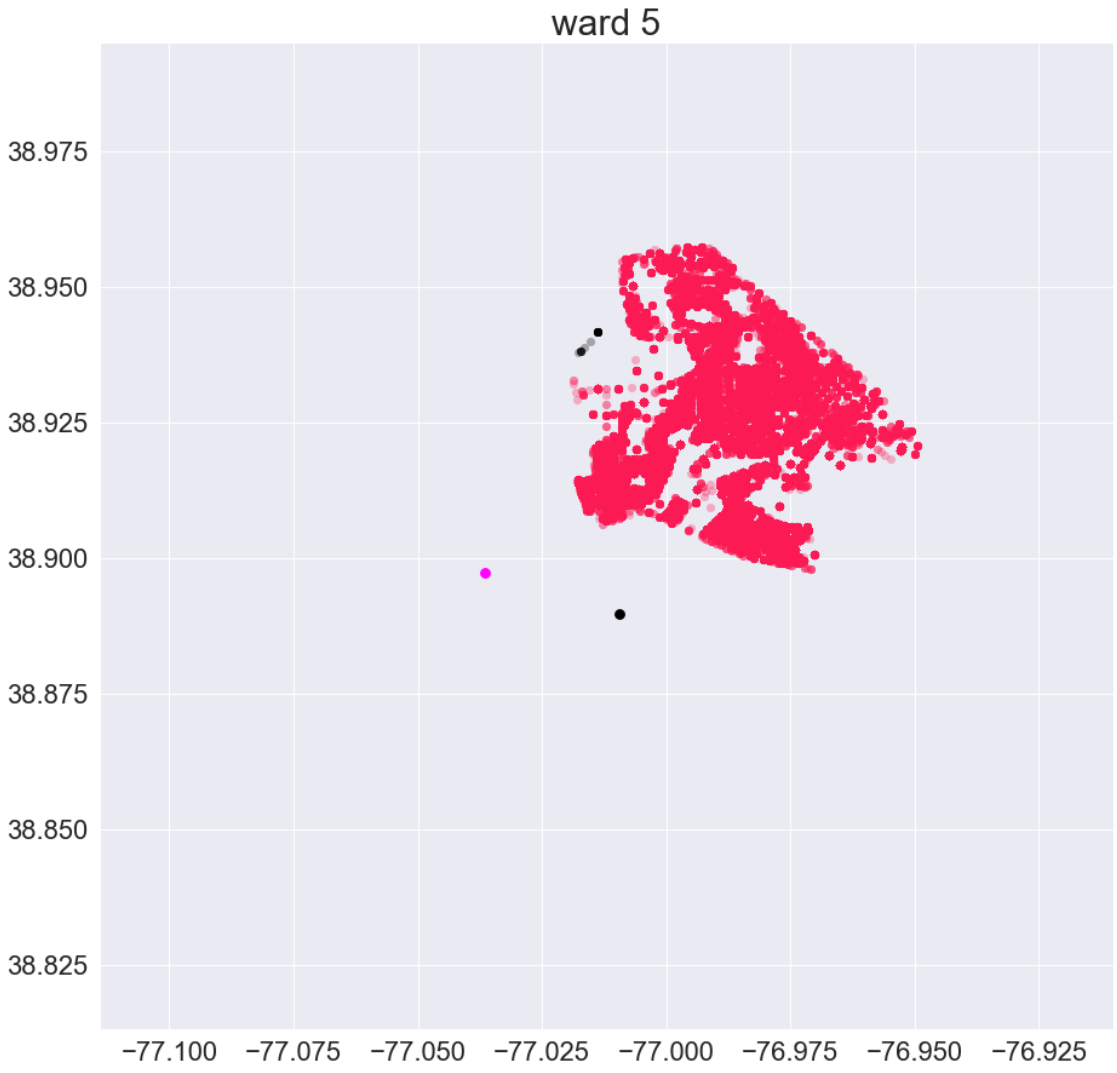
```
    see_cluster(data,max_index,'ward '+str(i),0)
# no outliers
```

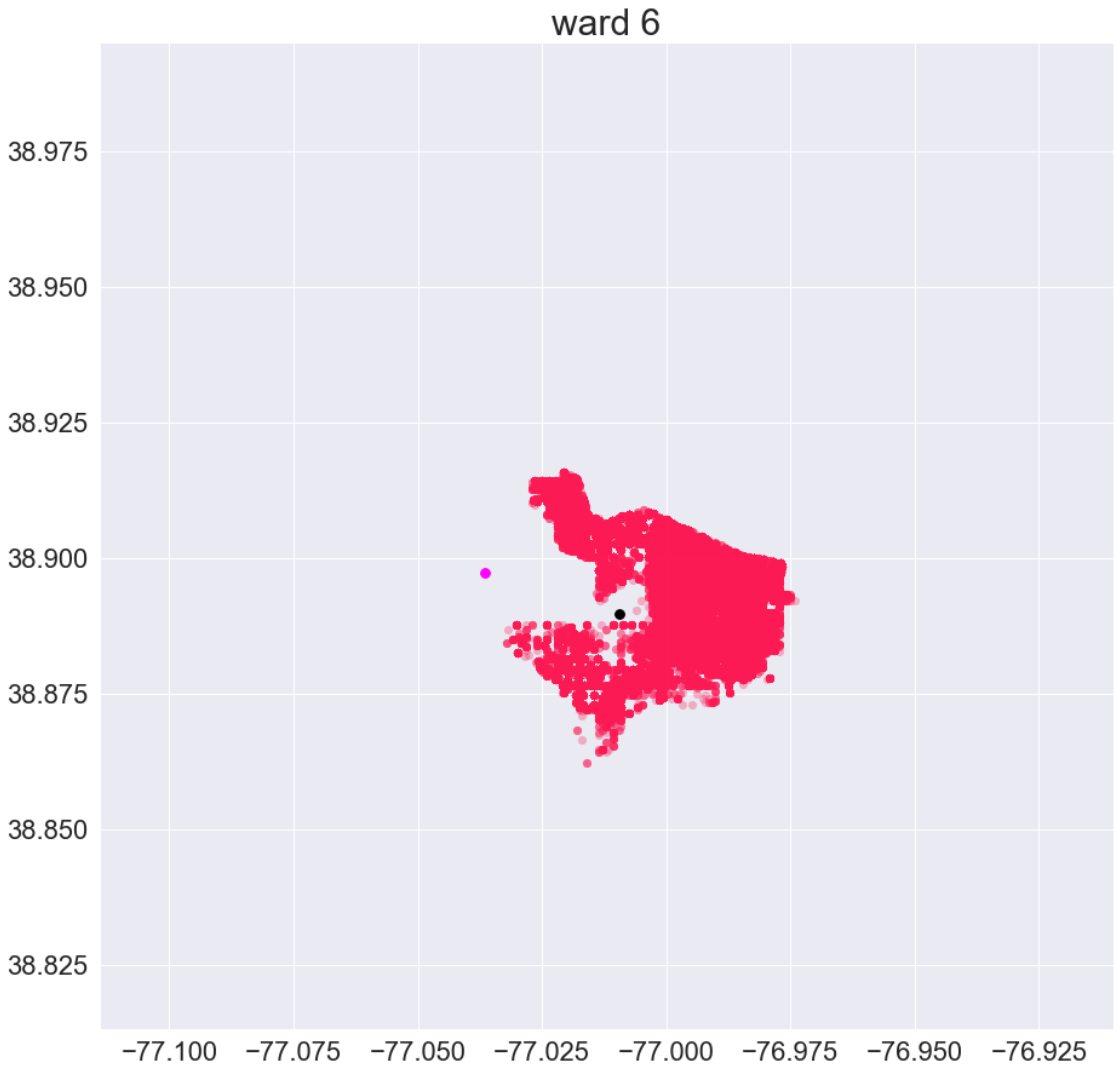


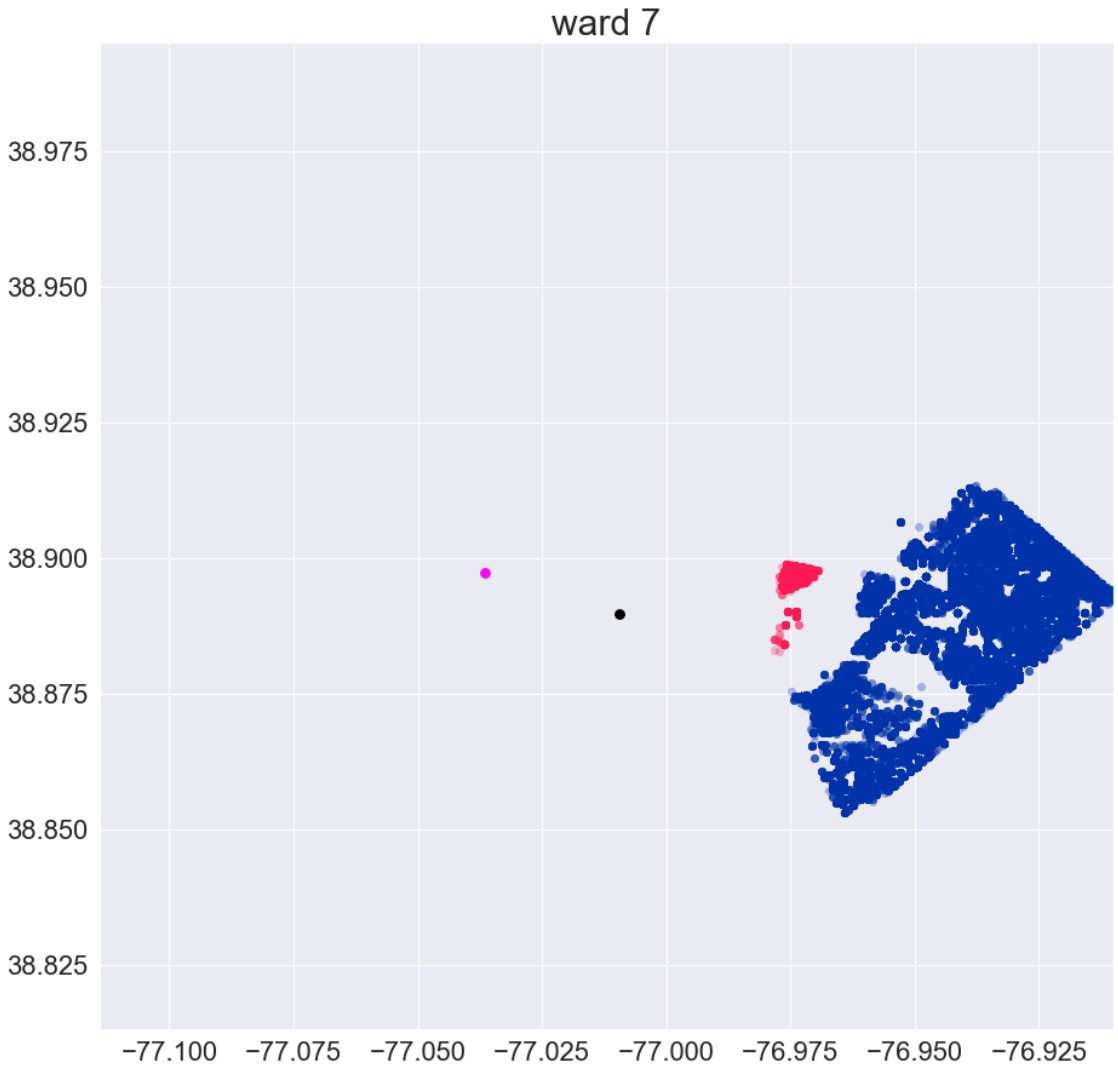

ward 2
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

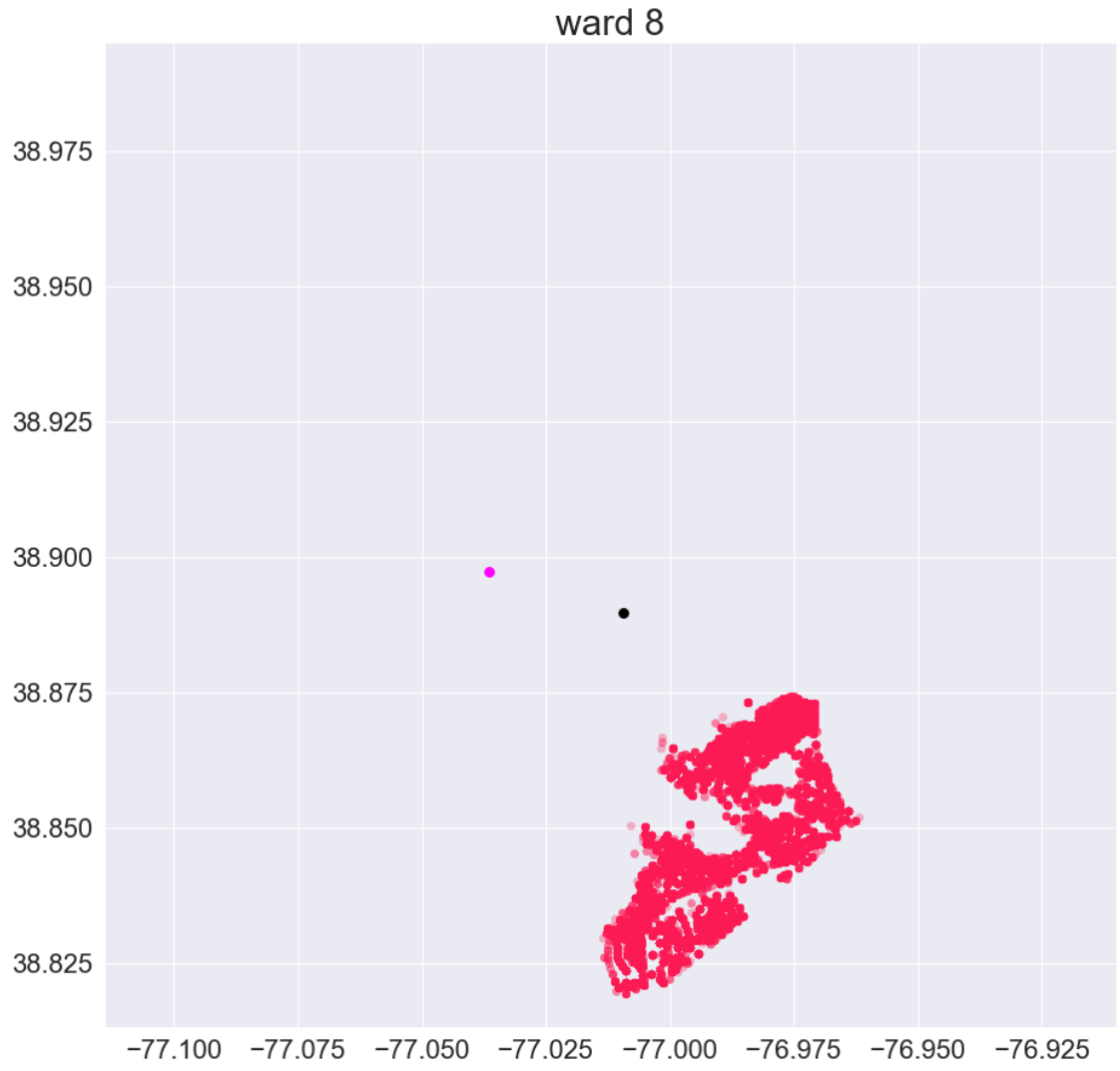

ward 3
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


ward 4
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


ward 5
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


ward 6
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


ward 7
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925



[59]:
```
# delete feature: cluster
del crime_filtered['cluster']
```

2-1-5 Character Encoding

[60]:
```
crime_temp = crime_filtered.copy()
```

[61]:
```
# NEIGHBORHOOD_CLUSTER:  e.g. CLUSTER 11---> 11
crime_filtered['NEIGHBORHOOD_CLUSTER'] = crime_filtered['NEIGHBORHOOD_CLUSTER'].
 ↪apply(str).map(lambda x: x.lstrip('cluster '))

# VOTING_PRECINCT:  e.g. presinct 12---> 12
crime_filtered['VOTING_PRECINCT'] = crime_filtered['VOTING_PRECINCT'].
 ↪apply(str).map(lambda x: x.lstrip('precinct '))
```

```
crime_filtered['REPORT_DAT'] = crime_filtered['REPORT_DAT'].apply(str).
 ↪map(lambda x: x.rstrip('Z'))
```

[62]:
```
# REPORT_DAT--->datetime
crime_filtered['REPORT_DAT'] = pd.to_datetime(crime_filtered['REPORT_DAT'])

# OFFENSE:
offense_mapping = {'theft/other':1, 'theft f/auto':2, 'burglary':3, 'assault w/
 ↪dangerous weapon':4, 'robbery':5,
                   'motor vehicle theft':6, 'homicide':7, 'sex abuse':8, 'arson':
 ↪9}
crime_filtered['OFFENSE_code'] = crime_filtered['OFFENSE'].str.lower().
 ↪map(offense_mapping).astype('category')
crime_filtered['OFFENSE'] = crime_filtered['OFFENSE'].str.replace('DANGEROUS␣
 ↪WEAPON', 'DW')

# DISTRICT
crime_filtered['DISTRICT'] = crime_filtered['DISTRICT'].astype(np.int64)

# PSA
crime_filtered['PSA'] = crime_filtered['PSA'].astype(np.int64)

# WARD
crime_filtered['WARD'] = crime_filtered['WARD'].astype(np.int64)

# ANC: A-->1; B-->2; C-->3; D-->4; E-->5; F-->6; G-->7
anc_mapping = {'1B':12, '1D':14, '1A':11, '1C':13, '6E':65, '4C':43, '5E':55,␣
 ↪'2B':22, '2D':24, '2F':26, '2C':23,
       '2E':25, '2A':21, '3C':33, '3E':35, '3B':32, '3D':34, '3F':36, '3G':37,␣
 ↪'4A':41, '4B':42, '4D':44,
       '5A':51, '5D':54, '5C':53, '5B':52, '6A':61, '6C':63, '6B':62, '6D':64,␣
 ↪'7D':74, '7C':73, '7E':75,
       '7B':72, '7F':76, '8A':81, '8B':82, '8C':83, '8D':84, '8E':85}
crime_filtered['ANC'] = crime_filtered['ANC'].map(anc_mapping).
 ↪astype('category')

# sector: e.g. 2D3--->23
sector_mapping = {'2D3':23, '3D2':32, '3D3':33, '1D2':12, '3D1':31, '5D2':52,␣
 ↪'2D2':22, '6D1':61, '4D1':41, '1D1':11, '6D3':63, '4D3':43,
                  '5D3':53, '4D2':42, '6D2':62, '5D1':51, '2D1':21, '1D3':13,␣
 ↪'7D2':72, '7D1':71, '7D3':73}
crime_filtered['sector'] = crime_filtered['sector'].map(sector_mapping).
 ↪astype('category')

# NEIGHBORHOOD_CLUSTER
```

```
crime_filtered['NEIGHBORHOOD_CLUSTER'] = crime_filtered['NEIGHBORHOOD_CLUSTER'].
 ↪astype(np.int64)

# CENSUS_TRACT
crime_filtered['CENSUS_TRACT'] = crime_filtered['CENSUS_TRACT'].astype(np.int64)

# VOTING_PRECINCT
crime_filtered['VOTING_PRECINCT'] = crime_filtered['VOTING_PRECINCT'].astype(np.
 ↪int64)

# CCN
crime_filtered['CCN'] = crime_filtered['CCN'].astype(np.int64)

# convert XBLOCK, YBLOCK to numeric
crime_filtered['XBLOCK'] = crime_filtered['XBLOCK'].astype(np.float64)
crime_filtered['YBLOCK'] = crime_filtered['YBLOCK'].astype(np.float64)

# START_DATE, END_DATE--->dateime
crime_filtered['START_DATE'] = pd.to_datetime(crime_filtered['START_DATE'])
crime_filtered['END_DATE'] = pd.to_datetime(crime_filtered['END_DATE'])
```

[63]:
```
print(crime_filtered.iloc[666])
# show
```

```
NEIGHBORHOOD_CLUSTER                                    26
CENSUS_TRACT                                          8100
offensegroup                                      property
LONGITUDE                                       -76.992842
END_DATE                                2017-09-23 10:59:44
SHIFT                                                 day
YBLOCK                                           135890.0
DISTRICT                                                1
WARD                                                    6
YEAR                                                 2017
sector                                                 12
PSA                                                   108
ucr-rank                                                7
VOTING_PRECINCT                                        85
XBLOCK                                           400621.0
BLOCK                   911 - 999 block of massachusetts avenue ne
START_DATE                              2017-09-23 01:00:06
CCN                                              17165240
OFFENSE                                       theft f/auto
ANC                                                    61
REPORT_DAT                              2017-09-23 14:59:46
METHOD                                             others
LATITUDE                                        38.890854
OFFENSE_code                                            2
```

```
Name: 4851, dtype: object
```

Creating new features Create additional feature SPAN to represent the span between the earliest crime and latest crime.

[64]:
```
crime_filtered_newfeature = crime_filtered.copy()
# SPAN = END_DATE - START_DATE
crime_filtered_newfeature['SPAN'] = (crime_filtered['END_DATE'] -␣
↪crime_filtered['START_DATE'])/np.timedelta64(1, 's')
```

Create new feature TIME_TO_REPORT to represent the time it took from crime to report it.

[65]:
```
# TIME_TO_REPORT = EPORT_DAT - END_DATE
crime_filtered_newfeature['TIME_TO_REPORT'] = (crime_filtered['REPORT_DAT'] -␣
↪crime_filtered['END_DATE'])/np.timedelta64(1, 's')
```

Create new feature DATE from START_DATE.

[66]:
```
crime_filtered_newfeature['DATE'] = pd.
↪to_datetime(crime_filtered['START_DATE'], format = '%d/%m/%Y %H:%M:%S')
```

Create new feature MONTH, DAY, hour, dayofyear, week, weekofyear, dayofweek, weekday, quarter from DATE.

[67]:
```
crime_filtered_newfeature['MONTH'] = pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).month
crime_filtered_newfeature['DAY'] = pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).day
crime_filtered_newfeature['hour'] = pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).hour
crime_filtered_newfeature['dayofyear'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).dayofyear
crime_filtered_newfeature['week'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).week
crime_filtered_newfeature['weekofyear'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).weekofyear
crime_filtered_newfeature['dayofweek'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).dayofweek
crime_filtered_newfeature['weekday'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).weekday
crime_filtered_newfeature['quarter'] =pd.
↪DatetimeIndex(crime_filtered_newfeature['DATE']).quarter
```

[68]:
```
crime_filtered_newfeature
# show
```

[68]:
```
        NEIGHBORHOOD_CLUSTER  CENSUS_TRACT offensegroup  LONGITUDE  \
2                          8          4702     property -77.020913
9                         27          7200     property -77.005025
```

```
10                        8          5800      violent -77.019909
23                        8          5800     property -77.022433
27                       26          6700     property -76.984577
…                         …             …            …         …
449163                   39          7304     property -76.988516
449169                   39          9807      violent -77.012634
449178                   39         10900     property -77.008913
449184                   38          7409     property -76.978424
449192                   39          9700     property -76.991523

                  END_DATE     SHIFT    YBLOCK  DISTRICT  WARD  YEAR  …  \
2       2017-04-29 11:10:57       day  137185.0         1     6  2017  …
9       2017-05-03 22:32:33   evening  134294.0         1     6  2017  …
10      2017-05-04 00:38:16  midnight  136801.0         1     2  2017  …
23      2017-03-21 12:34:19       day  136733.0         1     2  2017  …
27      2017-04-24 16:40:14   evening  135773.0         1     6  2017  …
…                       …         …         …         …   …    …  …
449163  2021-03-07 06:00:28       day  130527.0         7     8  2021  …
449169  2021-03-07 22:43:07  midnight  129279.0         7     8  2021  …
449178  2021-02-17 05:05:23       day  127956.0         7     8  2021  …
449184  2021-01-28 08:00:37   evening  130774.0         7     8  2021  …
449192  2021-03-22 03:50:49   evening  129205.0         7     8  2021  …

                     DATE  MONTH  DAY  hour  dayofyear week weekofyear  \
2       2017-04-29 10:43:33      4   29    10        119   17         17
9       2017-05-03 22:10:26      5    3    22        123   18         18
10      2017-05-03 23:35:24      5    3    23        123   18         18
23      2017-03-21 11:47:07      3   21    11         80   12         12
27      2017-04-24 16:38:50      4   24    16        114   17         17
…                       …    …  …   …         …  …         …
449163  2021-03-06 22:00:41      3    6    22         65    9          9
449169  2021-03-07 22:22:41      3    7    22         66    9          9
449178  2021-02-17 10:55:52      2   17    10         48    7          7
449184  2021-01-28 07:00:31      1   28     7         28    4          4
449192  2021-03-21 22:00:43      3   21    22         80   11         11

        dayofweek weekday quarter
2               5       5       2
9               2       2       2
10              2       2       2
23              1       1       1
27              0       0       2
…             …     …     …
449163          5       5       1
449169          6       6       1
449178          2       2       1
449184          3       3       1
```

```
449192          6       6       1

[440236 rows x 36 columns]
```

2-1-6 Data Scaling Do it during modeling if necessary.

2-1-7 Data Discretization Do it during modeling if necessary.

*Data preprocessing is preliminarily completed.

2-1-8 Visualization

[69]:
```
# define a function to draw geographic heatmap
def draw_heatmap(data,name):
    count = data.copy()
    count['count']=1
    grp_count = count.groupby(["LONGITUDE","LATITUDE"]).sum()['count']
    grp_pd=grp_count.reset_index()
    lon = np.array(grp_pd["LONGITUDE"])
    lat = np.array(grp_pd["LATITUDE"])
    number=np.array(grp_pd["count"]).astype(float)
    heatmap_plot = [ [lat[i],lon[i],number[i]] for i in range(len(grp_pd)) ]
    map_osm = folium.Map([38.906845, -77.013636],␣
↪tiles='stamentoner',zoom_start=14)
    HeatMap(heatmap_plot).add_to(map_osm)
    file_path = r"C:\\Users\\HP\\JupyterPythonProjects\\project\\map\\"+name+".
↪html"
    map_osm.save(file_path)
```

Geographical heatmap based on OFFENSE. The results are in the map file.

[70]:
```
theft_other=crime_filtered[crime_filtered['OFFENSE']=='theft/other']
theft_f_auto=crime_filtered[crime_filtered['OFFENSE']=='theft f/auto']
robbery=crime_filtered[crime_filtered['OFFENSE']=='robbery']
motor_vehicle_theft=crime_filtered[crime_filtered['OFFENSE']=='motor vehicle␣
↪theft']
burglary =crime_filtered[crime_filtered['OFFENSE']=='burglary']
assault_w_dangerous_weapon=crime_filtered[crime_filtered['OFFENSE']=='assault w/
↪dangerous weapon']
sex_abuse=crime_filtered[crime_filtered['OFFENSE']=='sex abuse']
homicide=crime_filtered[crime_filtered['OFFENSE']=='homicide']
arson=crime_filtered[crime_filtered['OFFENSE']=='arson']

draw_heatmap(theft_other,'offense_theft_other')
draw_heatmap(theft_f_auto,'offense_theft_f_auto')
draw_heatmap(robbery,'offense_robbery')
draw_heatmap(motor_vehicle_theft,'offense_motor_vehicle_theft')
draw_heatmap(burglary,'offense_burglary')
draw_heatmap(assault_w_dangerous_weapon,'offense_assault_w_dangerous_weapon')
```

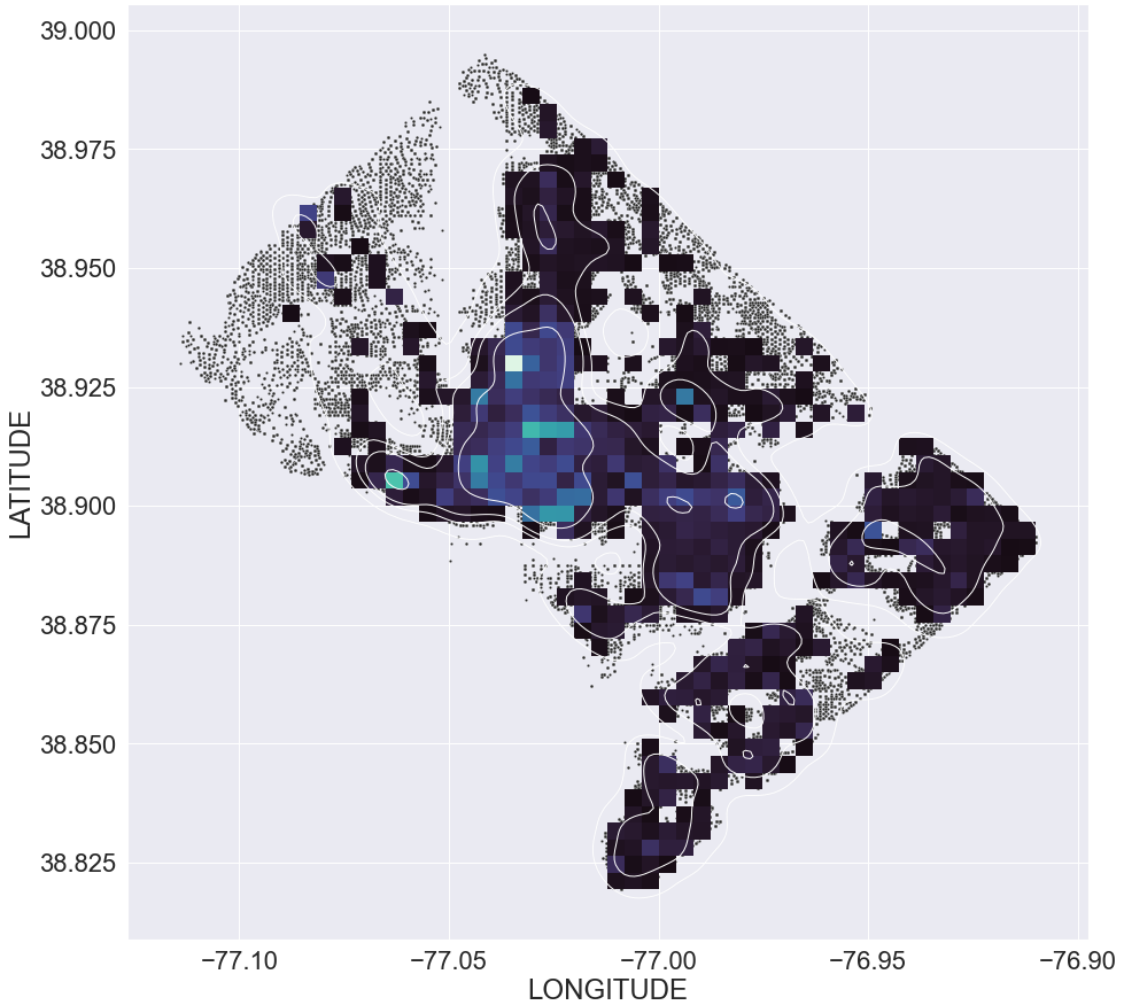
```
draw_heatmap(sex_abuse,'offense_sex_abuse')
draw_heatmap(homicide,'offense_homicide')
draw_heatmap(arson,'offense_arson')
```

[71]:
```
# crime data by years
crime_filtered_08=crime_filtered[crime_filtered.YEAR==2008]
crime_filtered_09=crime_filtered[crime_filtered.YEAR==2009]
crime_filtered_10=crime_filtered[crime_filtered.YEAR==2010]
crime_filtered_11=crime_filtered[crime_filtered.YEAR==2011]
crime_filtered_12=crime_filtered[crime_filtered.YEAR==2012]
crime_filtered_13=crime_filtered[crime_filtered.YEAR==2013]
crime_filtered_14=crime_filtered[crime_filtered.YEAR==2014]
crime_filtered_15=crime_filtered[crime_filtered.YEAR==2015]
crime_filtered_16=crime_filtered[crime_filtered.YEAR==2016]
crime_filtered_17=crime_filtered[crime_filtered.YEAR==2017]
crime_filtered_18=crime_filtered[crime_filtered.YEAR==2018]
crime_filtered_19=crime_filtered[crime_filtered.YEAR==2019]
crime_filtered_20=crime_filtered[crime_filtered.YEAR==2020]
crime_filtered_21=crime_filtered[crime_filtered.YEAR==2021]
```

General distribution.

[72]:
```
f, ax = plt.subplots(figsize=(15, 15))
sns.scatterplot(x=crime_filtered['LONGITUDE'], y=crime_filtered['LATITUDE'],␣
↪s=5, color=".15")
sns.histplot(x=crime_filtered['LONGITUDE'], y=crime_filtered['LATITUDE'],␣
↪bins=50, pthresh=.1, cmap="mako")
sns.kdeplot(x=crime_filtered['LONGITUDE'], y=crime_filtered['LATITUDE'],␣
↪levels=5, color="w", linewidths=1)
```

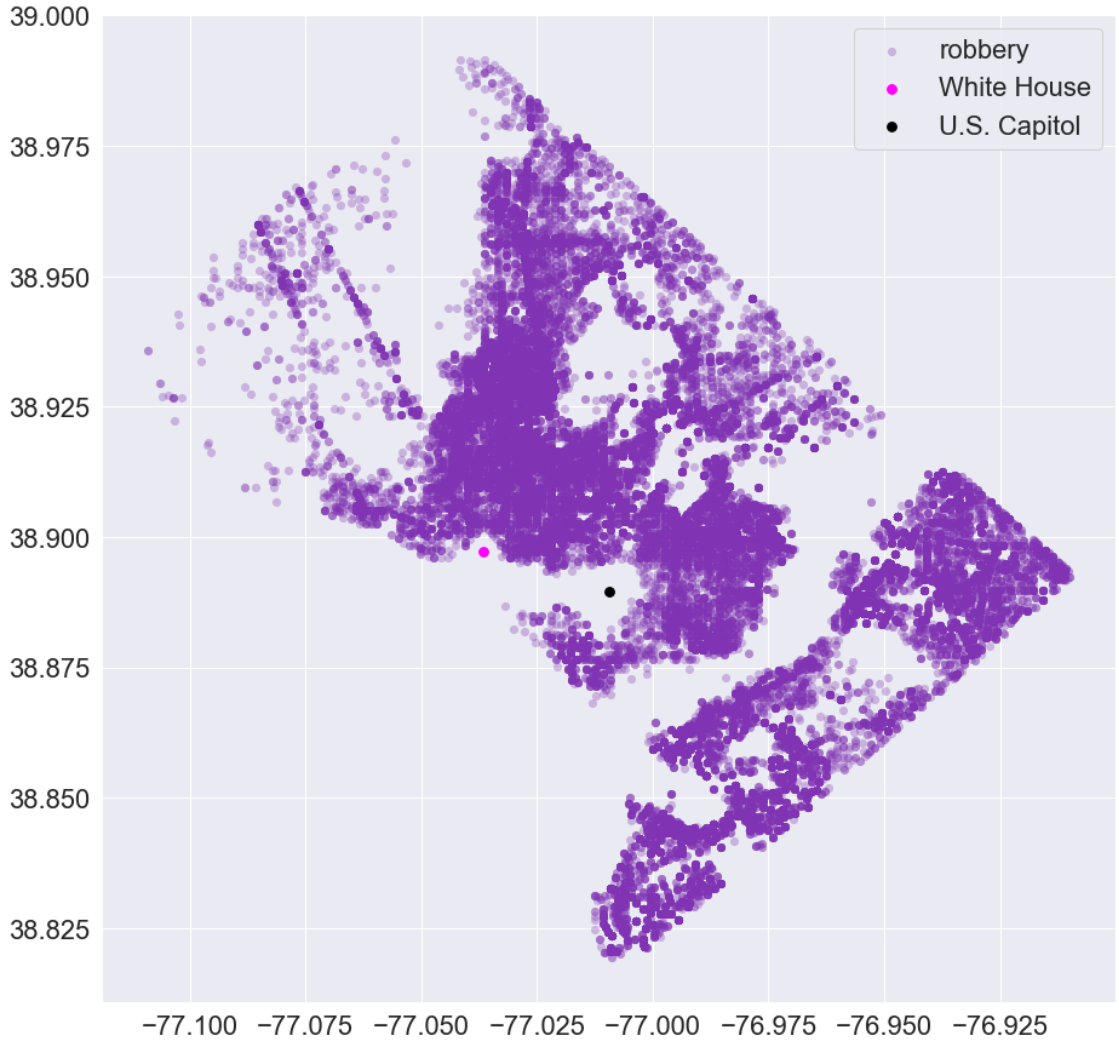
[72]:



Distribution of violent crimes by OFFENSE.

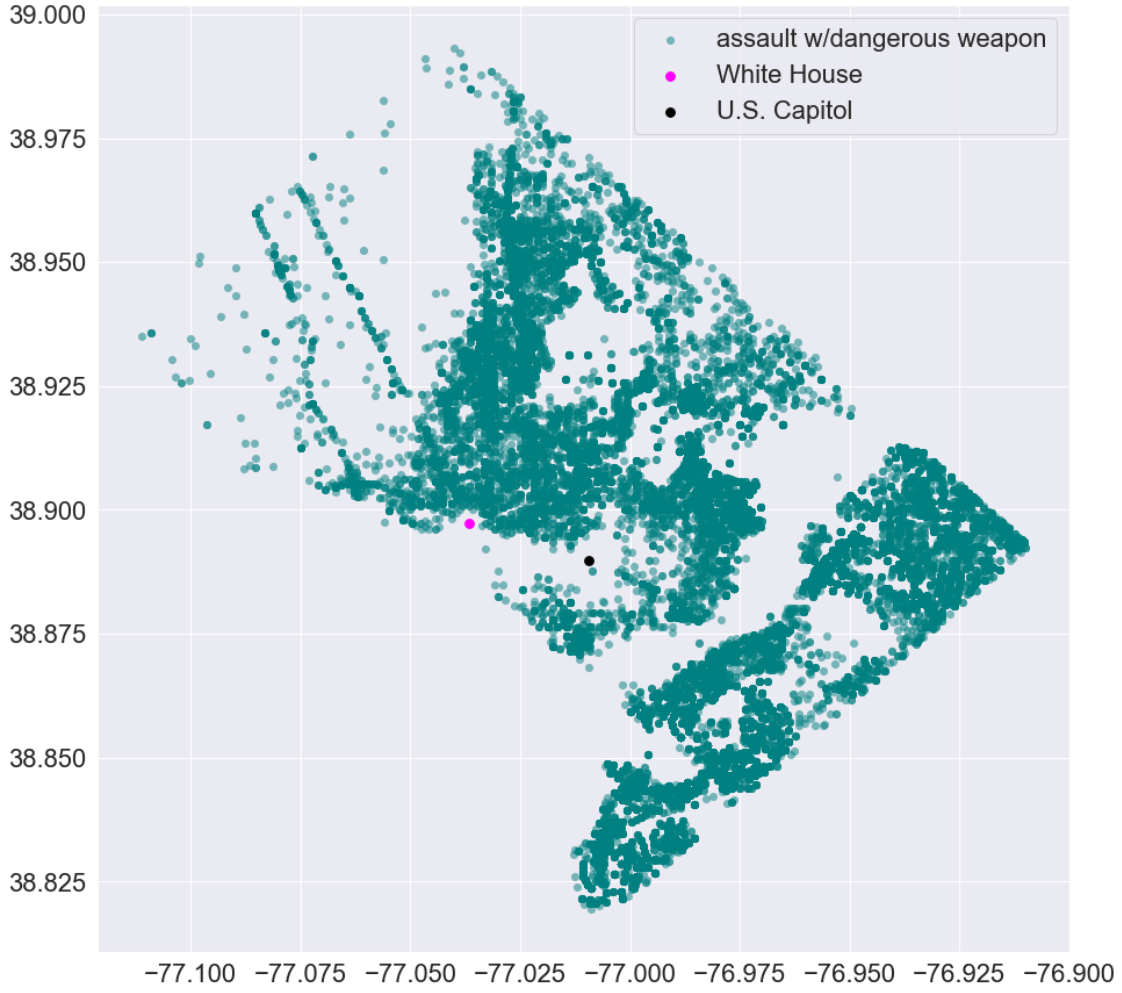
[73]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['OFFENSE']=='robbery'],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['OFFENSE']=='robbery'], s=50,␣
 ↪alpha=0.3, color=[0.5,0.2,0.7], lw=0, label='robbery')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
# robbery
```

[73]:



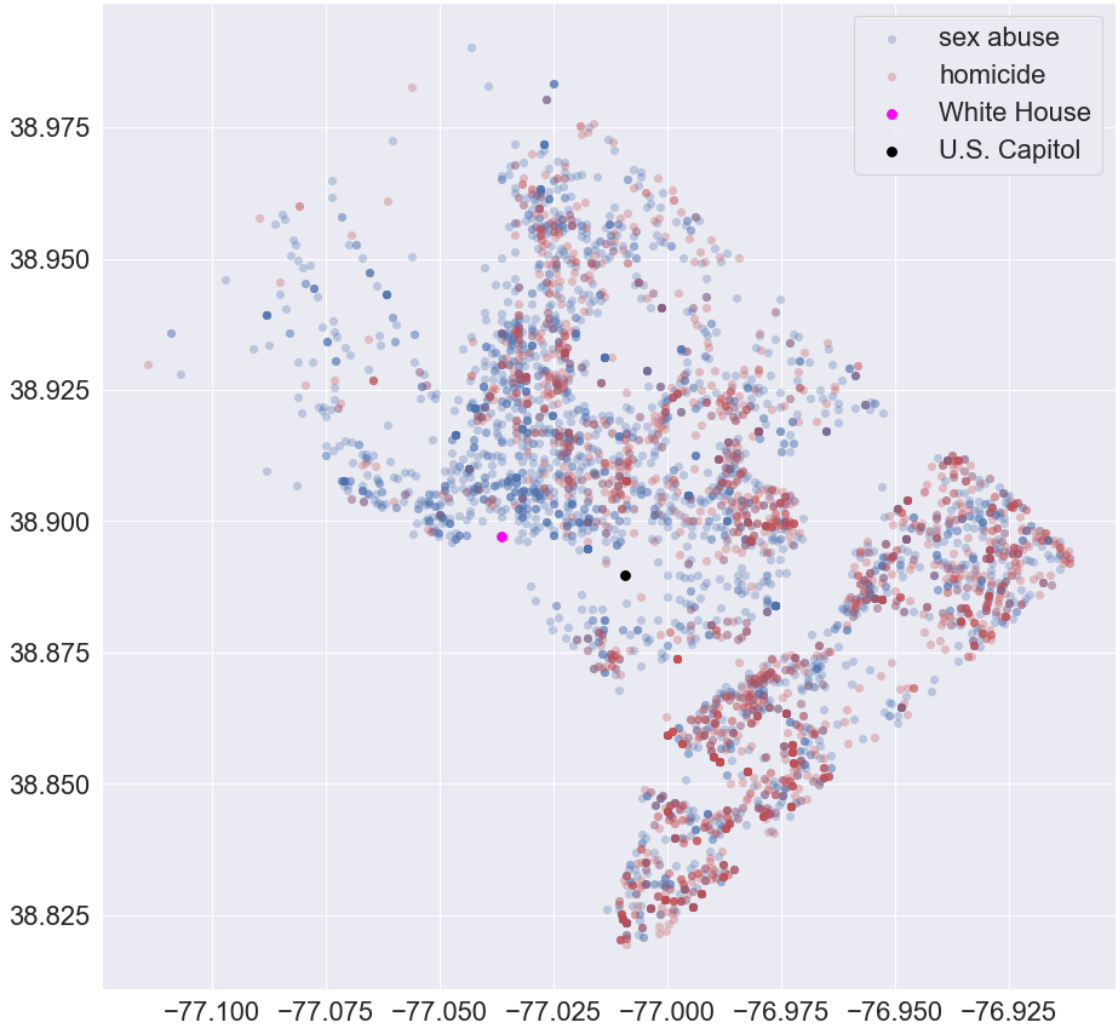
[74]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['OFFENSE']=='assault w/
 ↪dangerous weapon'],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['OFFENSE']=='assault w/dangerous␣
 ↪weapon'], s=50, alpha=0.5, color=[0,0.5,0.5], lw=0, label='assault w/
 ↪dangerous weapon')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
# assault w/dangerous weapon
```

[74]:



[75]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['OFFENSE']=='sex␣
 ↪abuse'], crime_filtered['LATITUDE'][crime_filtered['OFFENSE']=='sex abuse'],␣
 ↪s=50, alpha=0.3, color='b', lw=0, label='sex abuse')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['OFFENSE']=='homicide'],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['OFFENSE']=='homicide'], s=50,␣
 ↪alpha=0.3, color='r', lw=0, label='homicide')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
# sex abuse and homicide
```

[75]:



Distribution under offense and method conditions.

[76]:
```
sns.relplot(data=crime_filtered, x="LONGITUDE", y="LATITUDE",col="OFFENSE",␣
↪hue="METHOD", style="METHOD",kind="scatter",palette="flare")
```

[76]:
```
<seaborn.axisgrid.FacetGrid at 0x2c3b1c897c0>
```

Distribution under ucr-rank and method conditions.

[77]:
```
sns.relplot(data=crime_filtered, x="LONGITUDE", y="LATITUDE",col="ucr-rank",␣
↪hue="METHOD", style="METHOD",kind="scatter",palette="crest")
```

[77]:
```
<seaborn.axisgrid.FacetGrid at 0x2c3b05c5520>
```

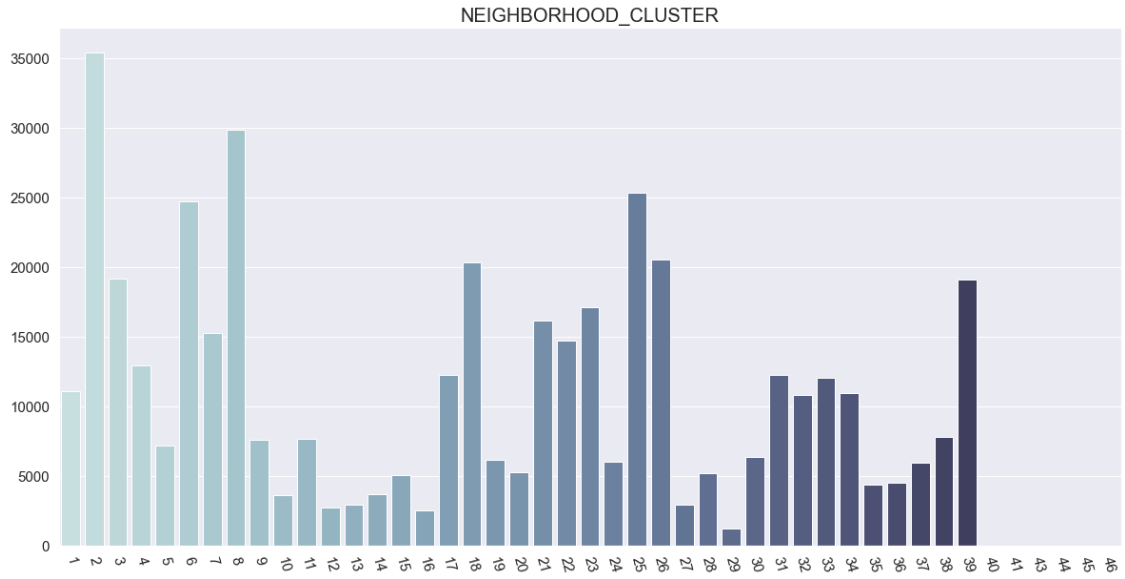
[78]:
```
# define a function to draw bar plots
def bar_plot(data,title,a,b,c,d,e,color,T):
    ax,fig=plt.subplots(figsize=(d,e))
    new_data=data.copy()
    if T==0:
        new_data.columns=['a','b']
    if T==1:
        new_data.columns=['b','a']
    sns.set_theme(style="darkgrid")
    sns.barplot(data=new_data,x="a",y="b",palette=color)
    plt.title(title,size=a)
    plt.xlabel("",size=a)
    plt.ylabel("",size=a)
    plt.tick_params(labelsize=b)
    plt.xticks(rotation=-c)
    plt.show()
    return()
```

[79]:
```
# data for bar plot
NEIGHBORHOOD_CLUSTER=pd.DataFrame(crime_filtered['NEIGHBORHOOD_CLUSTER'].
↪value_counts()).reset_index()
offensegroup=pd.DataFrame(crime_filtered['offensegroup'].value_counts()).
↪reset_index()
SHIFT=pd.DataFrame(crime_filtered['SHIFT'].value_counts()).reset_index()
sector=pd.DataFrame(crime_filtered['sector'].value_counts()).reset_index()
CENSUS_TRACT=pd.DataFrame(crime_filtered['CENSUS_TRACT'].value_counts()).
↪reset_index()
VOTING_PRECINCT=pd.DataFrame(crime_filtered['VOTING_PRECINCT'].value_counts()).
↪reset_index()
OFFENSE=pd.DataFrame(crime_filtered['OFFENSE'].value_counts()).reset_index()
ANC=pd.DataFrame(crime_filtered['ANC'].value_counts()).reset_index()
METHOD=pd.DataFrame(crime_filtered['METHOD'].value_counts()).reset_index()
DISTRICT=pd.DataFrame(crime_filtered['DISTRICT'].value_counts()).reset_index()
WARD=pd.DataFrame(crime_filtered['WARD'].value_counts()).reset_index()
YEAR=pd.DataFrame(crime_filtered['YEAR'].value_counts()).reset_index()
```

```
PSA=pd.DataFrame(crime_filtered['PSA'].value_counts()).reset_index()
ucr_rank=pd.DataFrame(crime_filtered['ucr-rank'].value_counts()).reset_index()
```

Visualization for character features.

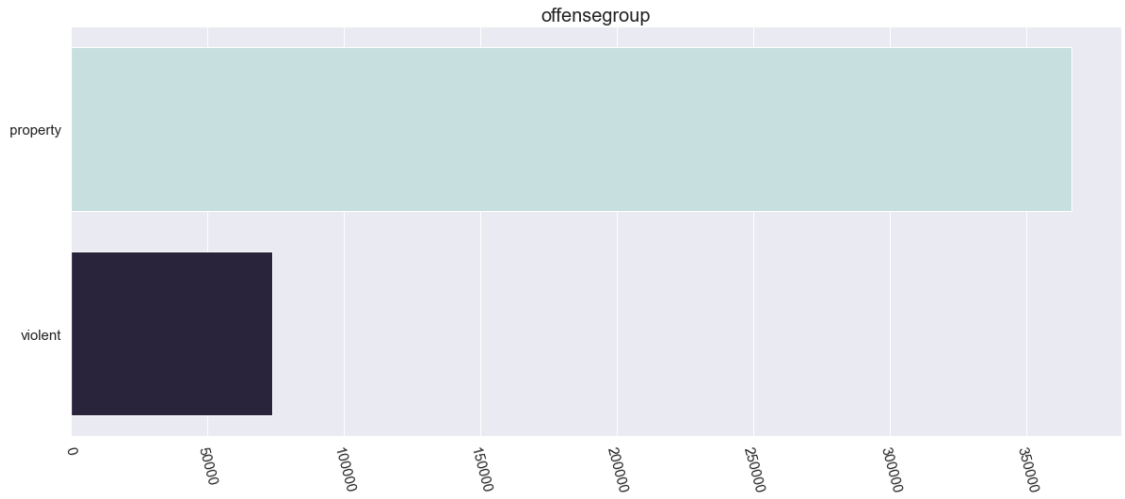
[80]:
```
bar_plot(NEIGHBORHOOD_CLUSTER,'NEIGHBORHOOD_CLUSTER',20,15,75,20,10,"ch:start=.
↪2,rot=-.3",0)
```



[80]: ()

*Conclusion: cluster2 and cluster8 had large amount of crimes while cluster40,41,43,44,45 and 46 had few crimes.

[81]:
```
bar_plot(offensegroup,'offensegroup',20,15,75,20,8,"ch:start=.2,rot=-.3",1)
```



[81]: ()

[82]:
```
crime_filtered['offensegroup'].value_counts()
```

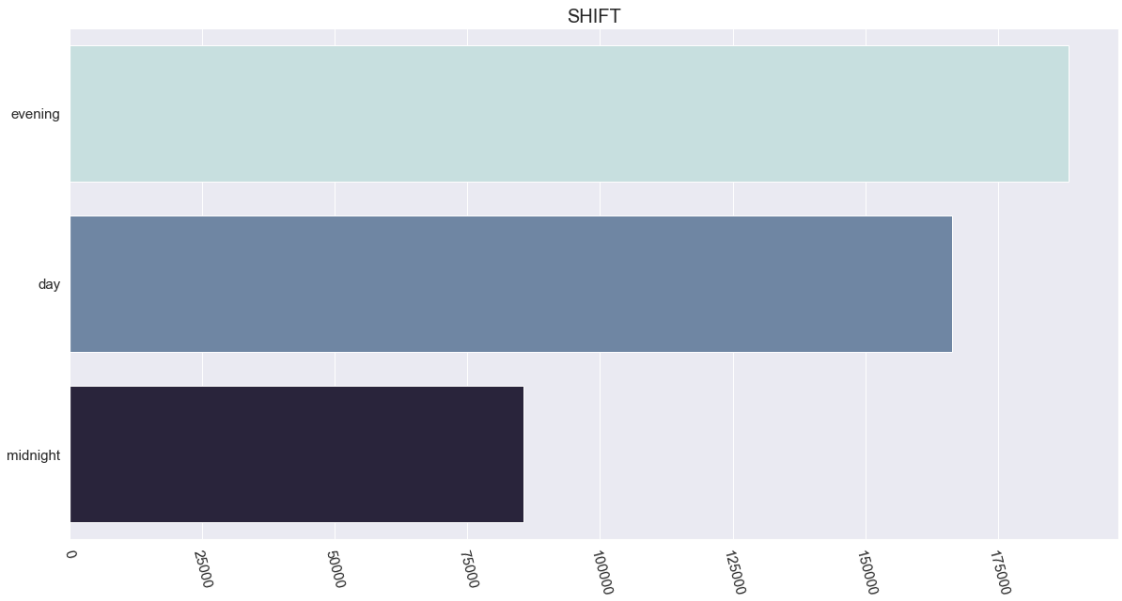
[82]:
```
property    366260
violent      73976
Name: offensegroup, dtype: int64
```

[83]:
```
367569/(367569+74624)
```

[83]: 0.8312411096512157

*Conclusion: about 83 percent of crimes were motivated by property, and only about 17 percent were violent.

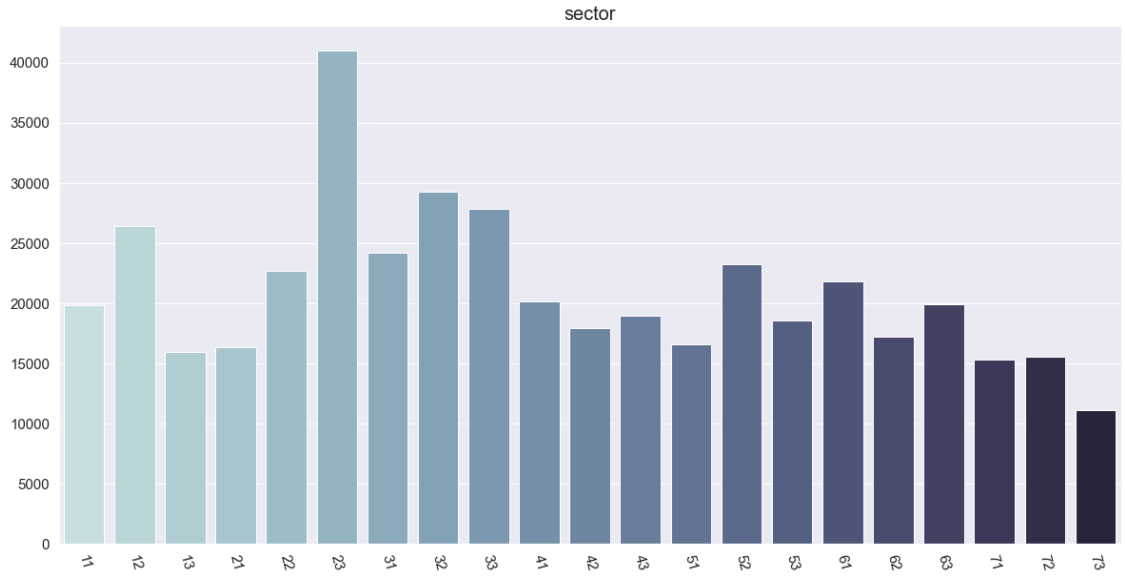
[84]:
```
bar_plot(SHIFT,'SHIFT',20,15,75,20,10,"ch:start=.2,rot=-.3",1)
```



[84]: ()

*Conclusion: most of the shifts of case reporting occurred in the evening, and even in the midnight.

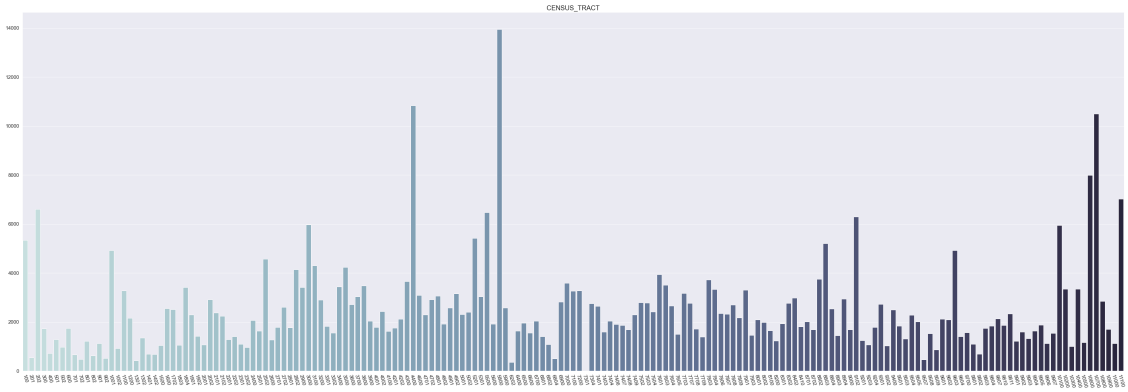
[85]:
```
bar_plot(sector,'sector',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
# e.g.  1D2 --- 12
```



[85]: ()

*Conclusion: sector 2D3 received the most cases while sector 7D3 received the least cases.

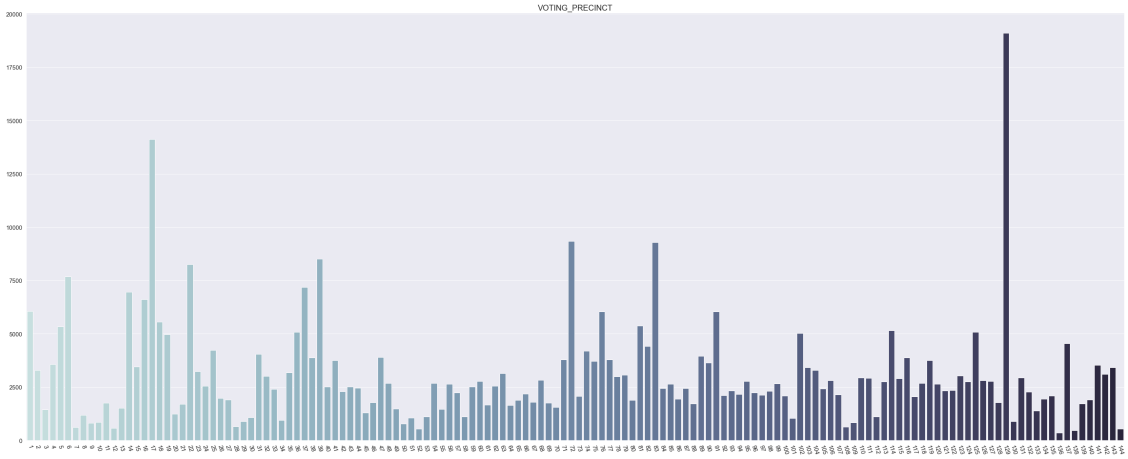
[86]:
```
bar_plot(CENSUS_TRACT,'CENSUS_TRACT',20,15,75,60,20,"ch:start=.2,rot=-.3",0)
```



[86]: ()

*Conclusion: CENSUS_TRACT 5800, 4400 and 10700 had large amount of crimes while CENSUS_TRACT 7301 had no crimes.

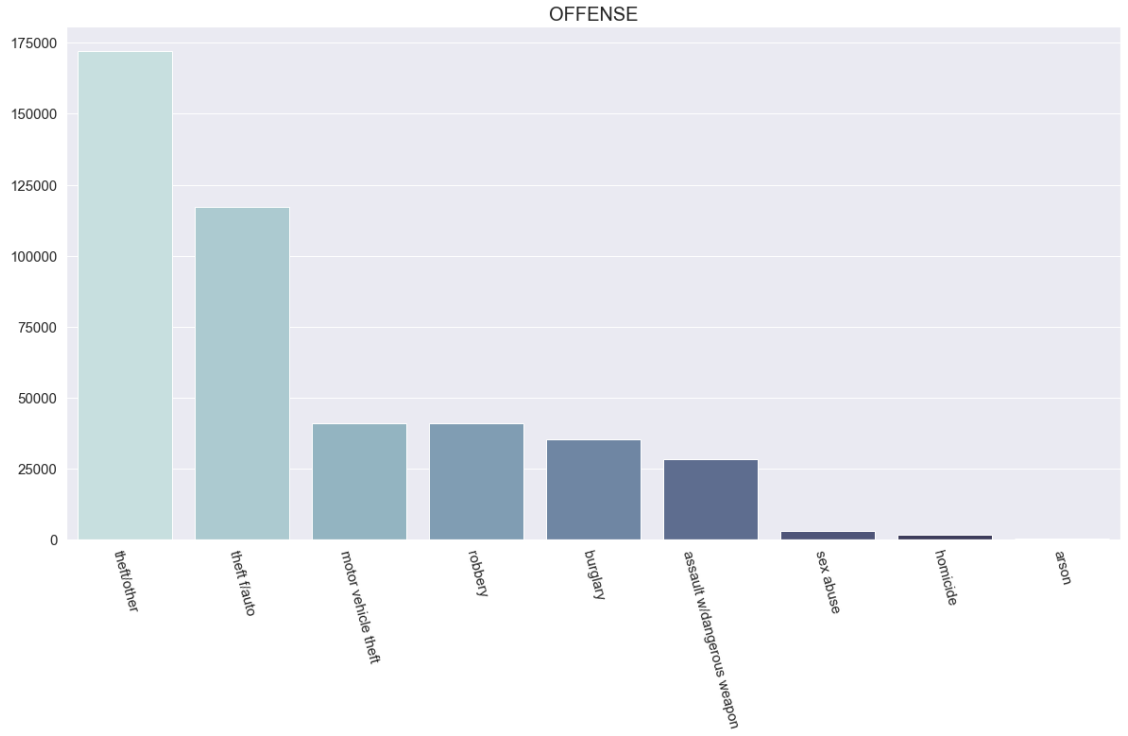
[87]:
```
bar_plot(VOTING_PRECINCT,'VOTING_PRECINCT',20,15,75,50,20,"ch:start=.2,rot=-.
↪3",0)
```



[87]: ()

*Conclusion: precinct129 and precinct 17 had large amount of crimes while precinct 126, precinct 138, precinct 144, precinct 52, precinct 12, precinct 7, precinct 108 and precinct 28 had few crimes.

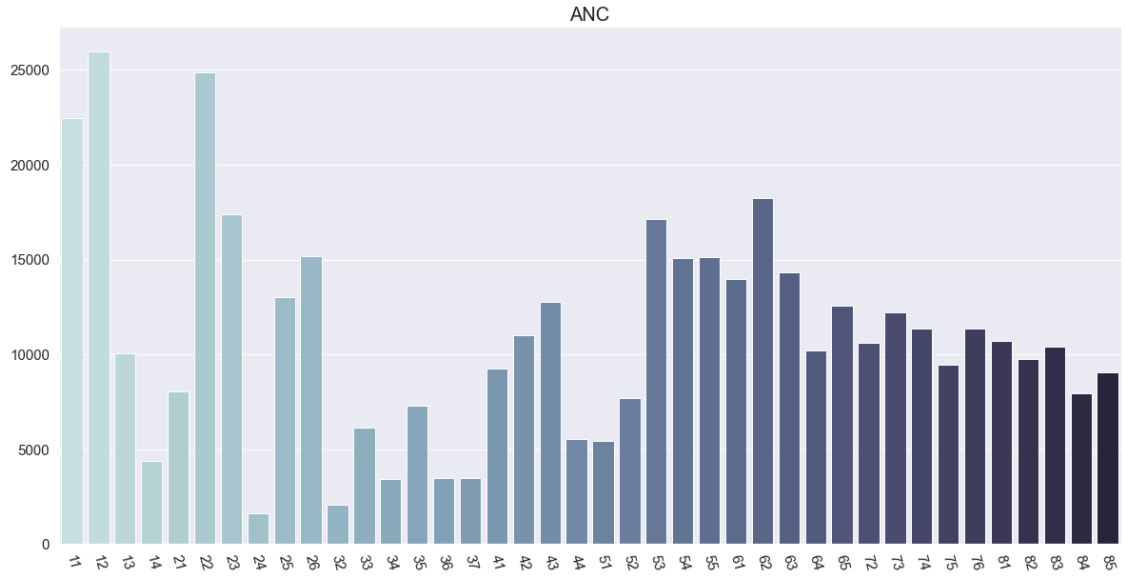
[88]:
```
bar_plot(OFFENSE,'OFFENSE',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



[88]: ()

*Conclusion: most crimes were theft. Sex abuse, homicide and arson were rare.

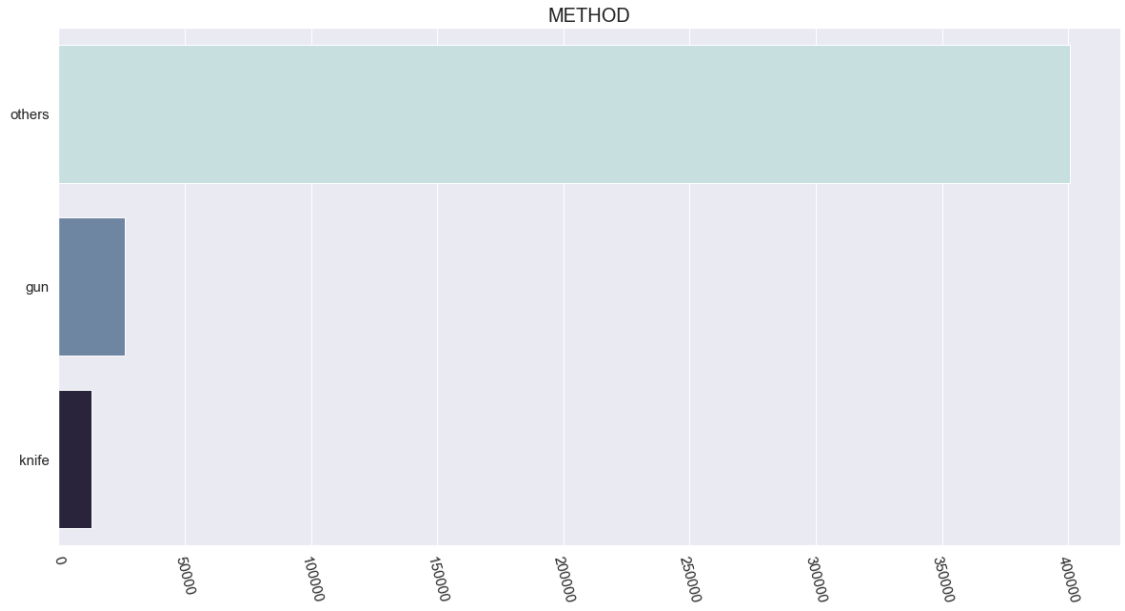
[89]:
```
bar_plot(ANC,'ANC',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
# A-1; B-2; C-3; D-4; E-5; F-6; G-7
```



[89]: ()

*Conclusion: 1B, 2B and 1A had large amount of crimes while 2D and 3B had few crimes.

[90]:
```
bar_plot(METHOD,'METHOD',20,15,75,20,10,"ch:start=.2,rot=-.3",1)
```



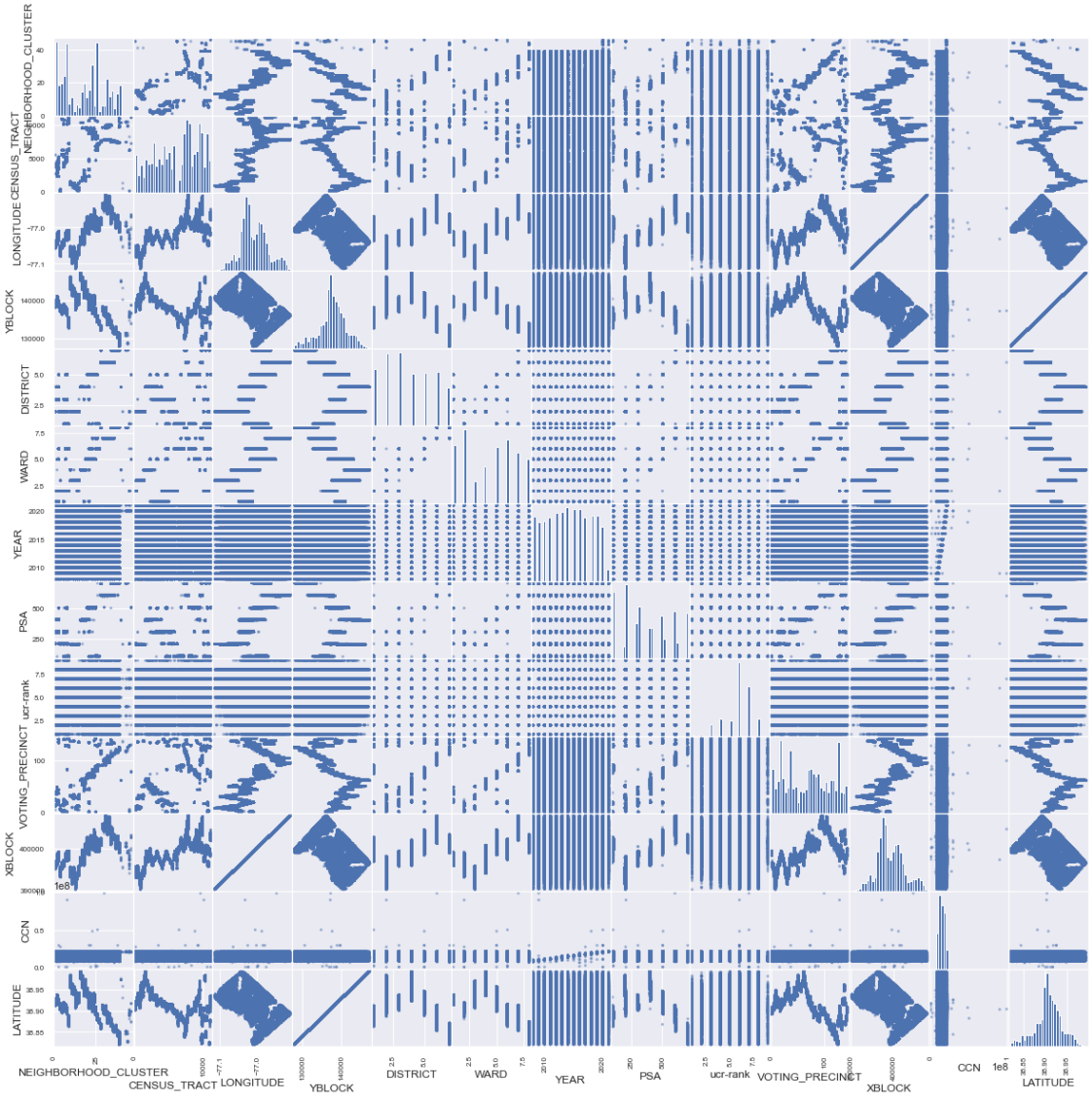
[90]: ()

*Conclusion: only few crimes had dangerous weapons like gun and knife, most crimes used other methods.

Visualization for numerical features.

[91]:
```
from pandas.plotting import scatter_matrix
scatter_matrix(crime_filtered, hist_kwds={'bins':30}, figsize=(20, 20))
# correlation between features
```

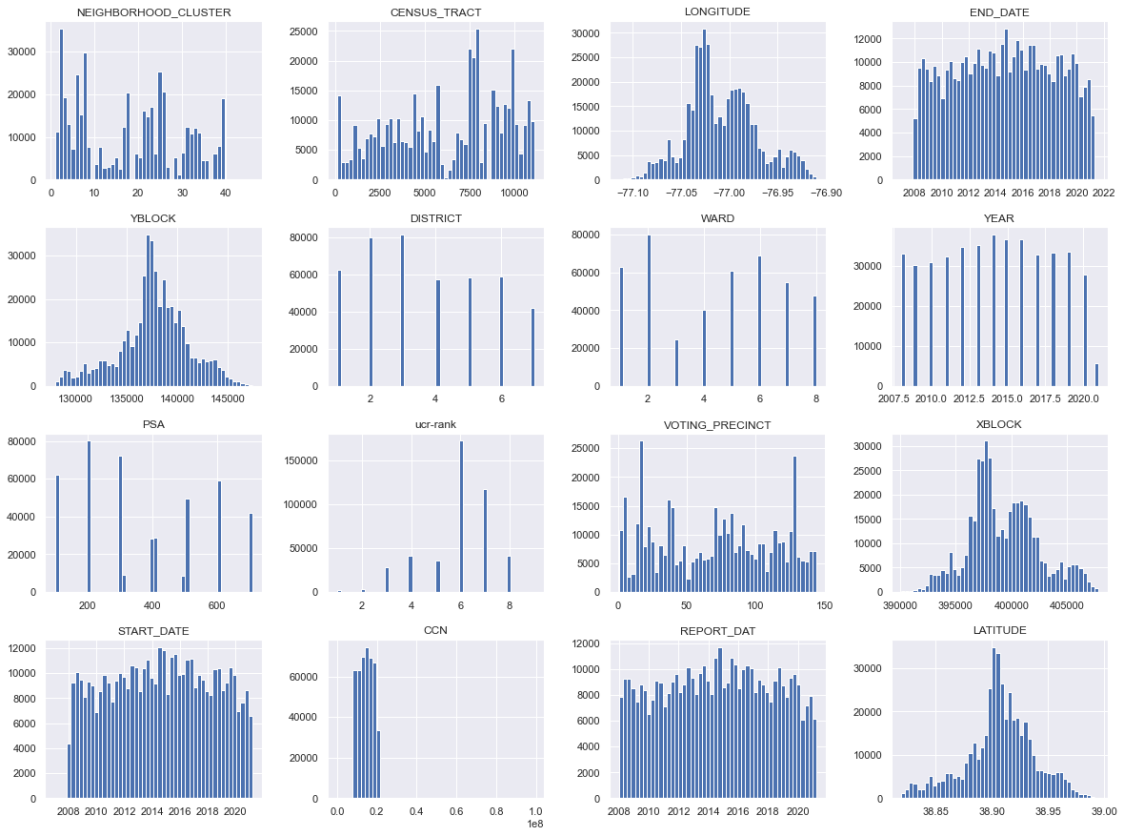
[91]:



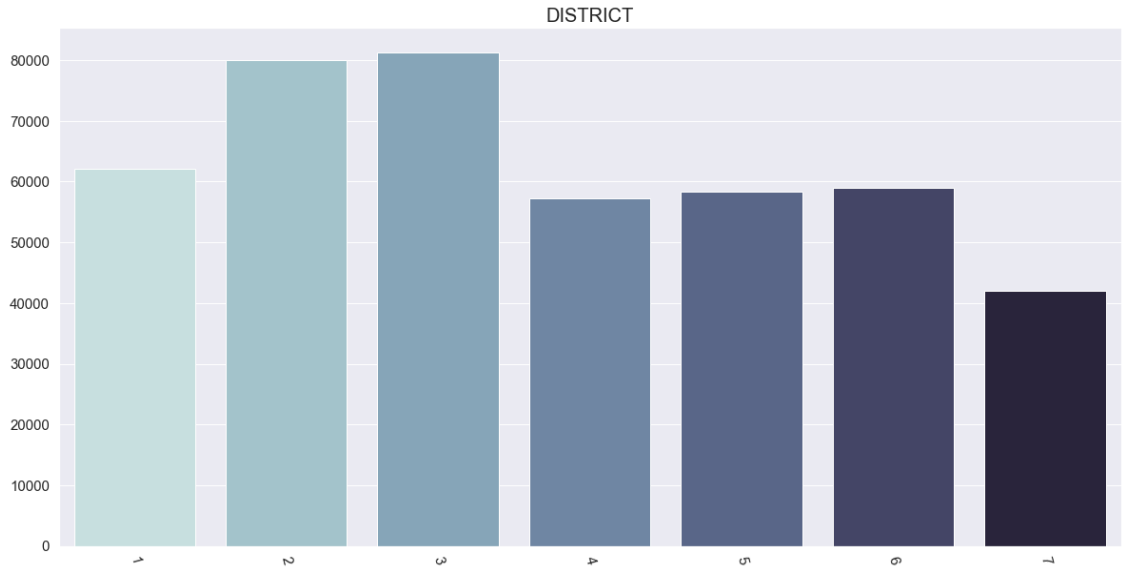
*Conlusion: XBLOCK and LONGITUDE are linear dependent. YBLOCK and LATITUDE are linear dependent. We can infer that block indices are based on latitude and longitude.

[92]:
```
crime_filtered.hist(bins=50,figsize=(20,15))
plt.show()
# bar plot
```

[92]:



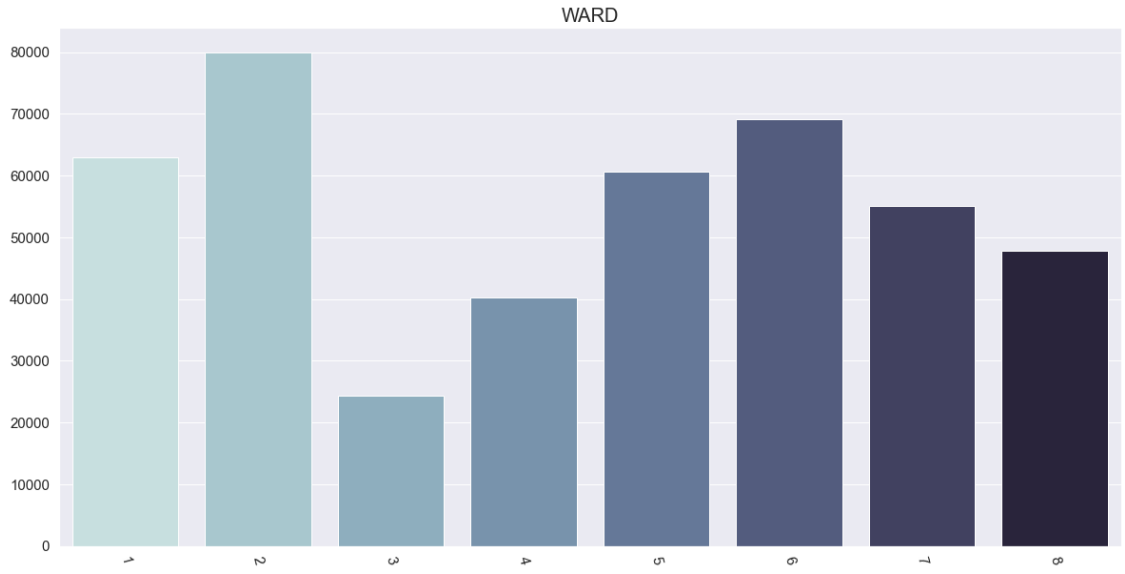
[93]:
```
bar_plot(DISTRICT,'DISTRICT',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



[93]: ()

*Conclusion: district 2 and 3 had large amount of crimes while district 7 had few crimes.

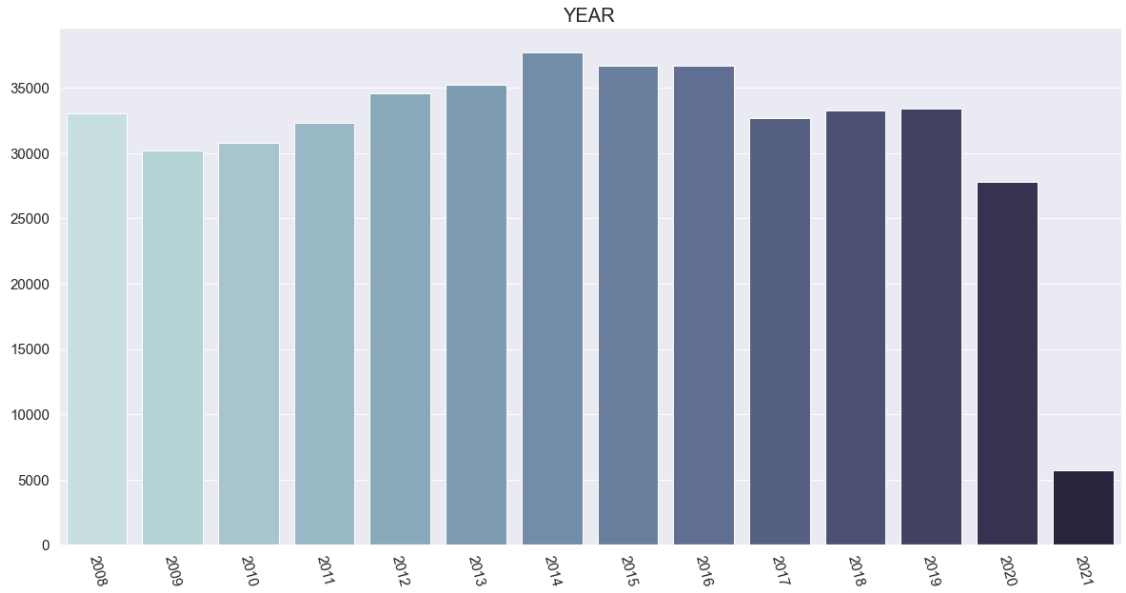
[94]:
```
bar_plot(WARD,'WARD',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



[94]: ()

*Conclusion: ward 2 had the most of crimes while ward 3 had the fewest crimes.

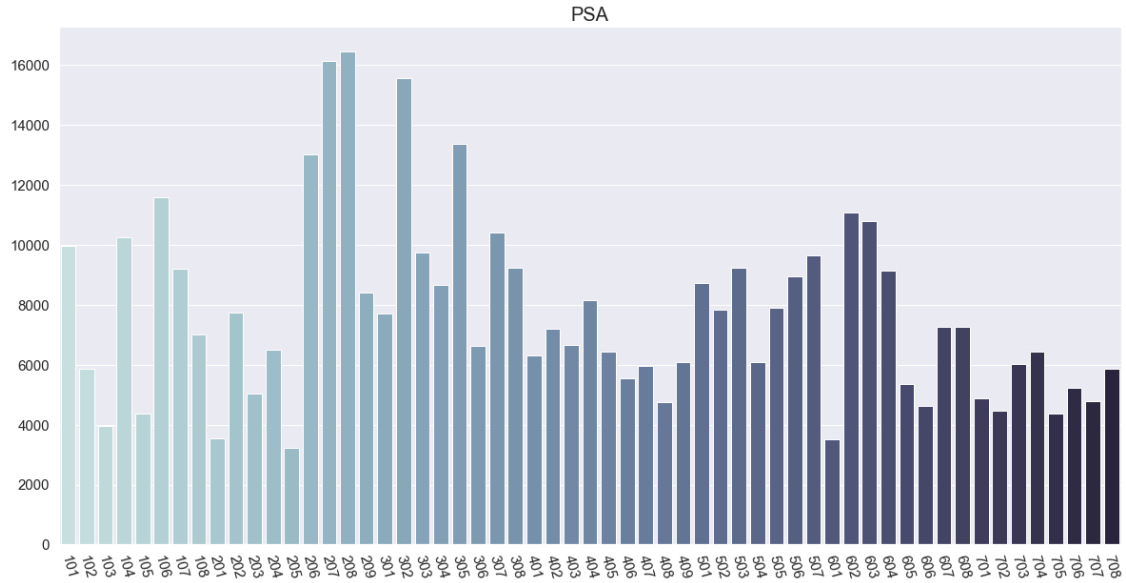
[95]:
```
bar_plot(YEAR,'YEAR',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



[95]:
```
()
```

*Conclusion: there was a largest amount of crimes in 2014 while there were the fewest crimes in 2020. Before 2014, it showed an upward trend, while after 2014, it showed a slow downward trend.

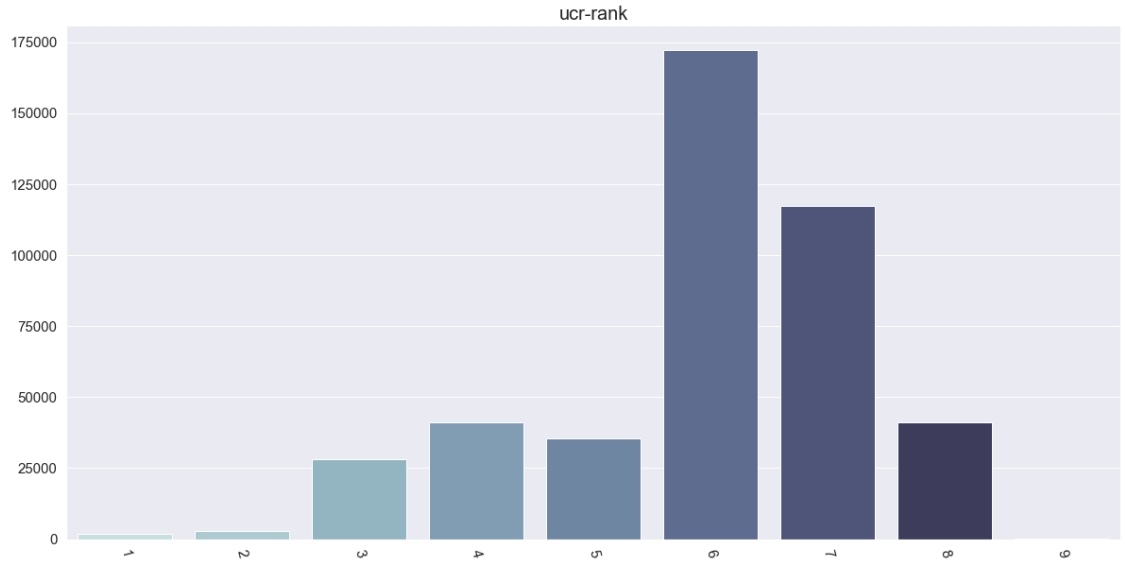
[96]:
```
bar_plot(PSA,'PSA',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



[96]: ()

*Conclusion: 208, 207 and 302 police station area received the largest amount of cases while 205 and 601 police station area received few cases.

[97]:
```
bar_plot(ucr_rank,'ucr-rank',20,15,75,20,10,"ch:start=.2,rot=-.3",0)
```



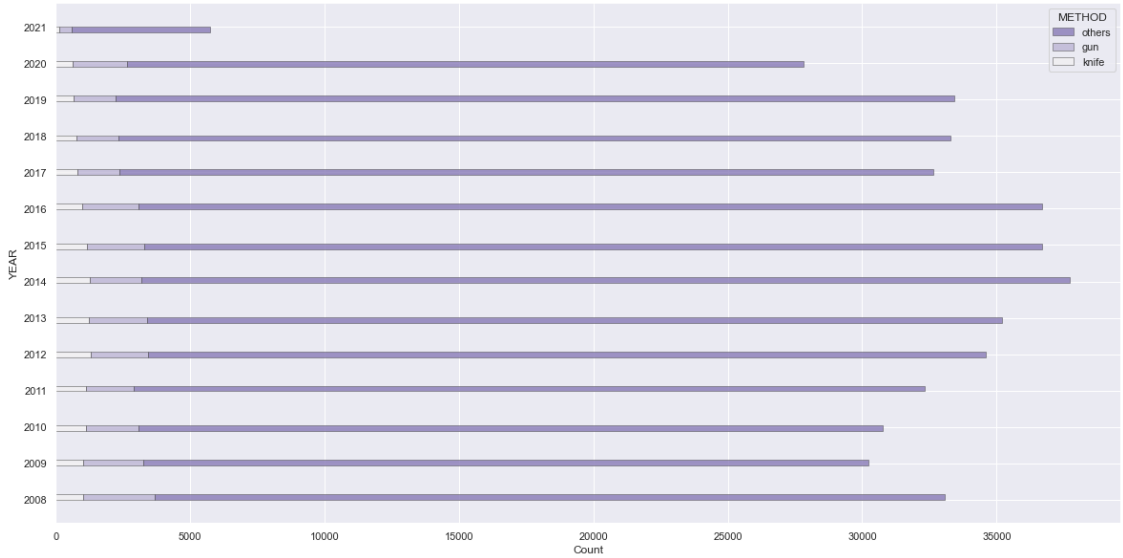
[97]: ()

*Conclusion: most of the crimes had a ucr-rank 6, some were 7. Few crimes had ucr-rank 1, 2 and 9. In other words, most crimes were severe and few crimes were mild.

[98]:
```
# define a function to draw bar plots with conditions
def barplotwith(name):
    f, ax = plt.subplots(figsize=(20, 10))
    sns.despine(f)
    sns.histplot(crime_filtered, y="YEAR",␣
 ↪hue=name,multiple="stack",palette="light:m_r",edgecolor=".3",linewidth=.
 ↪5,log_scale=False,)
    ax.xaxis.set_major_formatter(mpl.ticker.ScalarFormatter())
    ax.
 ↪set_yticks([2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021])
```

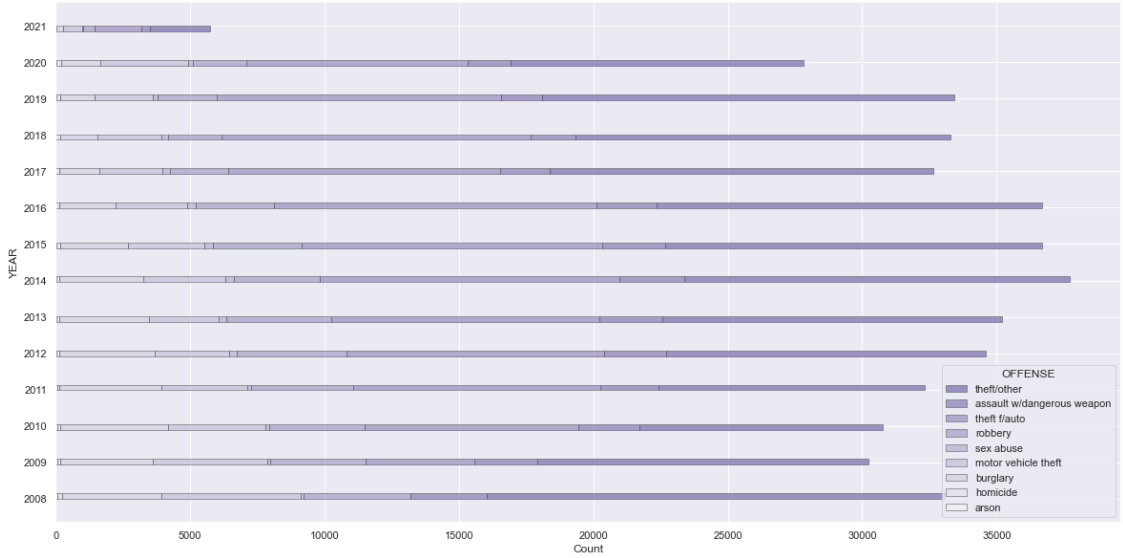
Condition on METHOD.

[99]:
```
barplotwith('METHOD')
```



Condition on OFFENSE.

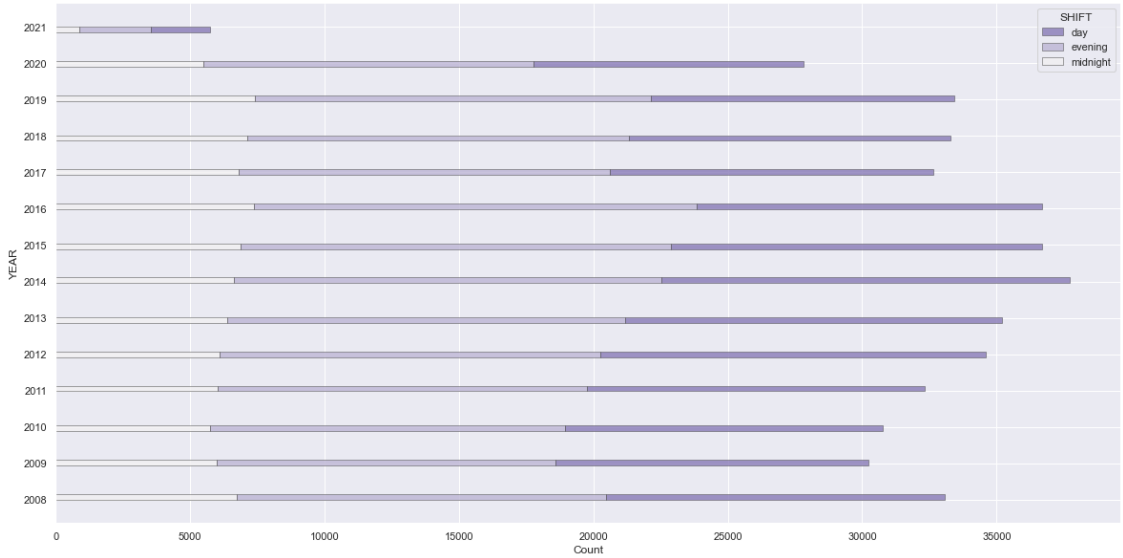
[100]:
```
barplotwith('OFFENSE')
```



*No theft f/auto in 2008.

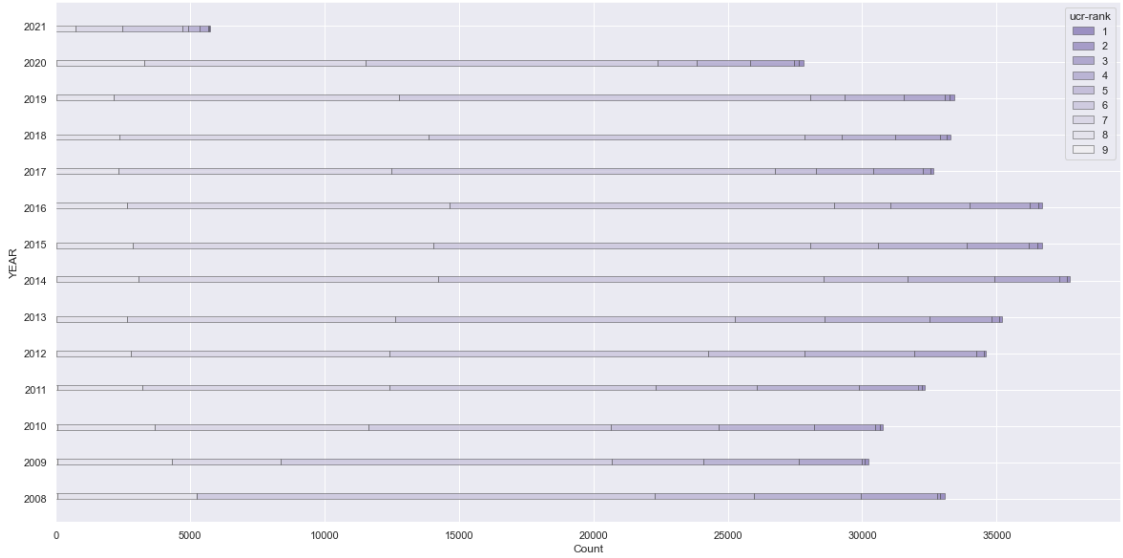
Condition on SHIFT.

[101]:
```
barplotwith('SHIFT')
```



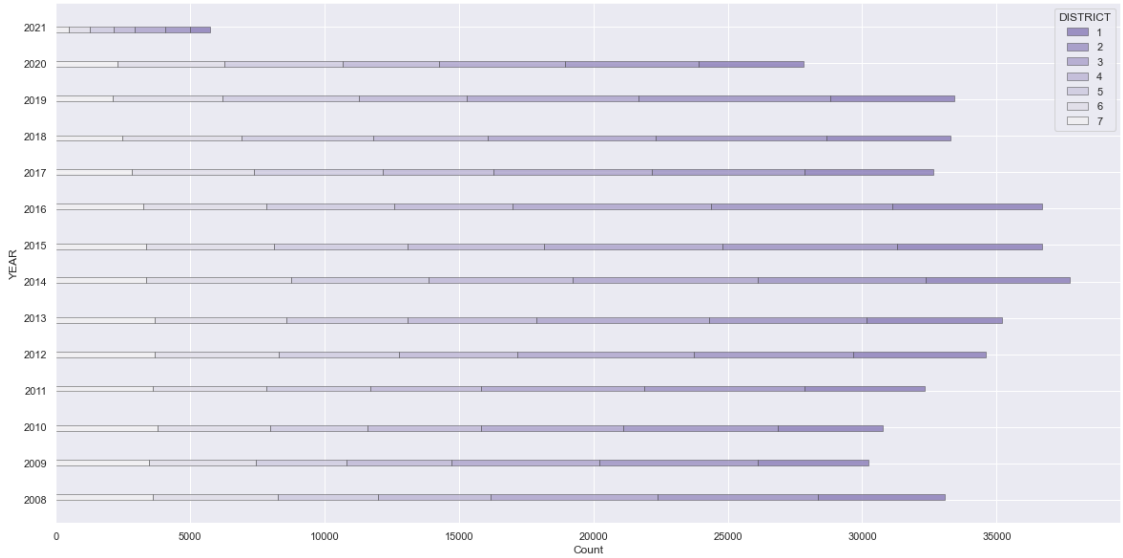
Condition on ucr-rank.

[102]:
```
barplotwith('ucr-rank')
```



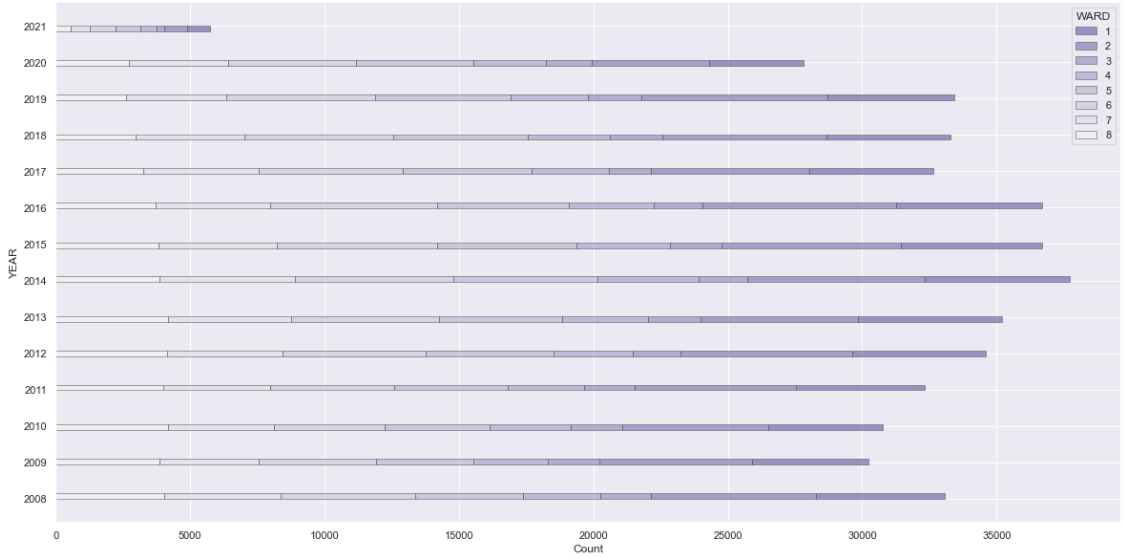
Condition on DISTRICT.

[103]:
```
barplotwith('DISTRICT')
```



Condition on WARD.

[104]:
```
barplotwith('WARD')
```

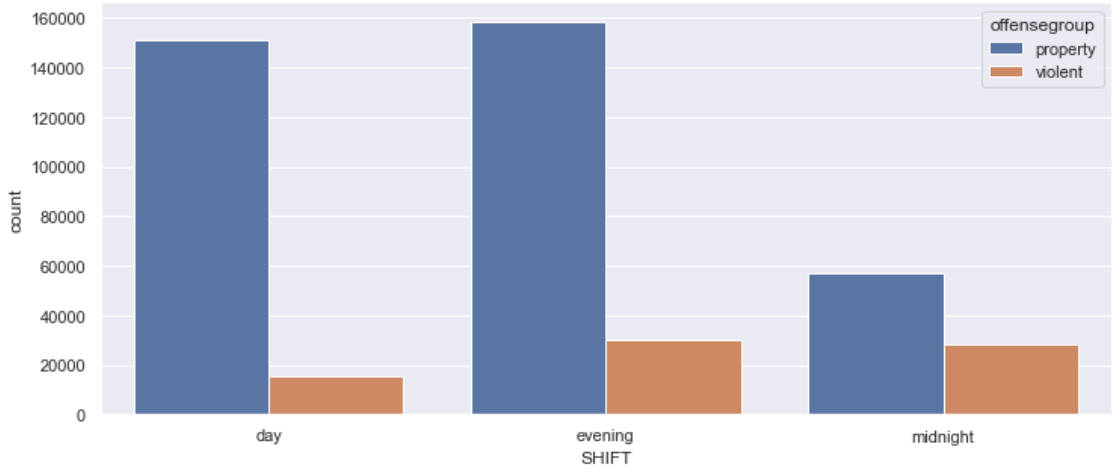


Other intersting visualizations.

[105]:
```
# define a function to draw bar plots with conditions (not by year)
def combine(a,b):
    temp = crime_filtered_newfeature[[a, b]]
    plt.figure(figsize=(12,5))
```

```
    sns.countplot(x=a, hue=b, data=temp)
```

[106]:
```
combine('SHIFT','offensegroup')
# crimes by SHIFT and offensegroup
```

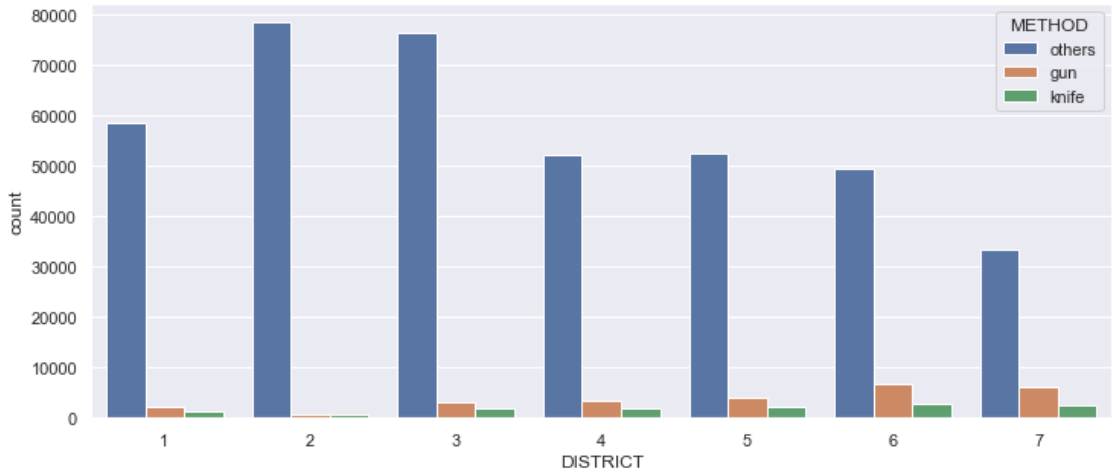


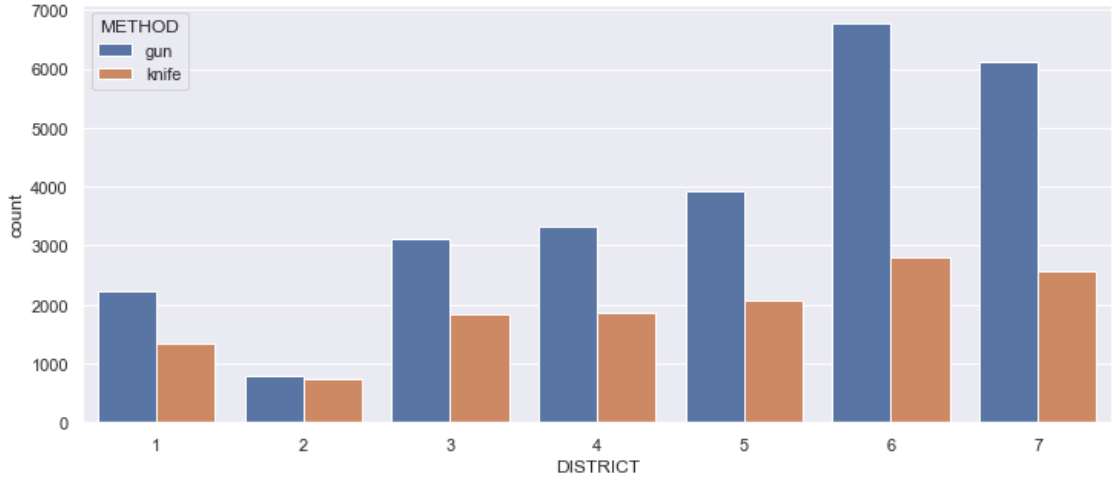
*Conlusion: It is dangerous at night since most violent crimes happened in evening or midnight.

[107]:
```
combine('DISTRICT','METHOD')

temp = crime_filtered_newfeature[['DISTRICT', 'METHOD']]
temp = temp[temp['METHOD'] != "others"]
plt.figure(figsize=(12, 5))
sns.countplot(x='DISTRICT', hue='METHOD', data=temp)
# crimes by DISTRICT and METHOD
```

[107]:
```
<Figure size 864x360 with 0 Axes>
```

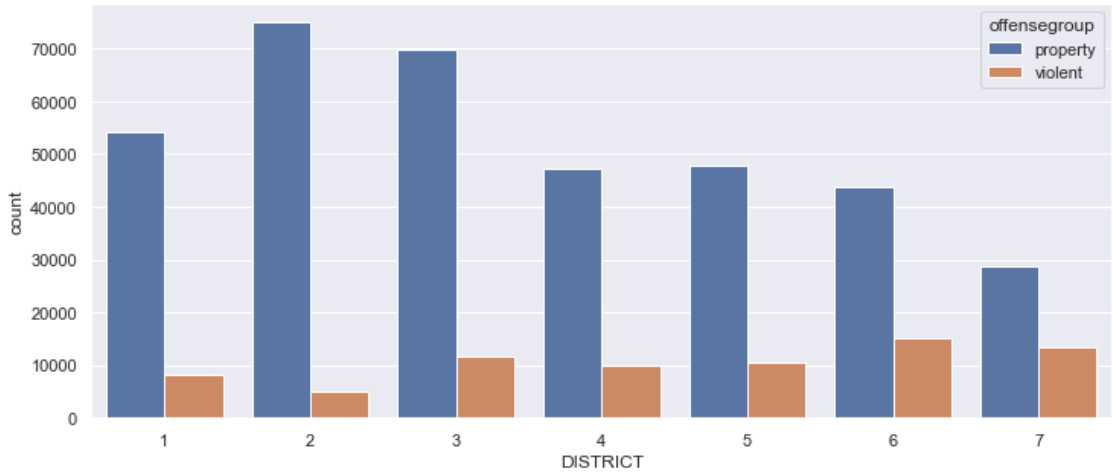
[107]:
```
<AxesSubplot:xlabel='DISTRICT', ylabel='count'>
```





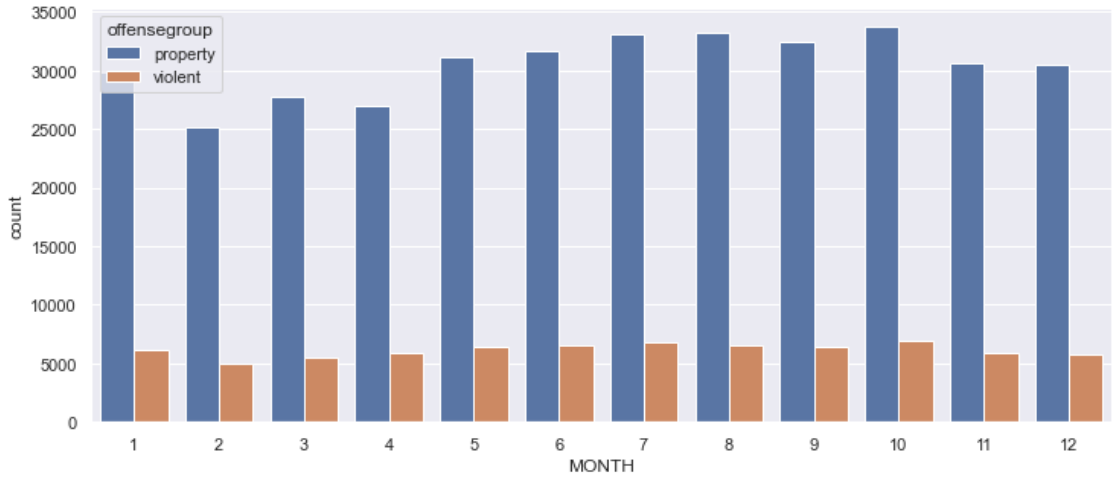
*Conclusion: Although district 2 and 3 have many crimes, most of them did not use knife or gun. District 6 and 7 have relatively few crimes, but the frequency of knife and gun increased.

[108]:
```
combine('DISTRICT','offensegroup')
# crimes by DISTRICT and offensegroup
```



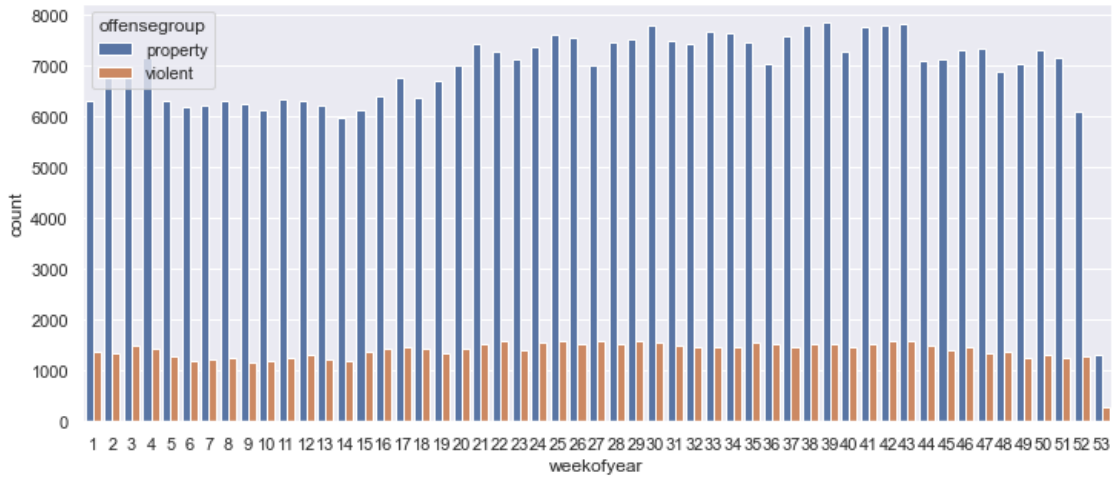
*Conclusion: most of crimes in district 2 and 3 were property while violent crimes had greater proportion in district 6 and 7. The reason may be that most crimes with method 'gun' or 'knife' were violent while most crimes with method 'others' were property, which corresponds to previous conclusion.

[109]:
```
combine('MONTH','offensegroup')
# crimes by MONTH and offensegroup
```

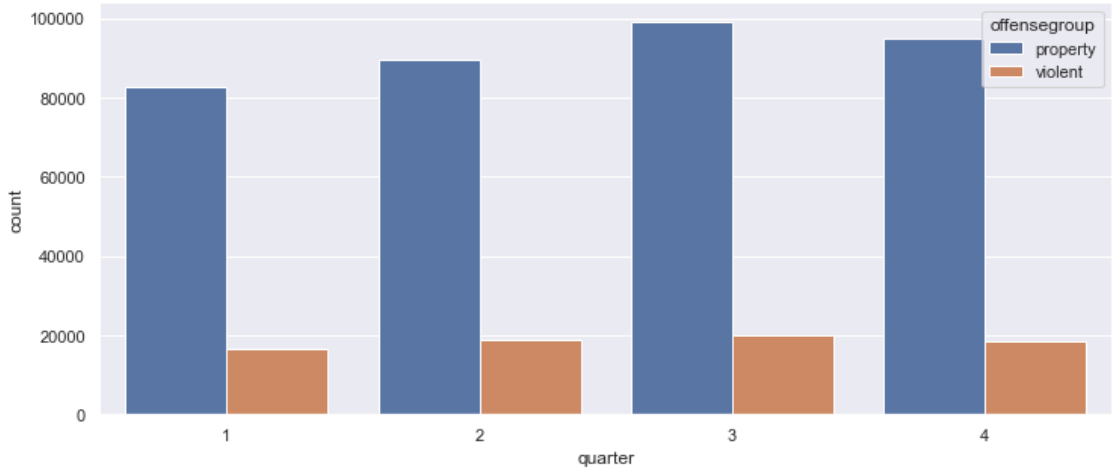


*Conclusion: crime was high from July to October while was low from February to April.

[110]:
```
combine('weekofyear','offensegroup')
# crimes by weekofyear and offensegroup
```

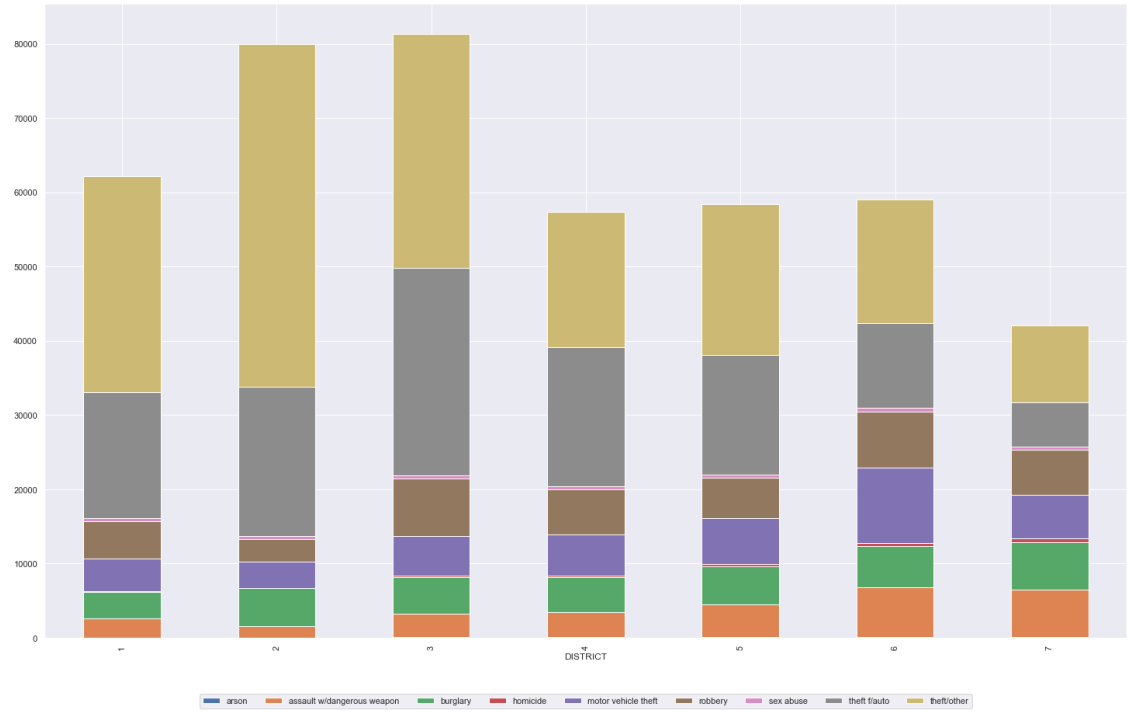


[111]:
```
combine('quarter','offensegroup')
# crimes by quater and offensegroup
```



[112]:
```
temp = pd.crosstab(crime_filtered_newfeature.DISTRICT,␣
↪crime_filtered_newfeature.OFFENSE)
temp.plot(kind='bar', stacked=True, figsize=(25, 15))
plt.legend(loc=10, bbox_to_anchor=(.5, -.1), ncol=10)
```

[112]:

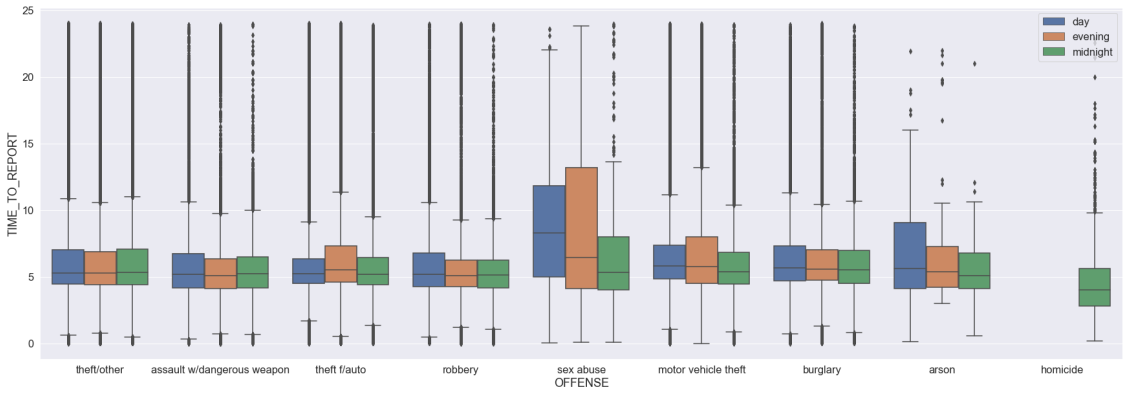


*Conclusion: some severe crimes like arson, assault w/dangerous weapon, sex abuse and homicide had greater proportions in district 6 and 7.

Boxplots.

[113]:
```
plt.figure(figsize=(30,10))
sns.set(font_scale=1.5)
plt_test = crime_filtered_newfeature[crime_filtered_newfeature.TIME_TO_REPORT <␣
 ↪86400][crime_filtered_newfeature.TIME_TO_REPORT > 0]
sns.boxplot(x=plt_test.OFFENSE, y=plt_test.TIME_TO_REPORT/3600.0, hue=plt_test.
 ↪SHIFT)
plt.legend(loc='upper right')
# TIME_TO_REPORT and OFFENSE with different SHIFT
```

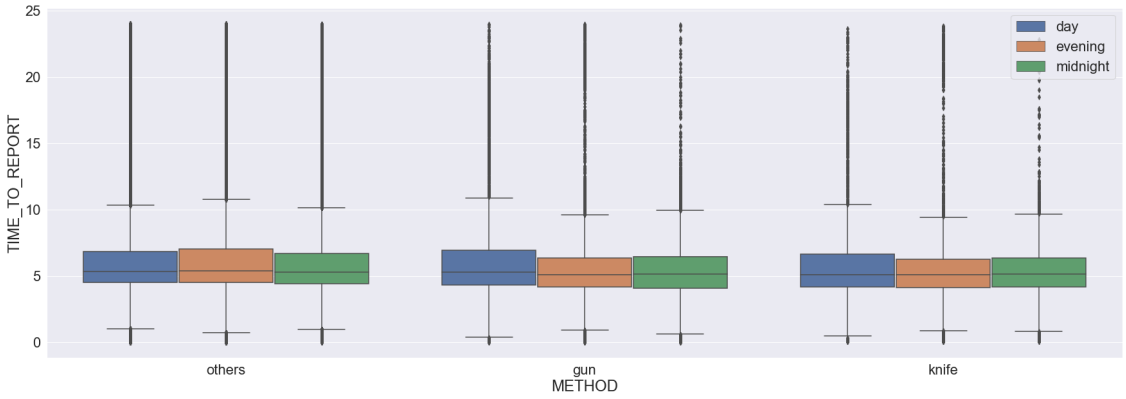
[113]:



*Conclusion: homicide happened only in midnight.
Sex abuse took significantly longer to report. This may be because the victim was hesitant to let people know about the abuse.

[114]:
```
plt.figure(figsize=(30,10))
sns.set(font_scale=2)
plt_test = crime_filtered_newfeature[crime_filtered_newfeature.TIME_TO_REPORT <␣
↪86400][crime_filtered_newfeature.TIME_TO_REPORT > 0]
sns.boxplot(x=plt_test.METHOD, y=plt_test.TIME_TO_REPORT/3600.0, hue=plt_test.
↪SHIFT)
plt.legend(loc='upper right')
# TIME_TO_REPORT and METHOD with different SHIFT
```

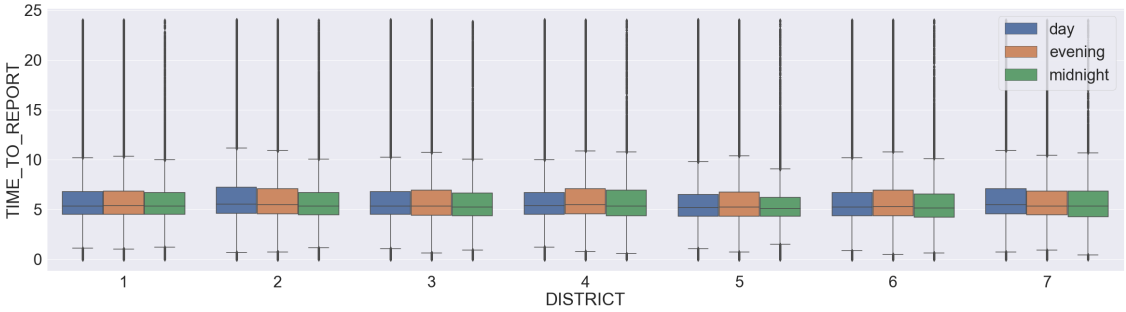
[114]:



[115]:
```
plt.figure(figsize=(40,10))
sns.set(font_scale=3)
plt_test = crime_filtered_newfeature[crime_filtered_newfeature.TIME_TO_REPORT <␣
↪86400][crime_filtered_newfeature.TIME_TO_REPORT > 0]
```

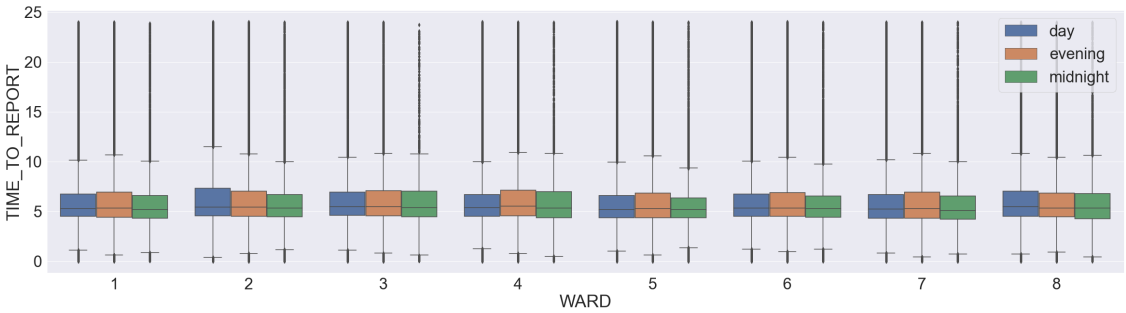
```
sns.boxplot(x=plt_test.DISTRICT, y=plt_test.TIME_TO_REPORT/3600.0, hue=plt_test.
 ↪SHIFT)
plt.legend(loc='upper right')
# TIME_TO_REPORT and DISTRICT with different SHIFT
```

[115]:



[116]:
```
plt.figure(figsize=(40,10))
sns.set(font_scale=3)
plt_test = crime_filtered_newfeature[crime_filtered_newfeature.TIME_TO_REPORT <␣
 ↪86400][crime_filtered_newfeature.TIME_TO_REPORT > 0]
sns.boxplot(x=plt_test.WARD, y=plt_test.TIME_TO_REPORT/3600.0, hue=plt_test.
 ↪SHIFT)
plt.legend(loc='upper right')
# TIME_TO_REPORT and WARD with different SHIFT
```

[116]:



3.0.2 2-2 Correlation Analysis-1(Task 2-2)

@task:Please also analyze the correlation between different features, such as the correlation between 'shift' and 'offense', and the correlation between'offense'and'method'. By dividing the time series into several parts, judge whether the correlations change are evident with the time changes.

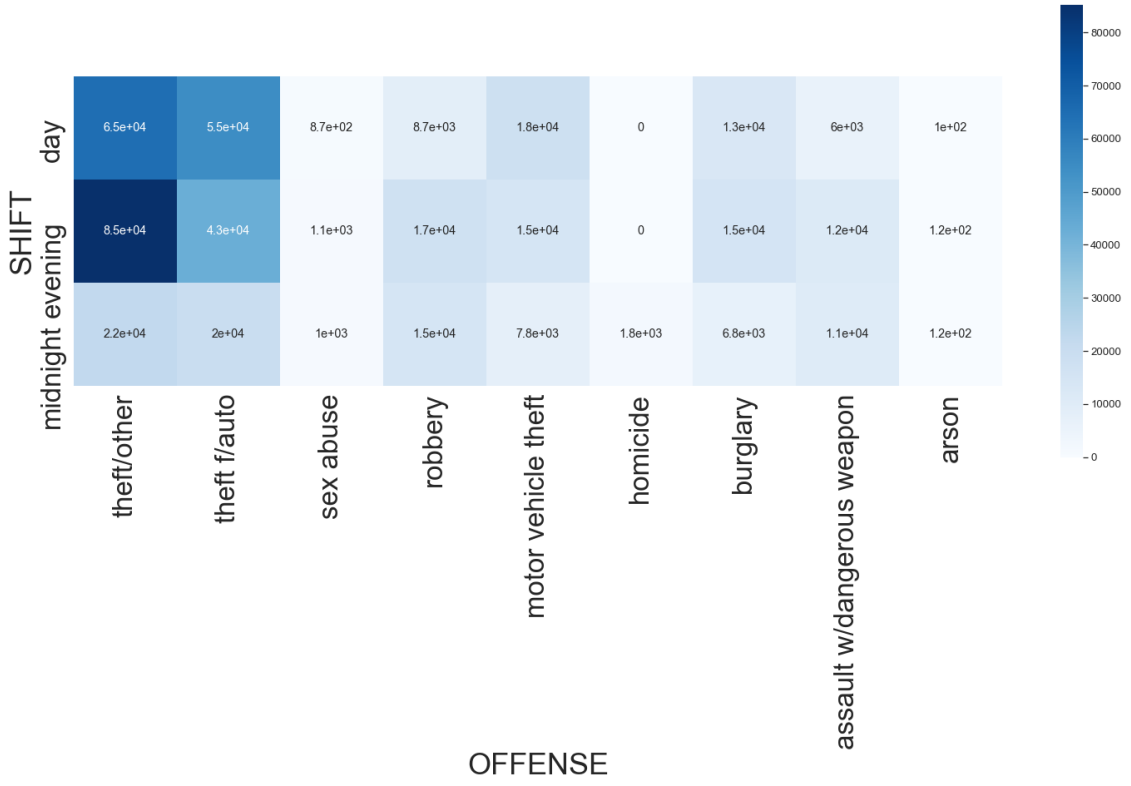
Correlation between character features. ('shift' and 'offense', 'offense' and 'method', so on.)

[117]:
```
cross1=pd.crosstab(crime_filtered['SHIFT'], crime_filtered['OFFENSE'])
cross2=pd.crosstab(crime_filtered['METHOD'], crime_filtered['OFFENSE'])
```

[118]:
```
# define a function to visualize the correlation between features
def correlation(cross,year,color):
    f, ax=plt.subplots(figsize=(25, 10))
    sns.set(font_scale=1.2)
    sns.heatmap(cross, cmap = color,annot = True,square=True)
    ax.set_ylim([3,0])
    ax.set_xlim([9,0])
    if year != '0':
        plt.title('YEAR='+year,fontsize=20)
    plt.show()
```

Correlation between 'shift' and 'offense'.

[119]:
```
correlation(cross1,'0','Blues')
```

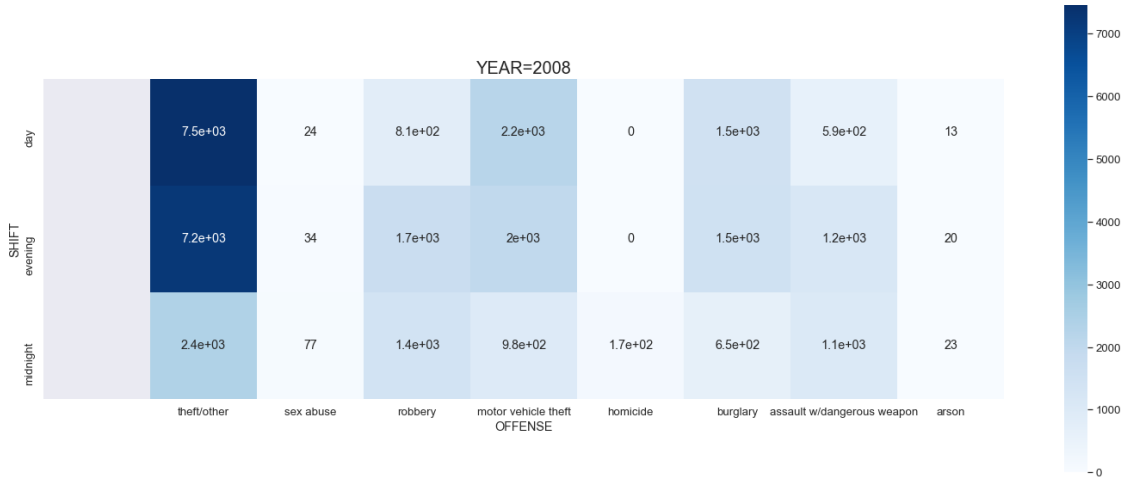


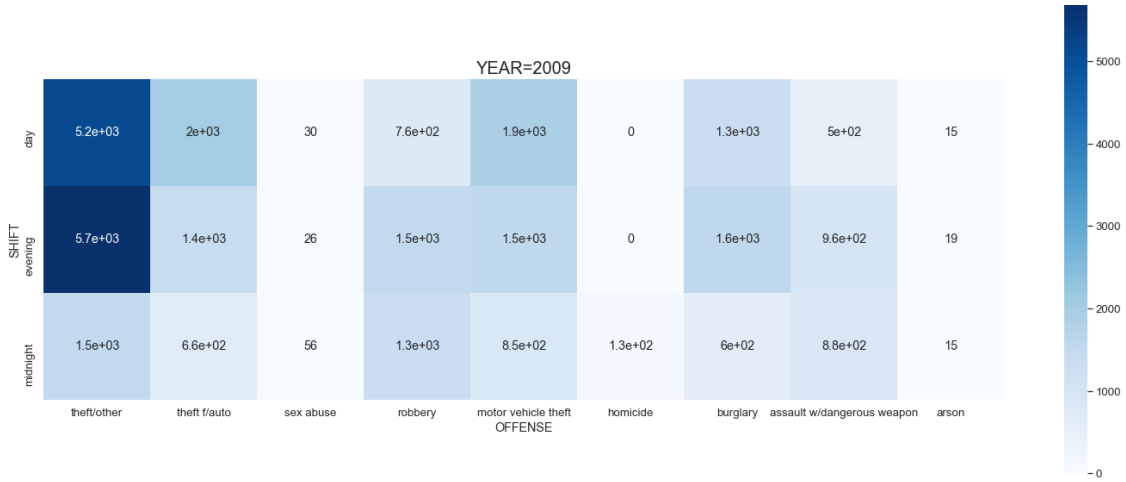
*Conclusion: Theft/other were mainly in the evening while theft f/auto were mainly in the day, and daytime theft were relatively rare. Robbery were less in the day. Motor verhicle theft were mainly in the day. Homicide were only in the midnight. Burglary were less in the midnight. Assault w/Dangerous weapon were less in the day. Sex abuse and arson have little to do with shift.

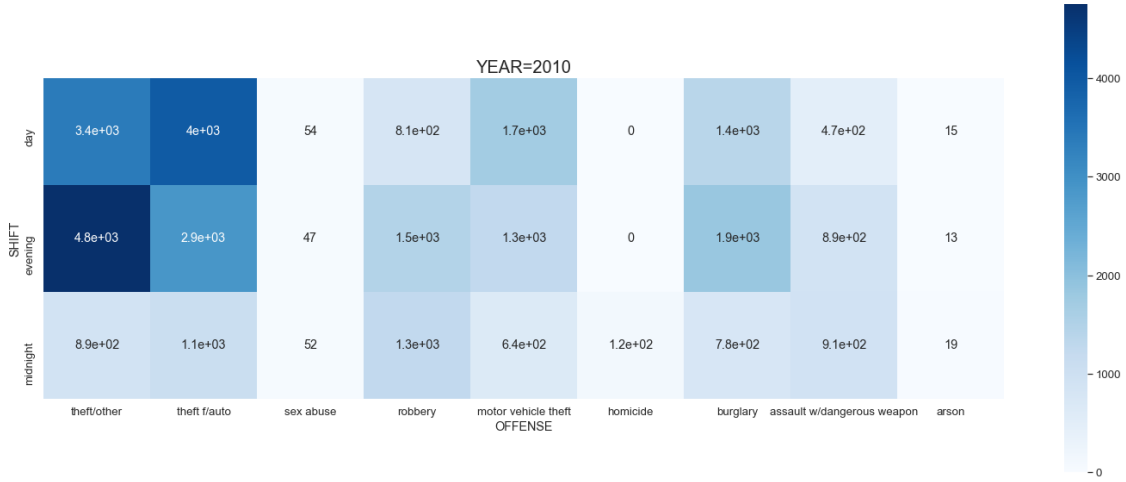
Correlation between ‘shift’ and ‘offense’ with the time changes.

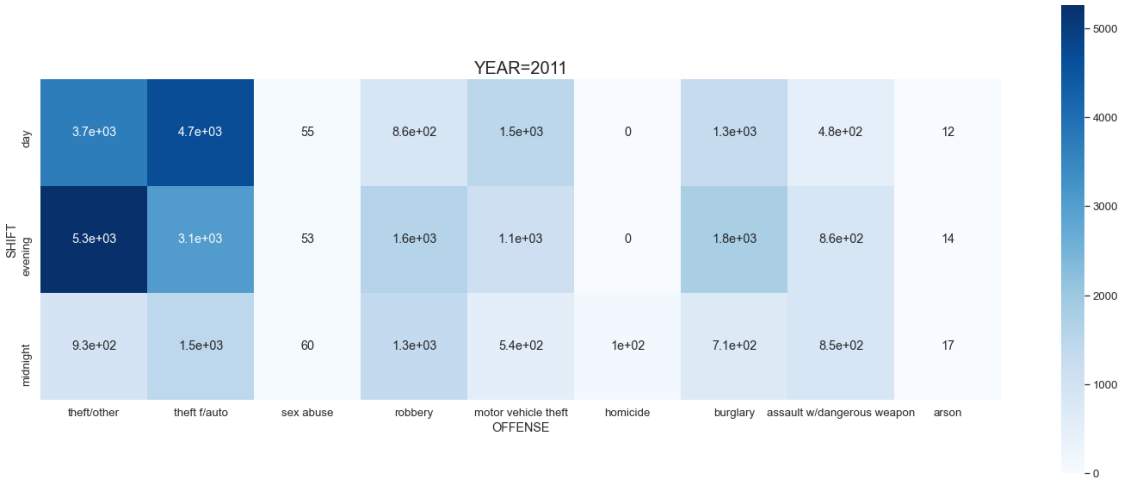
[120]:
```
# different years
cross1_08=pd.crosstab(crime_filtered_08['SHIFT'], crime_filtered_08['OFFENSE'])
cross1_09=pd.crosstab(crime_filtered_09['SHIFT'], crime_filtered_09['OFFENSE'])
cross1_10=pd.crosstab(crime_filtered_10['SHIFT'], crime_filtered_10['OFFENSE'])
cross1_11=pd.crosstab(crime_filtered_11['SHIFT'], crime_filtered_11['OFFENSE'])
cross1_12=pd.crosstab(crime_filtered_12['SHIFT'], crime_filtered_12['OFFENSE'])
cross1_13=pd.crosstab(crime_filtered_13['SHIFT'], crime_filtered_13['OFFENSE'])
cross1_14=pd.crosstab(crime_filtered_14['SHIFT'], crime_filtered_14['OFFENSE'])
cross1_15=pd.crosstab(crime_filtered_15['SHIFT'], crime_filtered_15['OFFENSE'])
cross1_16=pd.crosstab(crime_filtered_16['SHIFT'], crime_filtered_16['OFFENSE'])
cross1_17=pd.crosstab(crime_filtered_17['SHIFT'], crime_filtered_17['OFFENSE'])
cross1_18=pd.crosstab(crime_filtered_18['SHIFT'], crime_filtered_18['OFFENSE'])
cross1_19=pd.crosstab(crime_filtered_19['SHIFT'], crime_filtered_19['OFFENSE'])
cross1_20=pd.crosstab(crime_filtered_20['SHIFT'], crime_filtered_20['OFFENSE'])
cross1_21=pd.crosstab(crime_filtered_21['SHIFT'], crime_filtered_21['OFFENSE'])
```

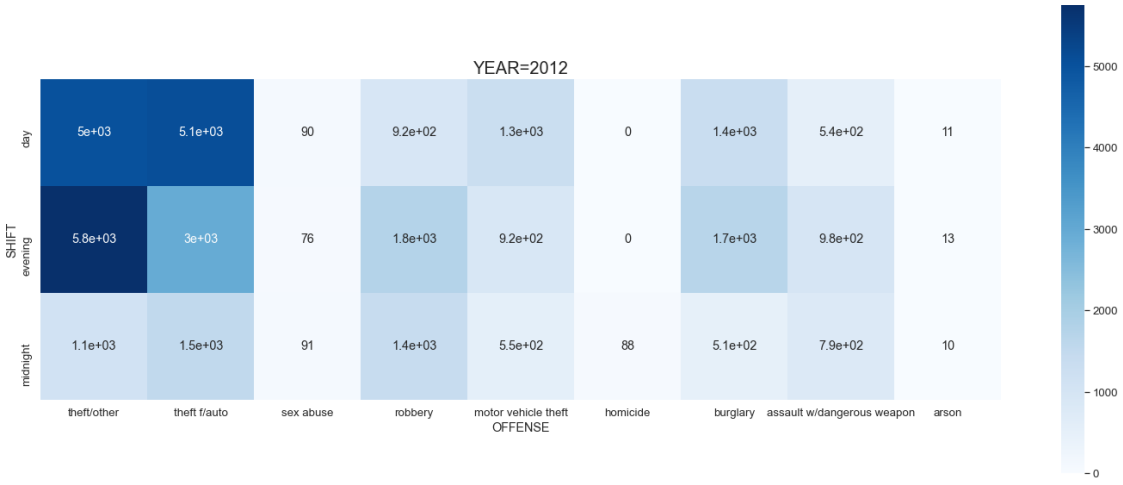
[121]:
```
correlation(cross1_08,'2008','Blues')
correlation(cross1_09,'2009','Blues')
correlation(cross1_10,'2010','Blues')
correlation(cross1_11,'2011','Blues')
correlation(cross1_12,'2012','Blues')
correlation(cross1_13,'2013','Blues')
correlation(cross1_14,'2014','Blues')
correlation(cross1_15,'2015','Blues')
correlation(cross1_16,'2016','Blues')
correlation(cross1_17,'2017','Blues')
correlation(cross1_18,'2018','Blues')
correlation(cross1_19,'2019','Blues')
correlation(cross1_20,'2020','Blues')
correlation(cross1_21,'2021','Blues')
```

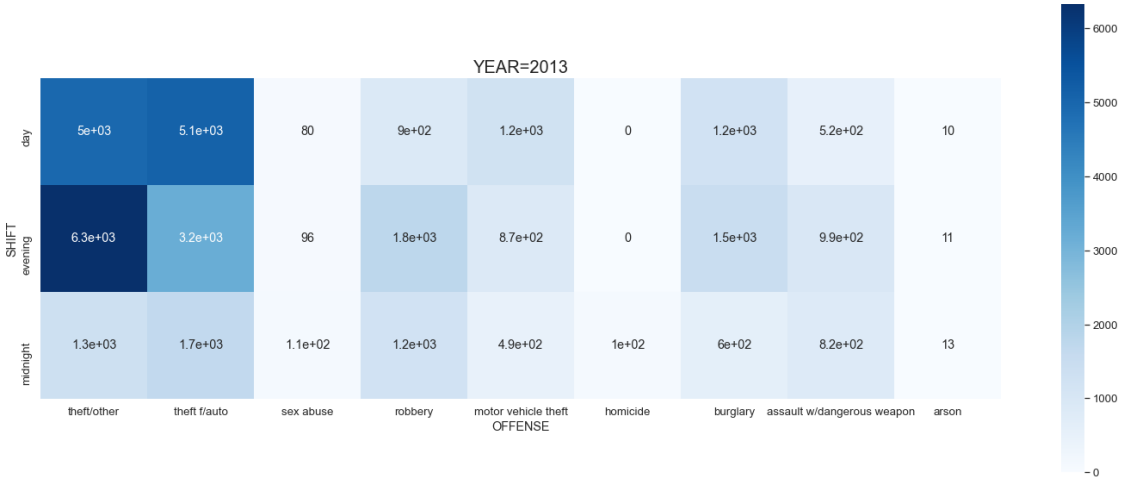


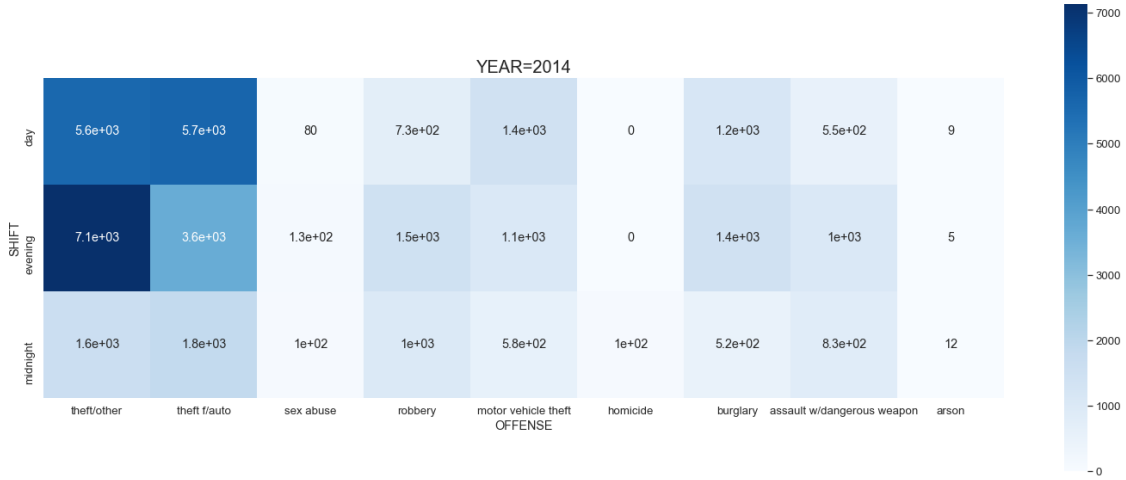

YEAR=2009
SHIFT
day
evening
midnight
5.2e+03
2e+03
30
7.6e+02
1.9e+03
0
1.3e+03
5e+02
15
5.7e+03
1.4e+03
26
1.5e+03
1.5e+03
0
1.6e+03
9.6e+02
19
1.5e+03
6.6e+02
56
1.3e+03
8.5e+02
1.3e+02
6e+02
8.8e+02
15
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
5000
4000
3000
2000
1000
0

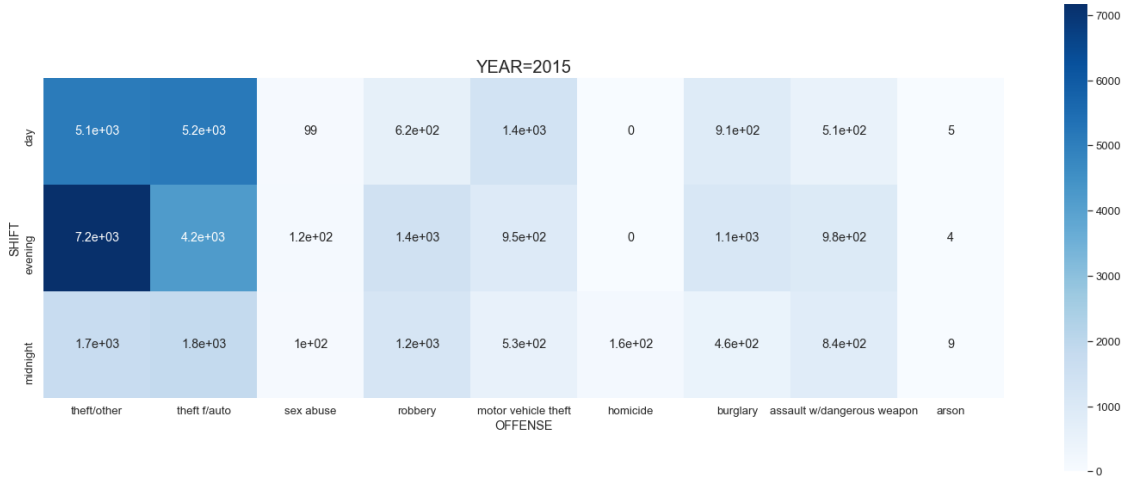

YEAR=2010
SHIFT
day
evening
midnight
3.4e+03
4e+03
54
8.1e+02
1.7e+03
0
1.4e+03
4.7e+02
15
4.8e+03
2.9e+03
47
1.5e+03
1.3e+03
0
1.9e+03
8.9e+02
13
8.9e+02
1.1e+03
52
1.3e+03
6.4e+02
1.2e+02
7.8e+02
9.1e+02
19
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
4000
3000
2000
1000
0

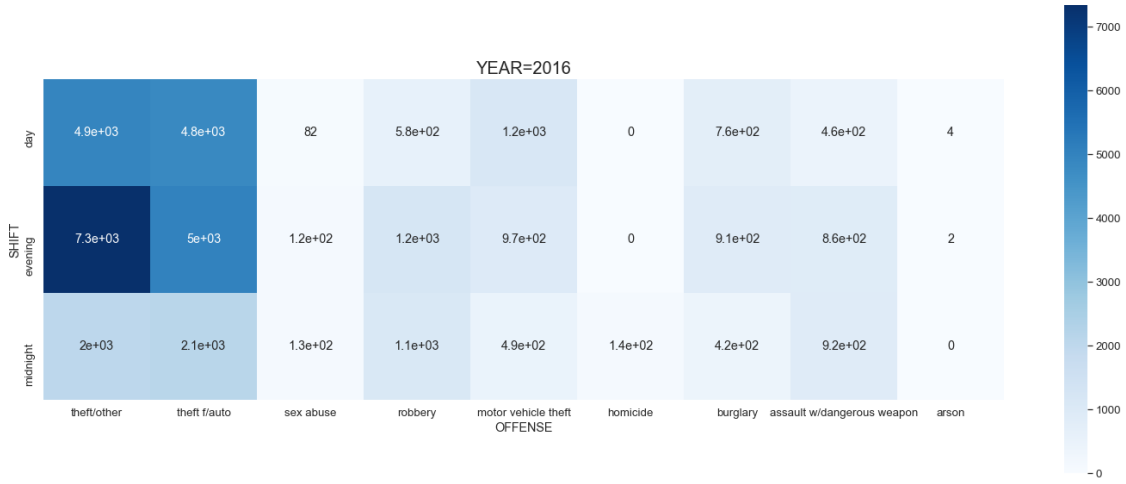

YEAR=2011
SHIFT
day
evening
midnight
3.7e+03
4.7e+03
55
8.6e+02
1.5e+03
0
1.3e+03
4.8e+02
12
5.3e+03
3.1e+03
53
1.6e+03
1.1e+03
0
1.8e+03
8.6e+02
14
9.3e+02
1.5e+03
60
1.3e+03
5.4e+02
1e+02
7.1e+02
8.5e+02
17
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
5000
4000
3000
2000
1000
0


YEAR=2012
SHIFT
day
evening
midnight
5e+03
5.1e+03
90
9.2e+02
1.3e+03
0
1.4e+03
5.4e+02
11
5.8e+03
3e+03
76
1.8e+03
9.2e+02
0
1.7e+03
9.8e+02
13
1.1e+03
1.5e+03
91
1.4e+03
5.5e+02
88
5.1e+02
7.9e+02
10
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
5000
4000
3000
2000
1000
0

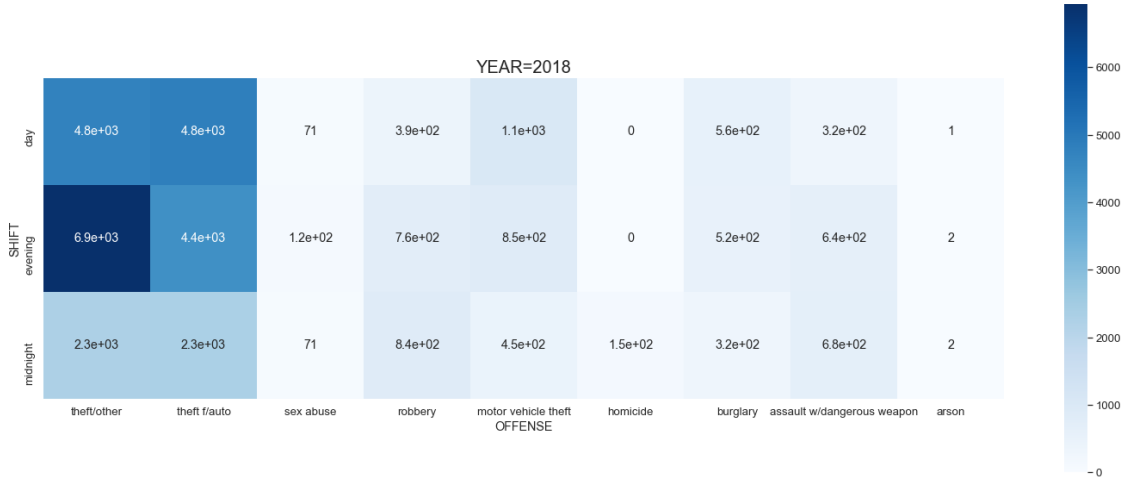

YEAR=2013
SHIFT
day
evening
midnight
5e+03
5.1e+03
80
9e+02
1.2e+03
0
1.2e+03
5.2e+02
10
6.3e+03
3.2e+03
96
1.8e+03
8.7e+02
0
1.5e+03
9.9e+02
11
1.3e+03
1.7e+03
1.1e+02
1.2e+03
4.9e+02
1e+02
6e+02
8.2e+02
13
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
6000
5000
4000
3000
2000
1000
0

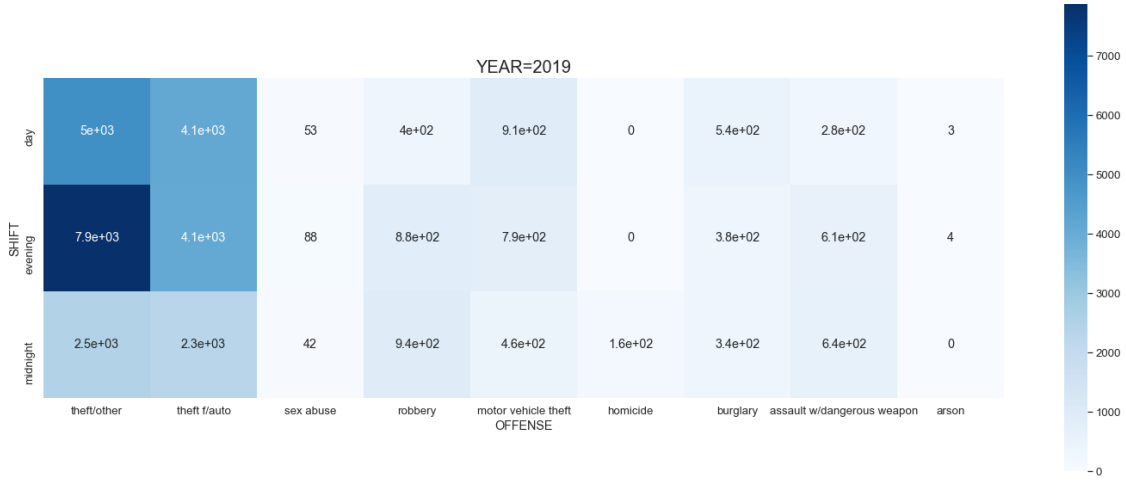

YEAR=2014
SHIFT
day
evening
midnight
5.6e+03
5.7e+03
80
7.3e+02
1.4e+03
0
1.2e+03
5.5e+02
9
7.1e+03
3.6e+03
1.3e+02
1.5e+03
1.1e+03
0
1.4e+03
1e+03
5
1.6e+03
1.8e+03
1e+02
1e+03
5.8e+02
1e+02
5.2e+02
8.3e+02
12
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
7000
6000
5000
4000
3000
2000
1000
0


YEAR=2015
SHIFT
day
evening
midnight
5.1e+03
5.2e+03
99
6.2e+02
1.4e+03
0
9.1e+02
5.1e+02
5
7.2e+03
4.2e+03
1.2e+02
1.4e+03
9.5e+02
0
1.1e+03
9.8e+02
4
1.7e+03
1.8e+03
1e+02
1.2e+03
5.3e+02
1.6e+02
4.6e+02
8.4e+02
9
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
7000
6000
5000
4000
3000
2000
1000
0

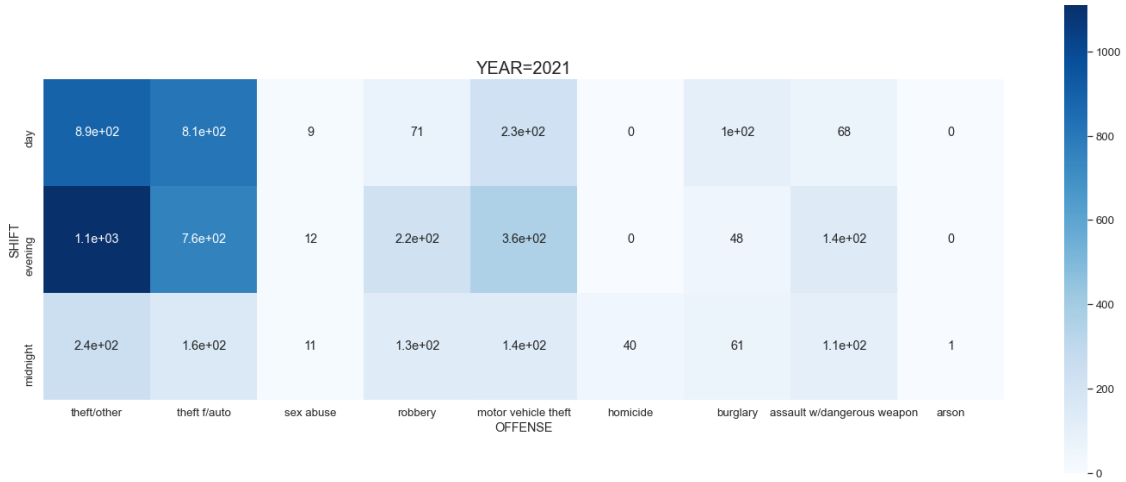

YEAR=2016
SHIFT
day
evening
midnight
4.9e+03
4.8e+03
82
5.8e+02
1.2e+03
0
7.6e+02
4.6e+02
4
7.3e+03
5e+03
1.2e+02
1.2e+03
9.7e+02
0
9.1e+02
8.6e+02
2
2e+03
2.1e+03
1.3e+02
1.1e+03
4.9e+02
1.4e+02
4.2e+02
9.2e+02
0
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
7000
6000
5000
4000
3000
2000
1000
0


YEAR=2017
SHIFT
day
evening
midnight
4.8e+03
4.6e+03
82
4.6e+02
1.1e+03
0
6.3e+02
3.8e+02
2
7.1e+03
3.7e+03
1.3e+02
8.3e+02
7.6e+02
0
5.3e+02
7.4e+02
2
2.4e+03
1.9e+03
75
8.5e+02
4.7e+02
1.1e+02
3.4e+02
7.2e+02
1
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
7000
6000
5000
4000
3000
2000
1000
0


YEAR=2018
SHIFT
day
evening
midnight
4.8e+03
4.8e+03
71
3.9e+02
1.1e+03
0
5.6e+02
3.2e+02
1
6.9e+03
4.4e+03
1.2e+02
7.6e+02
8.5e+02
0
5.2e+02
6.4e+02
2
2.3e+03
2.3e+03
71
8.4e+02
4.5e+02
1.5e+02
3.2e+02
6.8e+02
2
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
6000
5000
4000
3000
2000
1000
0


YEAR=2019
SHIFT
day
evening
midnight
5e+03
4.1e+03
53
4e+02
9.1e+02
0
5.4e+02
2.8e+02
3
7.9e+03
4.1e+03
88
8.8e+02
7.9e+02
0
3.8e+02
6.1e+02
4
2.5e+03
2.3e+03
42
9.4e+02
4.6e+02
1.6e+02
3.4e+02
6.4e+02
0
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
7000
6000
5000
4000
3000
2000
1000
0

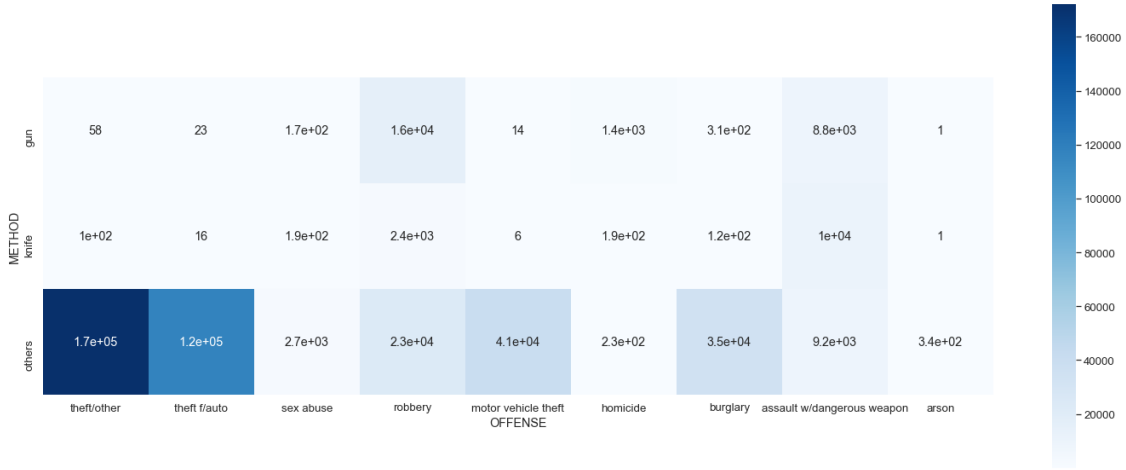

YEAR=2020
SHIFT
day
evening
midnight
3.9e+03
3.6e+03
63
4e+02
1.1e+03
0
6.2e+02
3.2e+02
4
5.5e+03
3.5e+03
64
7.7e+02
1.5e+03
0
3.5e+02
6.6e+02
7
1.5e+03
1.1e+03
42
8.2e+02
6.5e+02
2e+02
4.7e+02
6.5e+02
2
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
5000
4000
3000
2000
1000
0



*Conclusion: No theft f/auto in 2008. Besides that, changes are not evident.

Correlation between 'offense' and 'method'.

[122]:
```
correlation(cross2,'0','Blues')
```



*Conclusion: Theft, arson and burglary have strong relations with method, very few theft, arson and burglary crimes involved gun and knife. Assault w/Dangerous weapon have little difference in method.

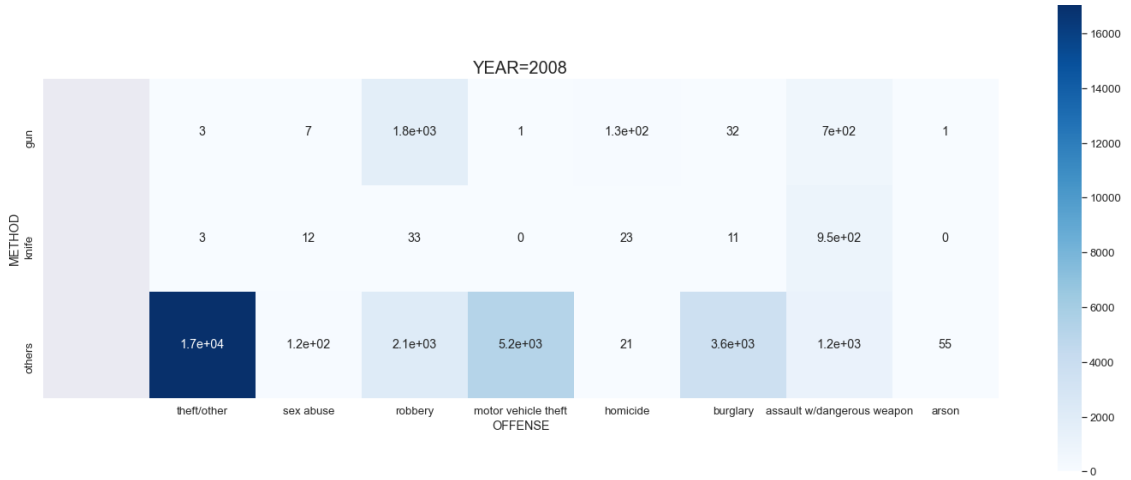
————————————-Approximately————————————

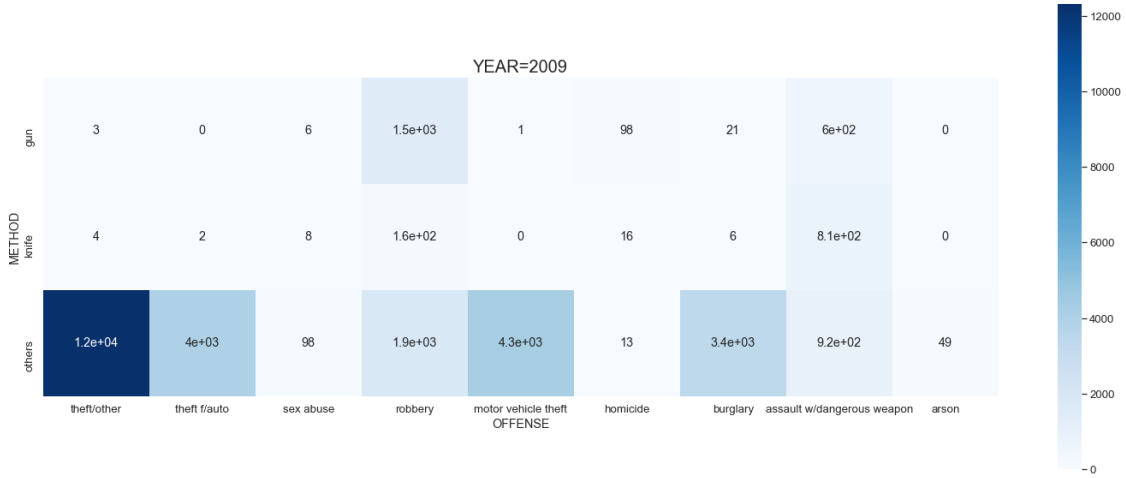
sex abuse: 5%gun——5%knife——90%others
robbery: 39%gun——6%knife——55%others
homicide: 77%gun——10%knife——13%others

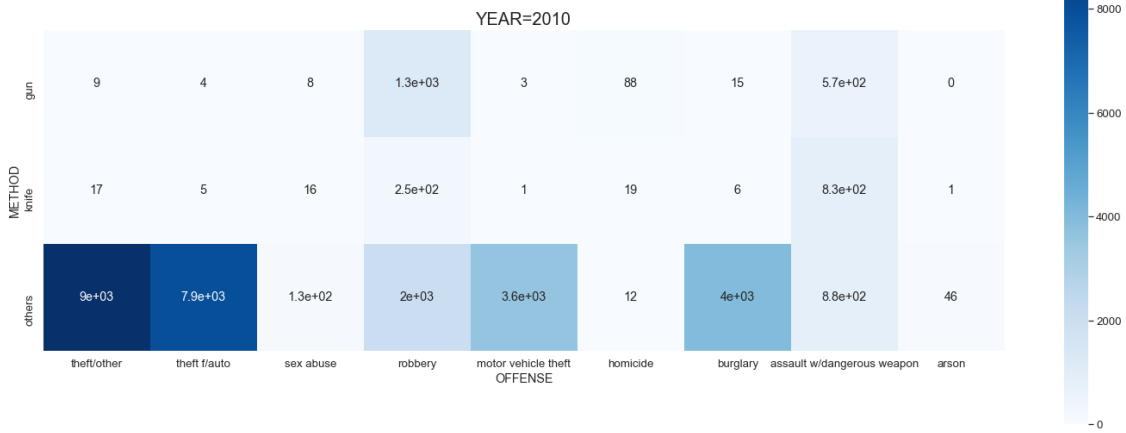
Correlation between 'offense' and 'method' with the time changes.

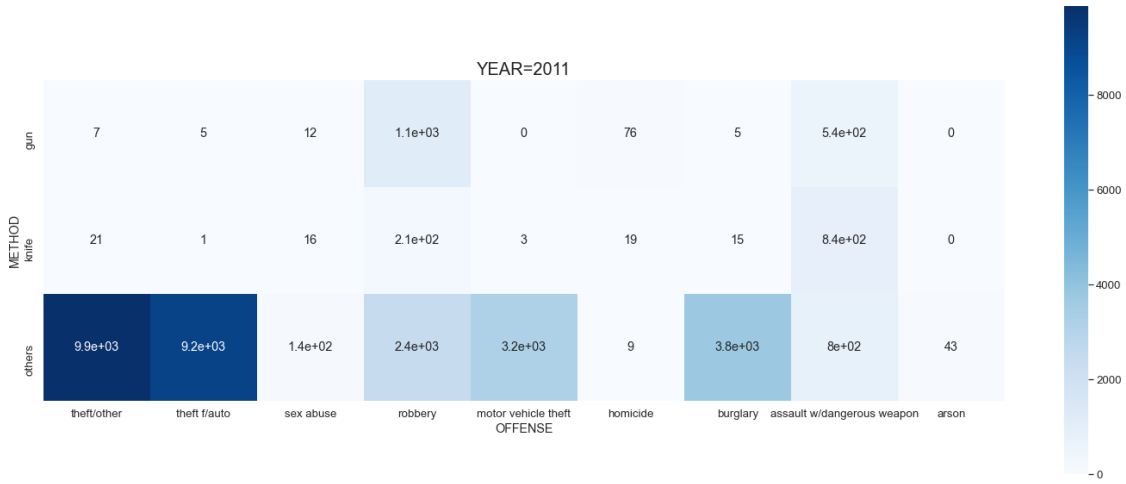
[123]:
```
cross2_08=pd.crosstab(crime_filtered_08['METHOD'], crime_filtered_08['OFFENSE'])
cross2_09=pd.crosstab(crime_filtered_09['METHOD'], crime_filtered_09['OFFENSE'])
cross2_10=pd.crosstab(crime_filtered_10['METHOD'], crime_filtered_10['OFFENSE'])
cross2_11=pd.crosstab(crime_filtered_11['METHOD'], crime_filtered_11['OFFENSE'])
cross2_12=pd.crosstab(crime_filtered_12['METHOD'], crime_filtered_12['OFFENSE'])
cross2_13=pd.crosstab(crime_filtered_13['METHOD'], crime_filtered_13['OFFENSE'])
cross2_14=pd.crosstab(crime_filtered_14['METHOD'], crime_filtered_14['OFFENSE'])
cross2_15=pd.crosstab(crime_filtered_15['METHOD'], crime_filtered_15['OFFENSE'])
cross2_16=pd.crosstab(crime_filtered_16['METHOD'], crime_filtered_16['OFFENSE'])
cross2_17=pd.crosstab(crime_filtered_17['METHOD'], crime_filtered_17['OFFENSE'])
cross2_18=pd.crosstab(crime_filtered_18['METHOD'], crime_filtered_18['OFFENSE'])
cross2_19=pd.crosstab(crime_filtered_19['METHOD'], crime_filtered_19['OFFENSE'])
cross2_20=pd.crosstab(crime_filtered_20['METHOD'], crime_filtered_20['OFFENSE'])
cross2_21=pd.crosstab(crime_filtered_21['METHOD'], crime_filtered_21['OFFENSE'])
```

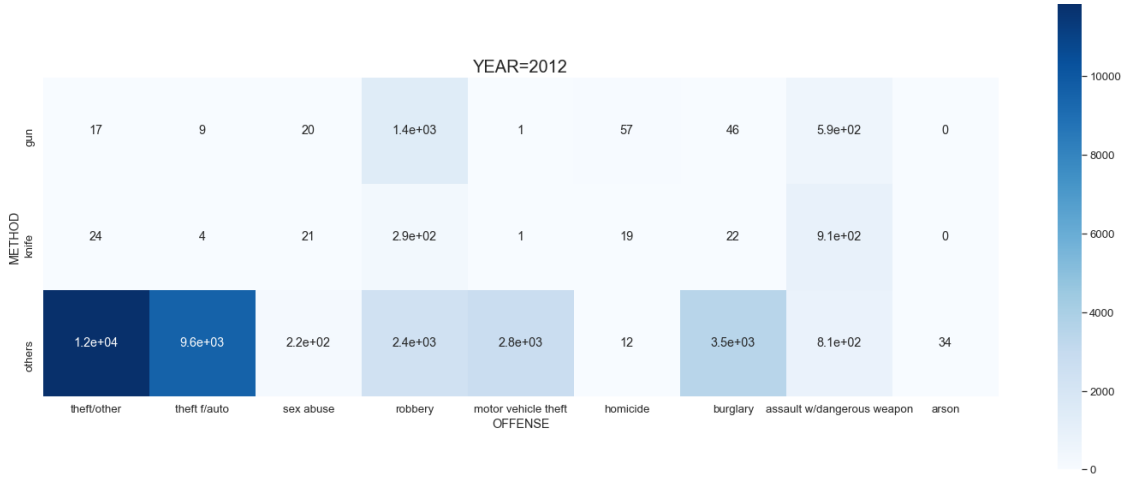
[124]:
```
correlation(cross2_08,'2008','Blues')
correlation(cross2_09,'2009','Blues')
correlation(cross2_10,'2010','Blues')
correlation(cross2_11,'2011','Blues')
correlation(cross2_12,'2012','Blues')
correlation(cross2_13,'2013','Blues')
correlation(cross2_14,'2014','Blues')
correlation(cross2_15,'2015','Blues')
correlation(cross2_16,'2016','Blues')
correlation(cross2_17,'2017','Blues')
correlation(cross2_18,'2018','Blues')
correlation(cross2_19,'2019','Blues')
correlation(cross2_20,'2020','Blues')
correlation(cross2_21,'2021','Blues')
```

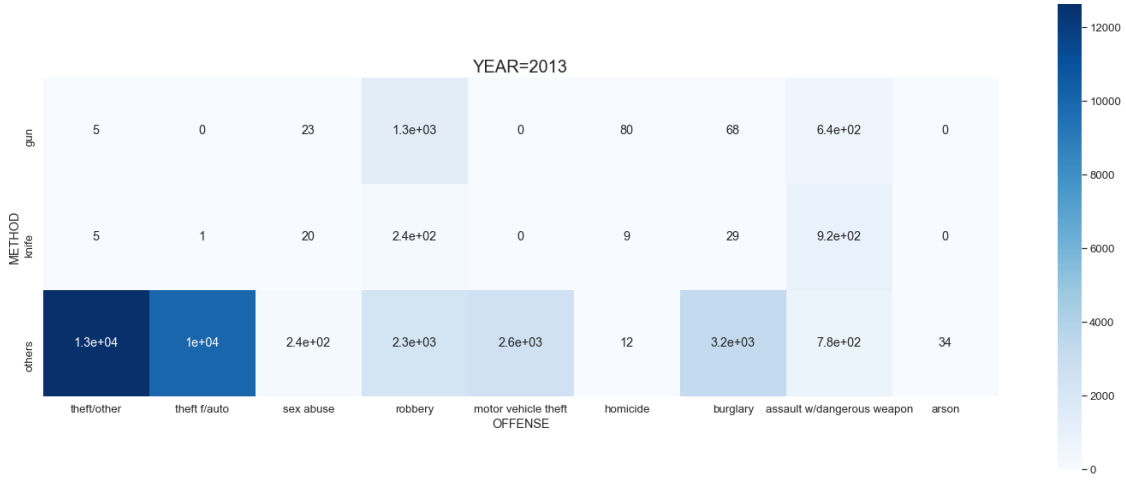


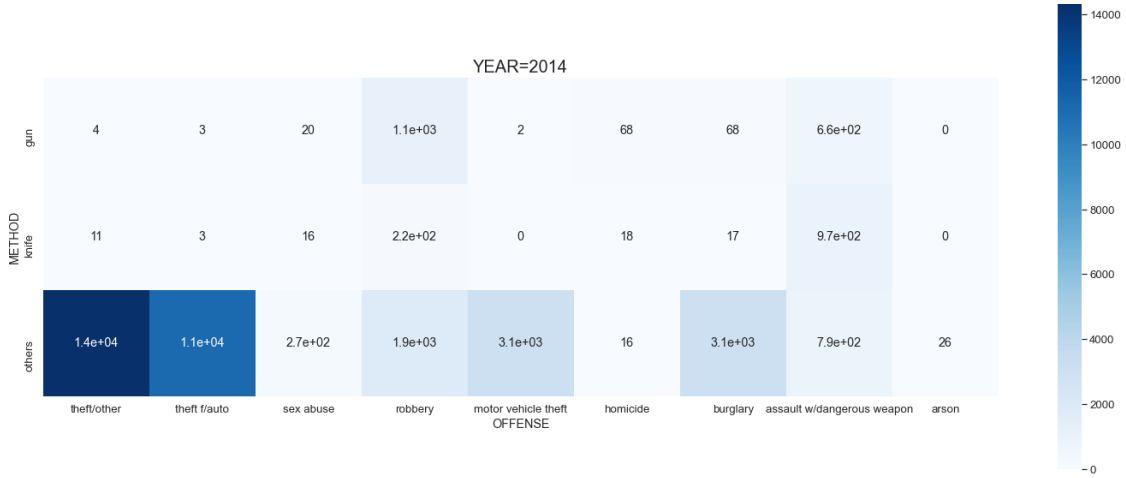

YEAR=2009
METHOD
gun
knife
others
3
0
6
1.5e+03
1
98
21
6e+02
0
4
2
8
1.6e+02
0
16
6
8.1e+02
0
1.2e+04
4e+03
98
1.9e+03
4.3e+03
13
3.4e+03
9.2e+02
49
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
12000
10000
8000
6000
4000
2000
0

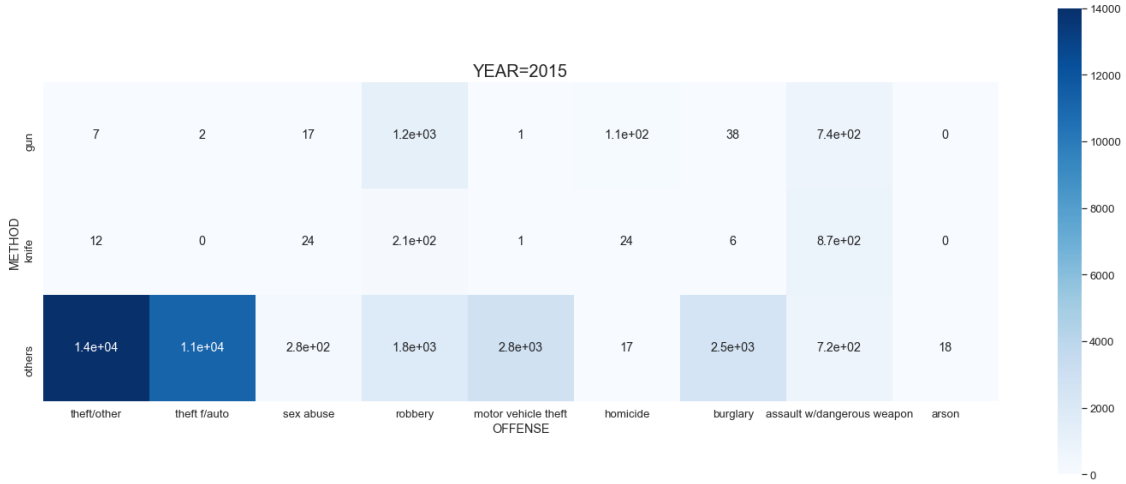

YEAR=2010
METHOD
gun
knife
others
9
4
8
1.3e+03
3
88
15
5.7e+02
0
17
5
16
2.5e+02
1
19
6
8.3e+02
1
9e+03
7.9e+03
1.3e+02
2e+03
3.6e+03
12
4e+03
8.8e+02
46
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
8000
6000
4000
2000
0

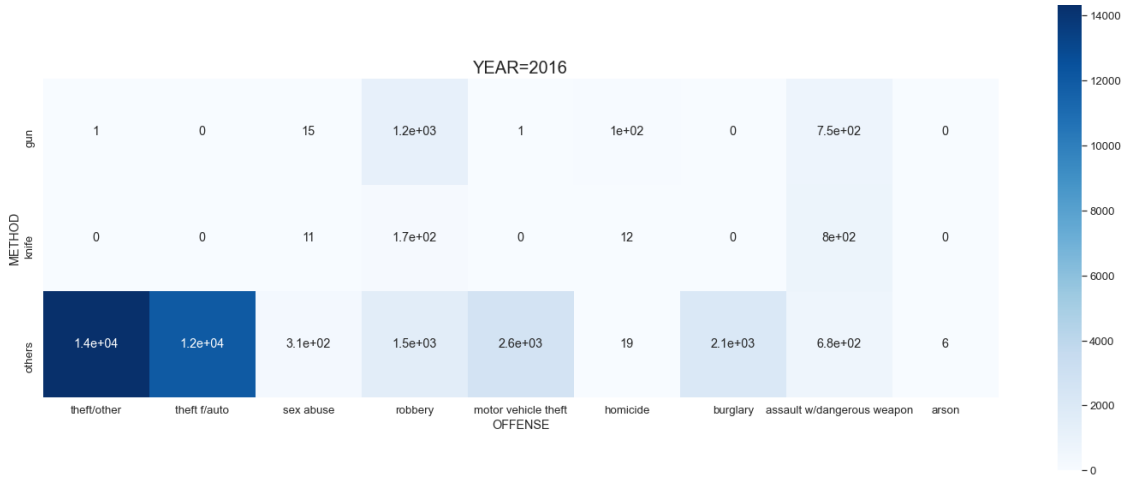

YEAR=2011
METHOD
gun
knife
others
7
5
12
1.1e+03
0
76
5
5.4e+02
0
21
1
16
2.1e+02
3
19
15
8.4e+02
0
9.9e+03
9.2e+03
1.4e+02
2.4e+03
3.2e+03
9
3.8e+03
8e+02
43
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
8000
6000
4000
2000
0


YEAR=2012
METHOD
gun
knife
others
17
9
20
1.4e+03
1
57
46
5.9e+02
0
24
4
21
2.9e+02
1
19
22
9.1e+02
0
1.2e+04
9.6e+03
2.2e+02
2.4e+03
2.8e+03
12
3.5e+03
8.1e+02
34
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
10000
8000
6000
4000
2000
0

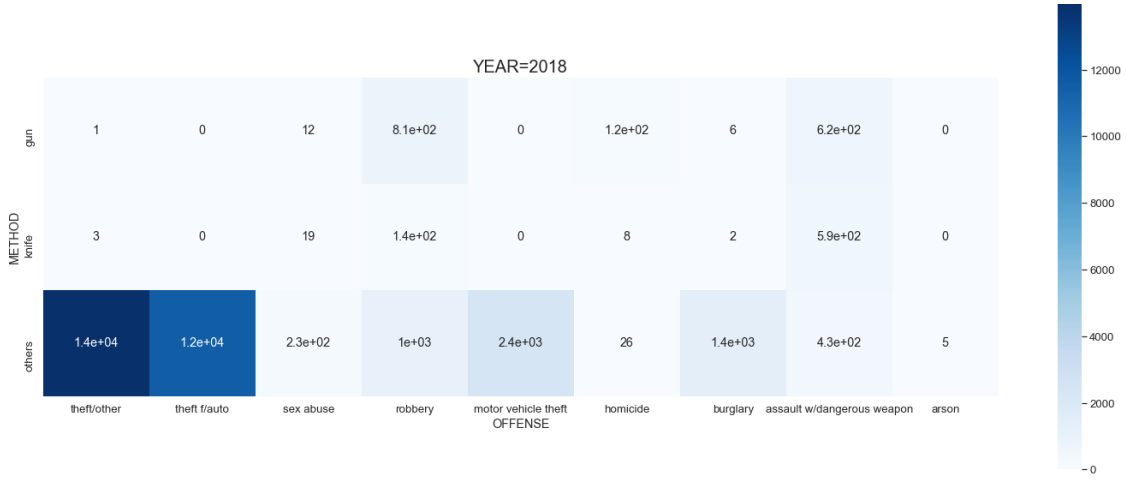

YEAR=2013
METHOD
gun
knife
others
5
0
23
1.3e+03
0
80
68
6.4e+02
0
5
1
20
2.4e+02
0
9
29
9.2e+02
0
1.3e+04
1e+04
2.4e+02
2.3e+03
2.6e+03
12
3.2e+03
7.8e+02
34
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
12000
10000
8000
6000
4000
2000
0

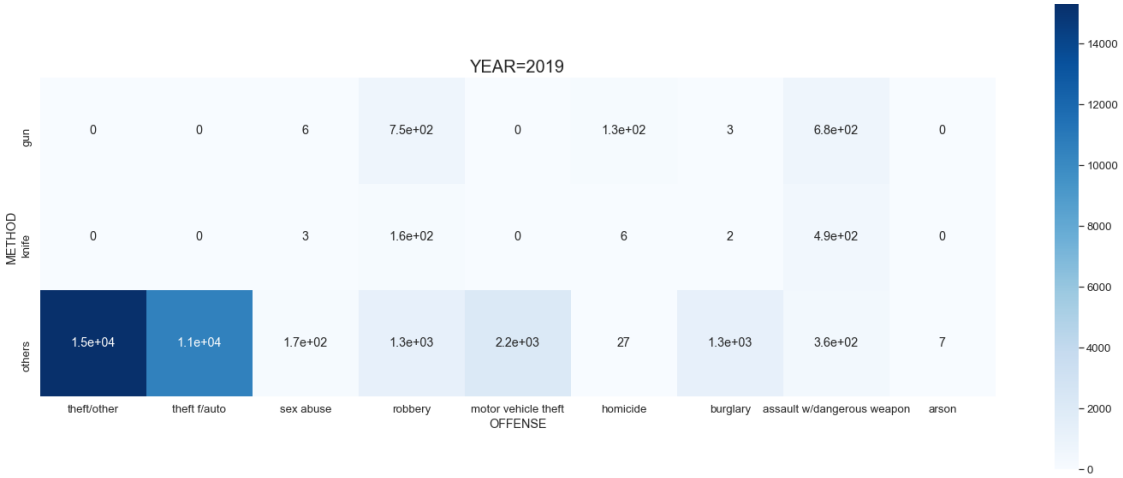

YEAR=2014
METHOD
gun
knife
others
4
3
20
1.1e+03
2
68
68
6.6e+02
0
11
3
16
2.2e+02
0
18
17
9.7e+02
0
1.4e+04
1.1e+04
2.7e+02
1.9e+03
3.1e+03
16
3.1e+03
7.9e+02
26
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
14000
12000
10000
8000
6000
4000
2000
0

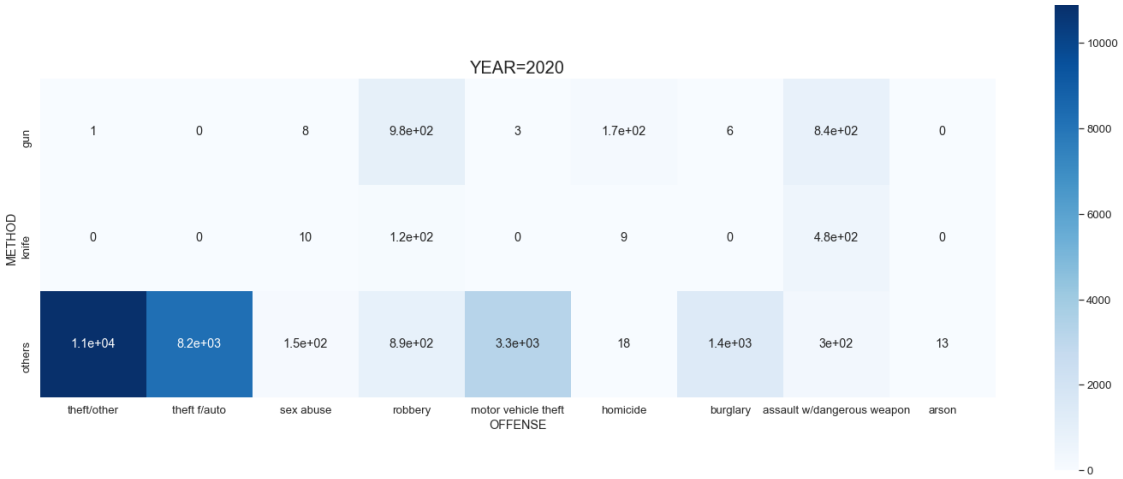

YEAR=2015
METHOD
gun
knife
others
7
2
17
1.2e+03
1
1.1e+02
38
7.4e+02
0
12
0
24
2.1e+02
1
24
6
8.7e+02
0
1.4e+04
1.1e+04
2.8e+02
1.8e+03
2.8e+03
17
2.5e+03
7.2e+02
18
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
14000
12000
10000
8000
6000
4000
2000
0

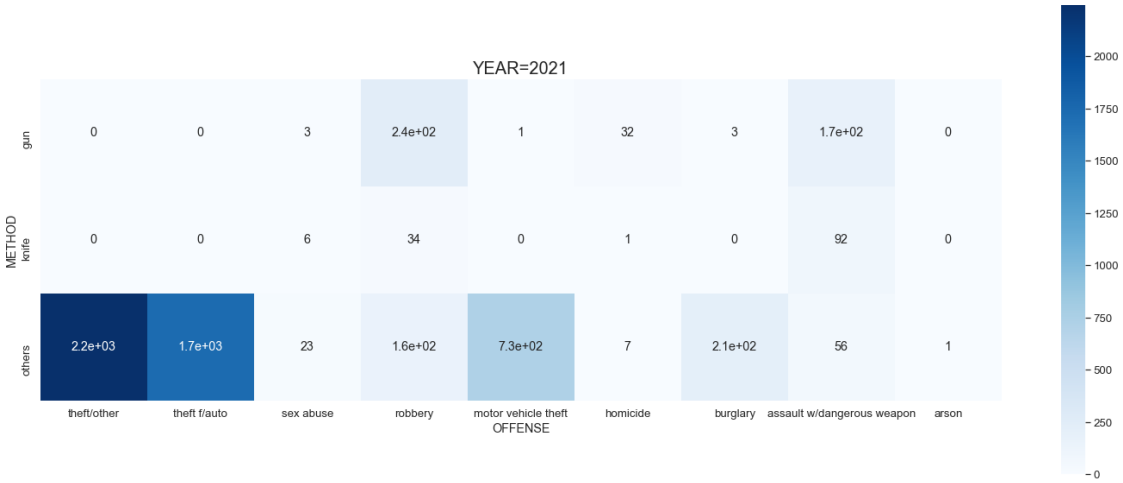

YEAR=2016
METHOD
gun
knife
others
1
0
15
1.2e+03
1
1e+02
0
7.5e+02
0
0
0
11
1.7e+02
0
12
0
8e+02
0
1.4e+04
1.2e+04
3.1e+02
1.5e+03
2.6e+03
19
2.1e+03
6.8e+02
6
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
14000
12000
10000
8000
6000
4000
2000
0


YEAR=2017
METHOD
gun
knife
others
0
0
11
7.9e+02
0
85
3
6.7e+02
0
2
0
12
1.5e+02
0
9
1
6.4e+02
0
1.4e+04
1e+04
2.6e+02
1.2e+03
2.3e+03
18
1.5e+03
5.2e+02
5
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
14000
12000
10000
8000
6000
4000
2000
0


YEAR=2018
METHOD
gun
knife
others
1
0
12
8.1e+02
0
1.2e+02
6
6.2e+02
0
3
0
19
1.4e+02
0
8
2
5.9e+02
0
1.4e+04
1.2e+04
2.3e+02
1e+03
2.4e+03
26
1.4e+03
4.3e+02
5
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
12000
10000
8000
6000
4000
2000
0


YEAR=2019
METHOD
gun
knife
others
0 0 6 7.5e+02 0 1.3e+02 3 6.8e+02 0
0 0 3 1.6e+02 0 6 2 4.9e+02 0
1.5e+04 1.1e+04 1.7e+02 1.3e+03 2.2e+03 27 1.3e+03 3.6e+02 7
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
14000
12000
10000
8000
6000
4000
2000
0


YEAR=2020
METHOD
gun
knife
others
1 0 8 9.8e+02 3 1.7e+02 6 8.4e+02 0
0 0 10 1.2e+02 0 9 0 4.8e+02 0
1.1e+04 8.2e+03 1.5e+02 8.9e+02 3.3e+03 18 1.4e+03 3e+02 13
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
10000
8000
6000
4000
2000
0


YEAR=2021
METHOD
gun
knife
others
0 0 3 2.4e+02 1 32 3 1.7e+02 0
0 0 6 34 0 1 0 92 0
2.2e+03 1.7e+03 23 1.6e+02 7.3e+02 7 2.1e+02 56 1
theft/other
theft f/auto
sex abuse
robbery
motor vehicle theft
homicide
burglary
assault w/dangerous weapon
arson
OFFENSE
2000
1750
1500
1250
1000
750
500
250
0

*Concslusion: changes are not evident.

Correlation between numerical features.

[125]:
```
num_crime=crime_filtered.select_dtypes(exclude=object)
corr_matrix=num_crime.corr()
corr_matrix
```

[125]:

	NEIGHBORHOOD_CLUSTER	CENSUS_TRACT	LONGITUDE	YBLOCK
NEIGHBORHOOD_CLUSTER	1.000000	0.560625	0.752110	-0.585083
CENSUS_TRACT	0.560625	1.000000	0.657579	-0.546590
LONGITUDE	0.752110	0.657579	1.000000	-0.451234
YBLOCK	-0.585083	-0.546590	-0.451234	1.000000
DISTRICT	0.682657	0.384141	0.621790	-0.363784
WARD	0.930668	0.569928	0.735048	-0.609351
YEAR	-0.017776	0.005625	0.001822	0.029207
PSA	0.683386	0.386111	0.622299	-0.367331
ucr-rank	-0.110959	-0.062285	-0.091978	0.113288
VOTING_PRECINCT	0.639953	0.507534	0.596980	-0.547420
XBLOCK	0.752173	0.657463	1.000000	-0.451164
CCN	-0.018740	0.005267	0.000915	0.029530
LATITUDE	-0.585093	-0.546538	-0.451225	1.000000

	DISTRICT	WARD	YEAR	PSA	ucr-rank
NEIGHBORHOOD_CLUSTER	0.682657	0.930668	-0.017776	0.683386	-0.110959
CENSUS_TRACT	0.384141	0.569928	0.005625	0.386111	-0.062285
LONGITUDE	0.621790	0.735048	0.001822	0.622299	-0.091978
YBLOCK	-0.363784	-0.609351	0.029207	-0.367331	0.113288
DISTRICT	1.000000	0.591582	-0.021697	0.998871	-0.123505
WARD	0.591582	1.000000	-0.009860	0.592013	-0.107295
YEAR	-0.021697	-0.009860	1.000000	-0.021016	0.088321
PSA	0.998871	0.592013	-0.021016	1.000000	-0.123619
ucr-rank	-0.123505	-0.107295	0.088321	-0.123619	1.000000
VOTING_PRECINCT	0.418401	0.599708	-0.000049	0.418048	-0.079630
XBLOCK	0.621857	0.735089	0.001815	0.622366	-0.091981
CCN	-0.023020	-0.010728	0.998122	-0.022338	0.089290
LATITUDE	-0.363796	-0.609352	0.029211	-0.367343	0.113287

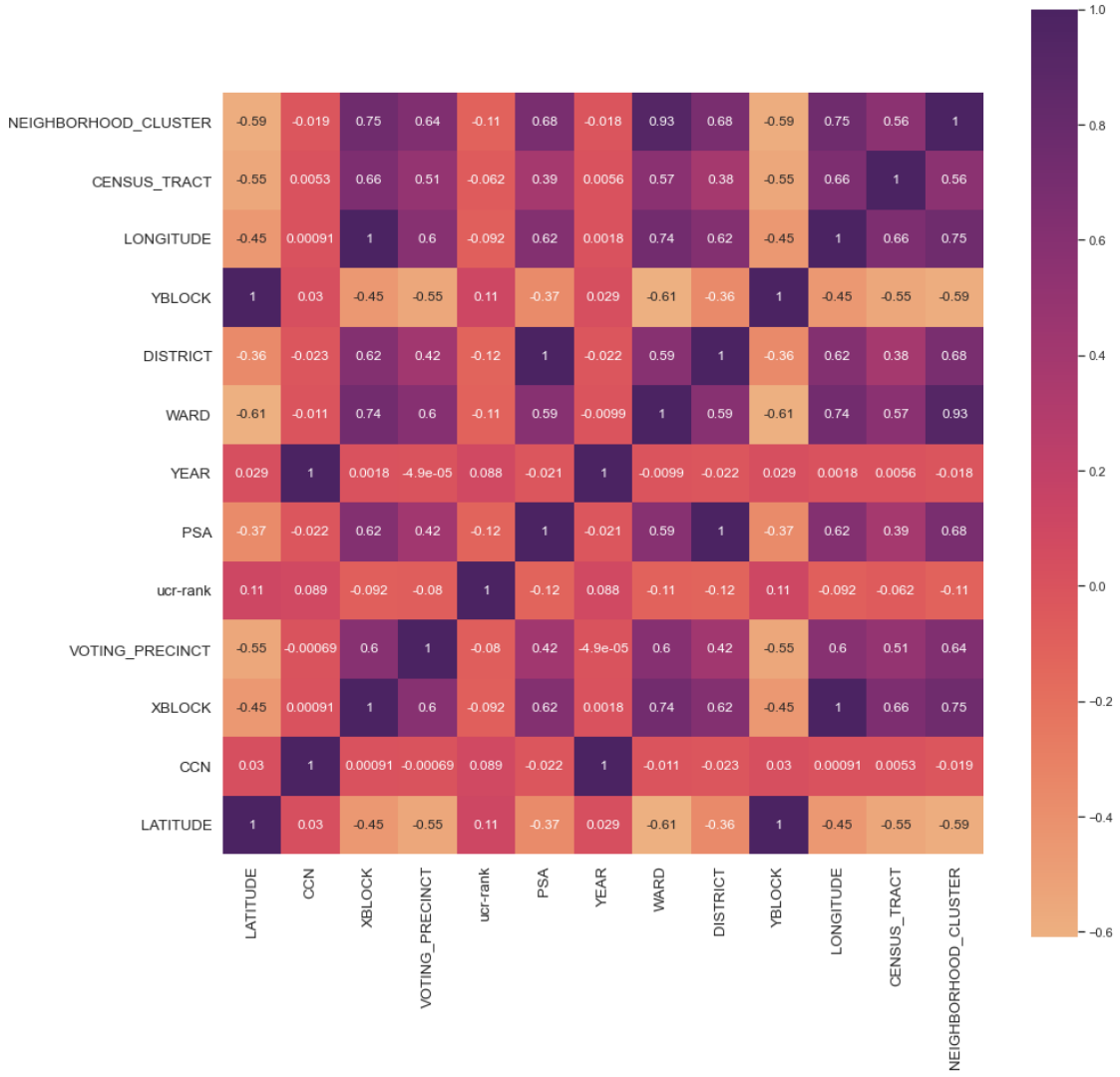
	VOTING_PRECINCT	XBLOCK	CCN	LATITUDE
NEIGHBORHOOD_CLUSTER	0.639953	0.752173	-0.018740	-0.585093
CENSUS_TRACT	0.507534	0.657463	0.005267	-0.546538
LONGITUDE	0.596980	1.000000	0.000915	-0.451225
YBLOCK	-0.547420	-0.451164	0.029530	1.000000
DISTRICT	0.418401	0.621857	-0.023020	-0.363796
WARD	0.599708	0.735089	-0.010728	-0.609352

```
YEAR                    -0.000049  0.001815  0.998122  0.029211
PSA                      0.418048  0.622366 -0.022338 -0.367343
ucr-rank                -0.079630 -0.091981  0.089290  0.113287
VOTING_PRECINCT          1.000000  0.596980 -0.000695 -0.547383
XBLOCK                   0.596980  1.000000  0.000908 -0.451155
CCN                     -0.000695  0.000908  1.000000  0.029534
LATITUDE                -0.547383 -0.451155  0.029534  1.000000
```

Heat map for correlation.

[126]:
```
# define a function to draw correlation heatmaps
def heat(corr_matrix,year):
    f, ax=plt.subplots(figsize=(15, 15))
    sns.set(font_scale=1)
    sns.heatmap(corr_matrix, cmap = 'flare',annot = True,square=True)
    if year !='0':
        plt.title('YEAR='+year,fontsize=20)
        ax.set_ylim([12,0])
        ax.set_xlim([12,0])
        plt.show()
    else:
        ax.set_ylim([13,0])
        ax.set_xlim([13,0])
        plt.show()
```

[127]:
```
heat(corr_matrix,'0')
```

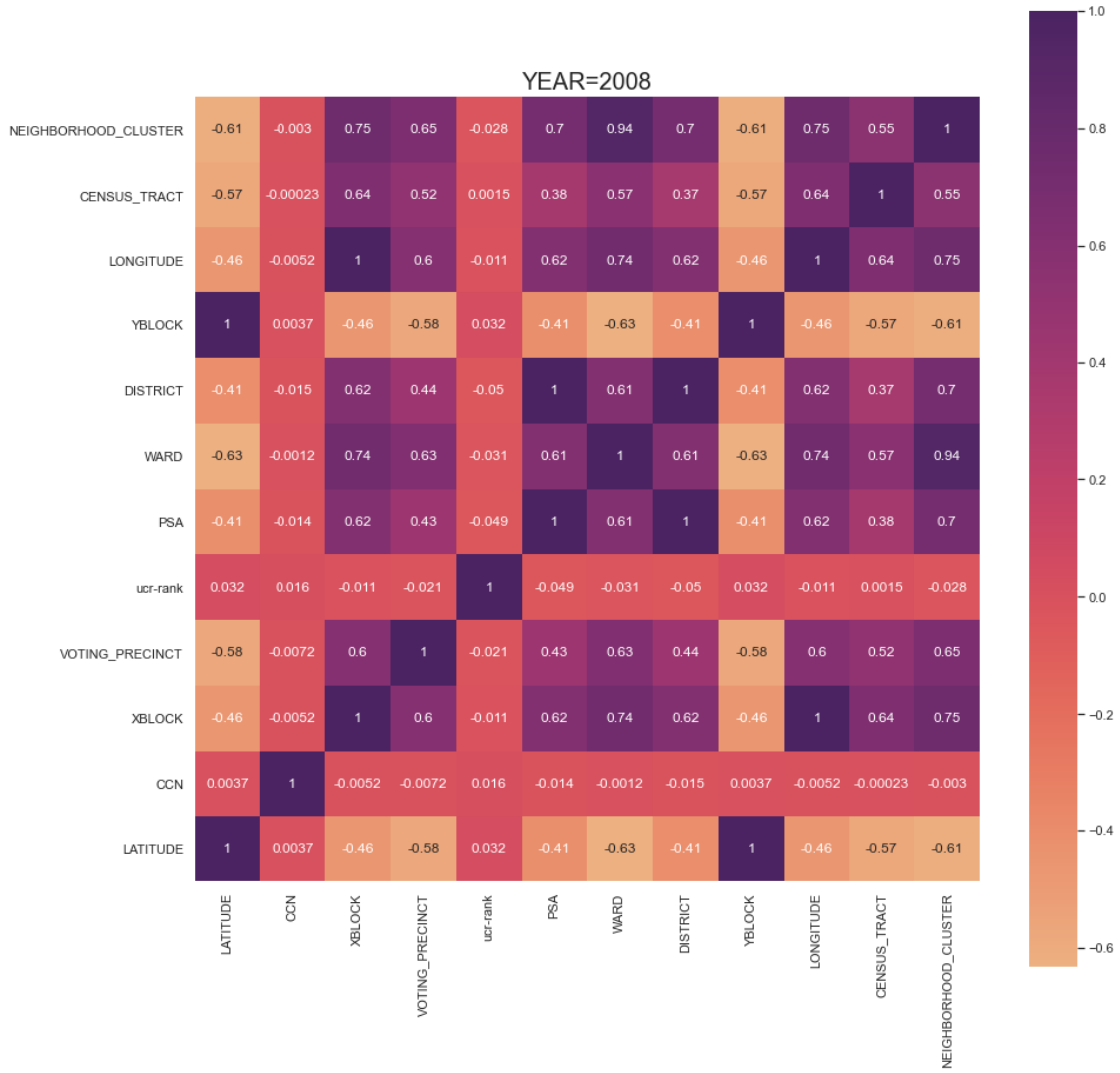


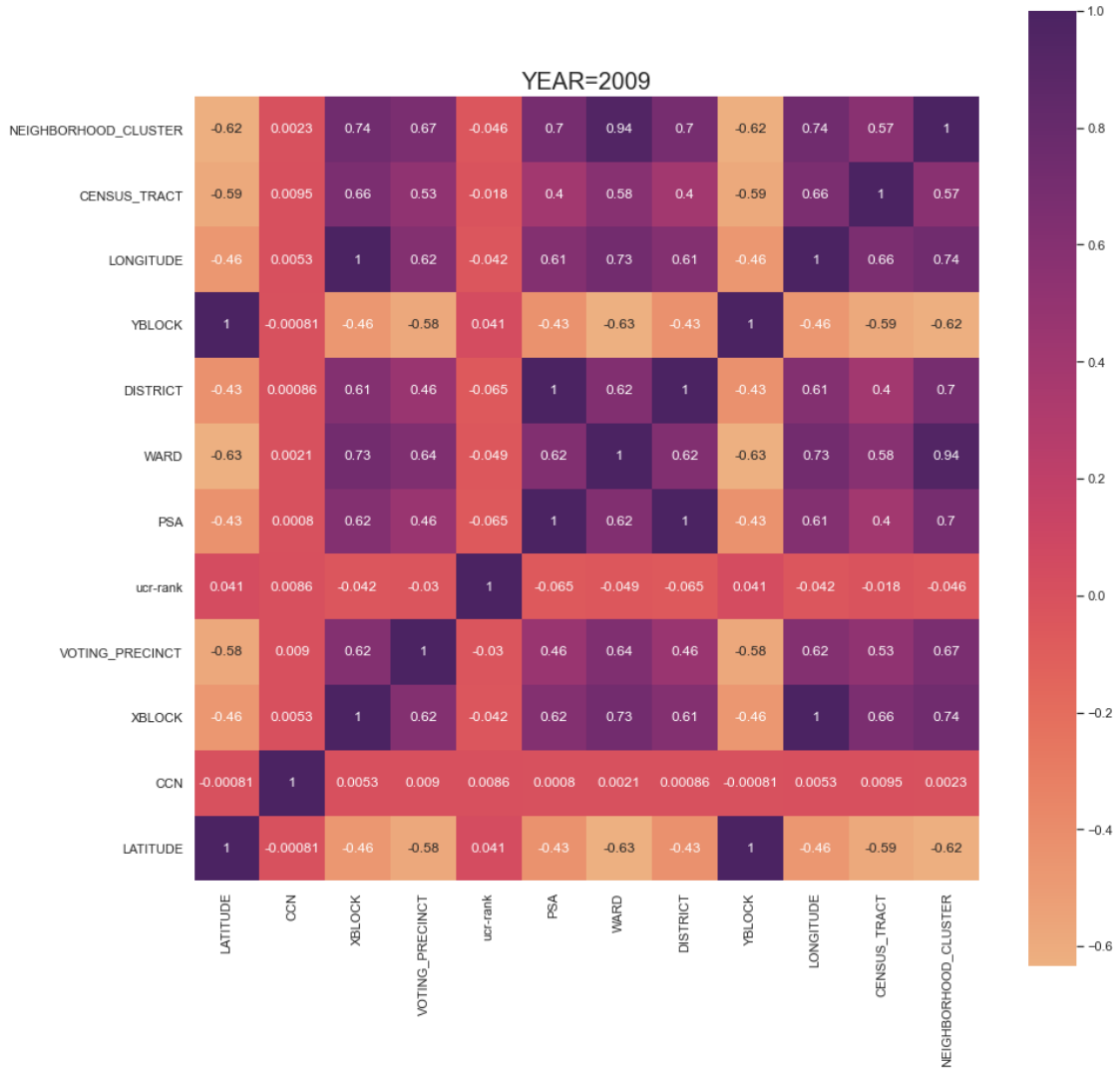
Correlations between numerical features with time changes.

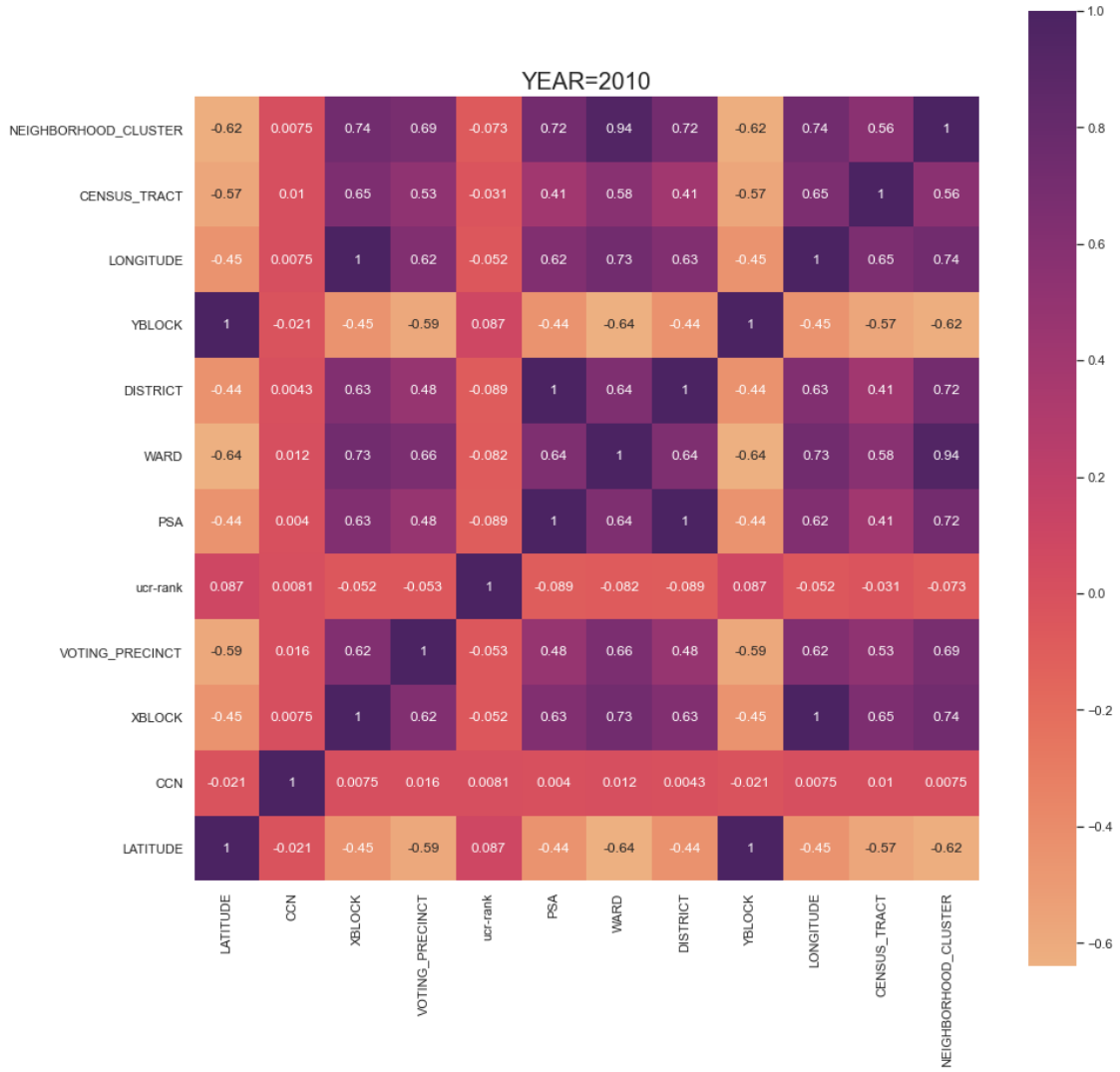
[128]:
```
# correlation matrix of different years
corr_2021=num_crime[num_crime.YEAR==2021].drop(columns='YEAR').corr()
corr_2020=num_crime[num_crime.YEAR==2020].drop(columns='YEAR').corr()
corr_2019=num_crime[num_crime.YEAR==2019].drop(columns='YEAR').corr()
corr_2018=num_crime[num_crime.YEAR==2018].drop(columns='YEAR').corr()
corr_2017=num_crime[num_crime.YEAR==2017].drop(columns='YEAR').corr()
corr_2016=num_crime[num_crime.YEAR==2016].drop(columns='YEAR').corr()
corr_2015=num_crime[num_crime.YEAR==2015].drop(columns='YEAR').corr()
corr_2014=num_crime[num_crime.YEAR==2014].drop(columns='YEAR').corr()
corr_2013=num_crime[num_crime.YEAR==2013].drop(columns='YEAR').corr()
corr_2012=num_crime[num_crime.YEAR==2012].drop(columns='YEAR').corr()
corr_2011=num_crime[num_crime.YEAR==2011].drop(columns='YEAR').corr()
```

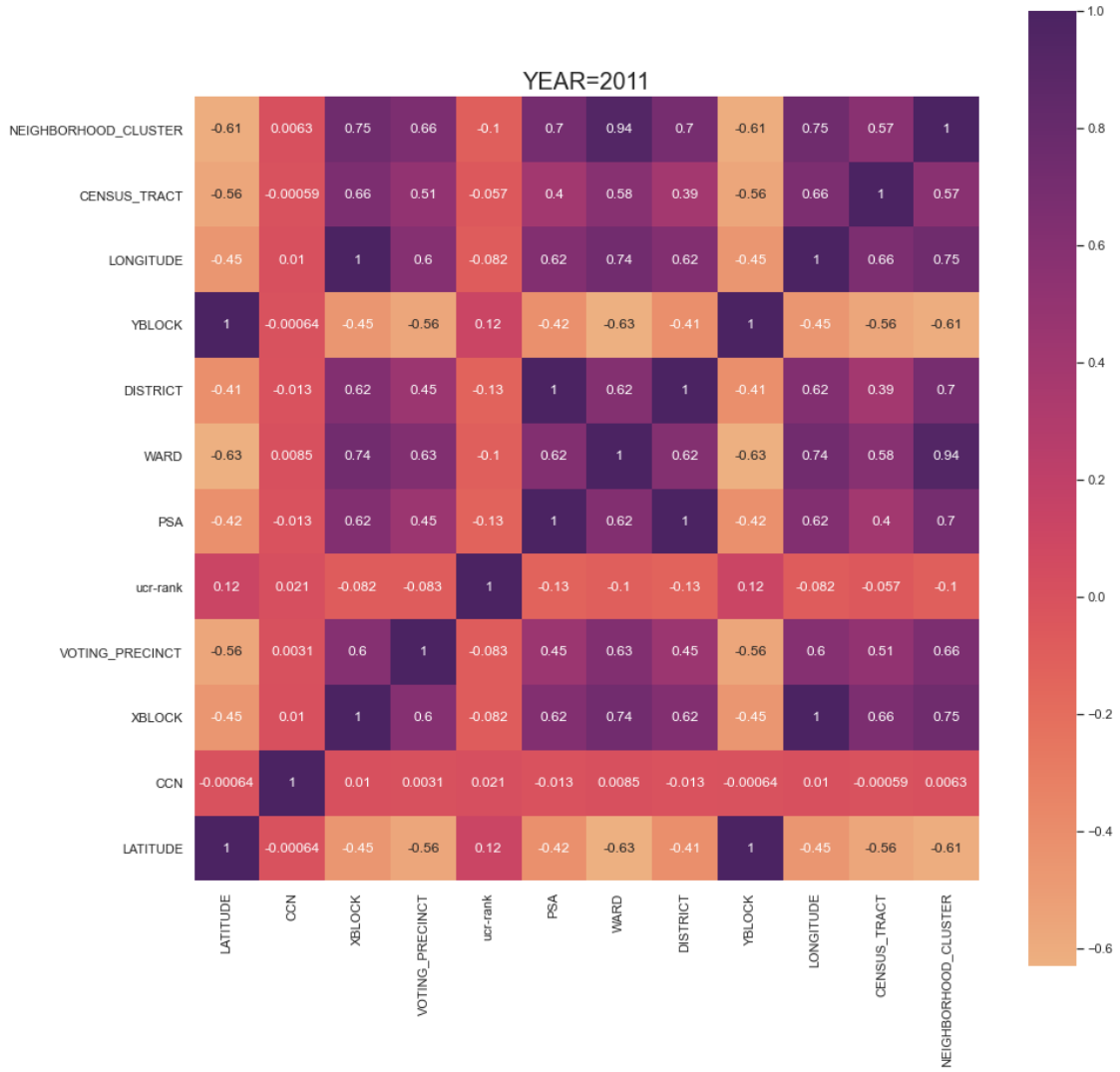
```
corr_2010=num_crime[num_crime.YEAR==2010].drop(columns='YEAR').corr()
corr_2009=num_crime[num_crime.YEAR==2009].drop(columns='YEAR').corr()
corr_2008=num_crime[num_crime.YEAR==2008].drop(columns='YEAR').corr()
```

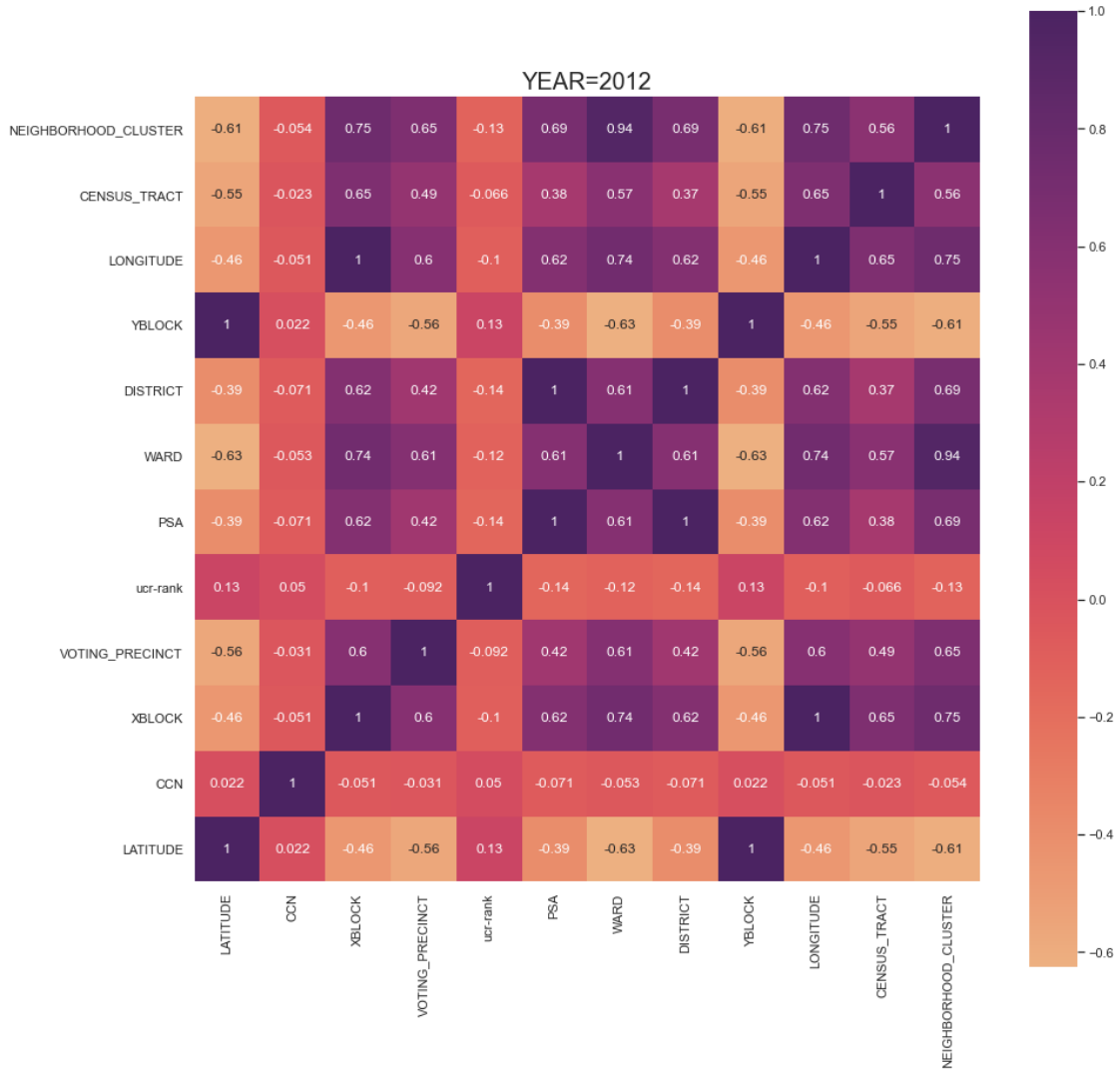
[129]:
```
heat(corr_2008,'2008')
heat(corr_2009,'2009')
heat(corr_2010,'2010')
heat(corr_2011,'2011')
heat(corr_2012,'2012')
heat(corr_2013,'2013')
heat(corr_2014,'2014')
heat(corr_2015,'2015')
heat(corr_2016,'2016')
heat(corr_2017,'2017')
heat(corr_2018,'2018')
heat(corr_2019,'2019')
heat(corr_2020,'2020')
heat(corr_2021,'2021')
```

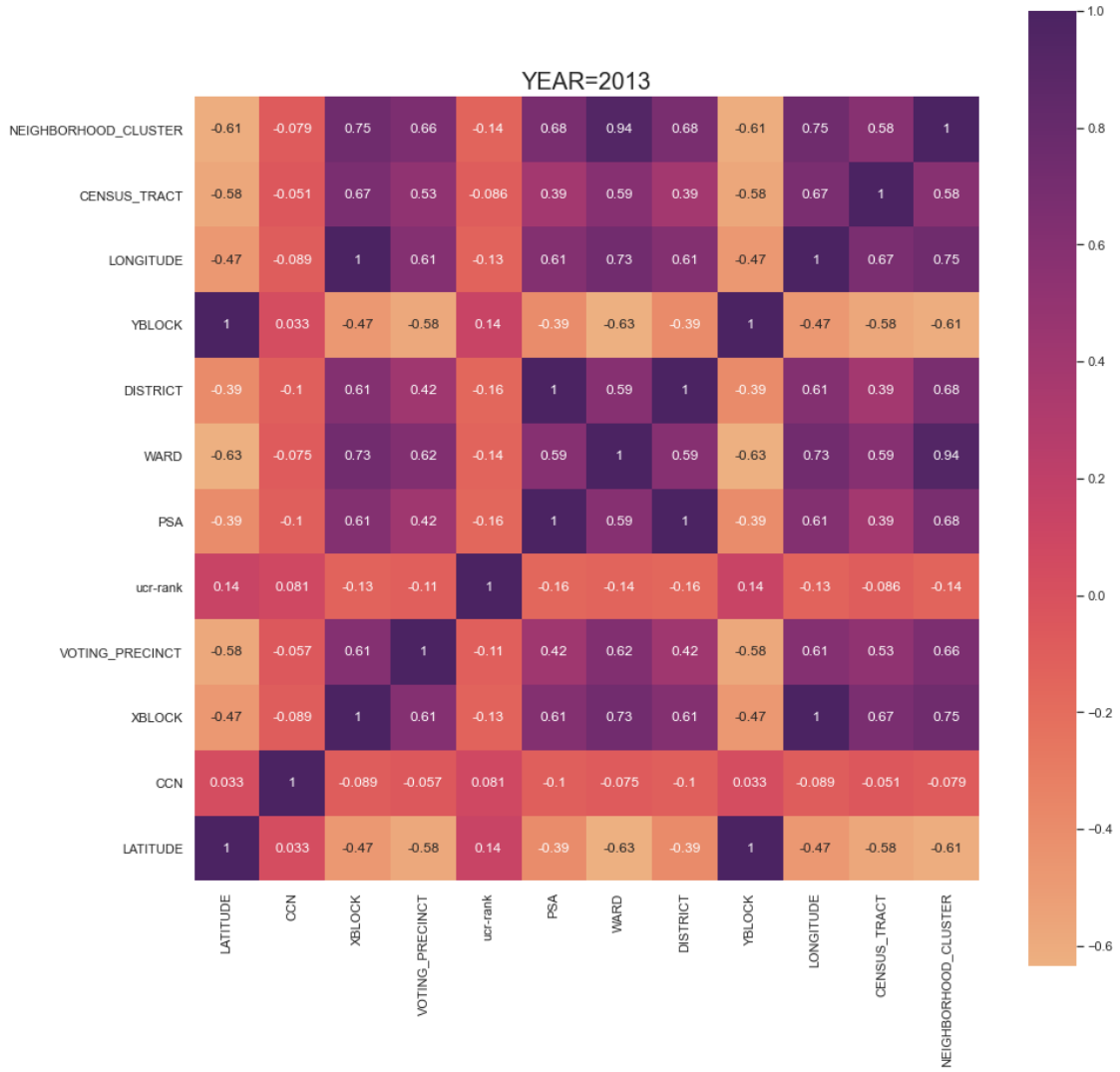

YEAR=2008
NEIGHBORHOOD_CLUSTER -0.61 -0.003 0.75 0.65 -0.028 0.7 0.94 0.7 -0.61 0.75 0.55 1
CENSUS_TRACT -0.57 -0.00023 0.64 0.52 0.0015 0.38 0.57 0.37 -0.57 0.64 1 0.55
LONGITUDE -0.46 -0.0052 1 0.6 -0.011 0.62 0.74 0.62 -0.46 1 0.64 0.75
YBLOCK 1 0.0037 -0.46 -0.58 0.032 -0.41 -0.63 -0.41 1 -0.46 -0.57 -0.61
DISTRICT -0.41 -0.015 0.62 0.44 -0.05 1 0.61 1 -0.41 0.62 0.37 0.7
WARD -0.63 -0.0012 0.74 0.63 -0.031 0.61 1 0.61 -0.63 0.74 0.57 0.94
PSA -0.41 -0.014 0.62 0.43 -0.049 1 0.61 1 -0.41 0.62 0.38 0.7
ucr-rank 0.032 0.016 -0.011 -0.021 1 -0.049 -0.031 -0.05 0.032 -0.011 0.0015 -0.028
VOTING_PRECINCT -0.58 -0.0072 0.6 1 -0.021 0.43 0.63 0.44 -0.58 0.6 0.52 0.65
XBLOCK -0.46 -0.0052 1 0.6 -0.011 0.62 0.74 0.62 -0.46 1 0.64 0.75
CCN 0.0037 1 -0.0052 -0.0072 0.016 -0.014 -0.0012 -0.015 0.0037 -0.0052 -0.00023 -0.003
LATITUDE 1 0.0037 -0.46 -0.58 0.032 -0.41 -0.63 -0.41 1 -0.46 -0.57 -0.61
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

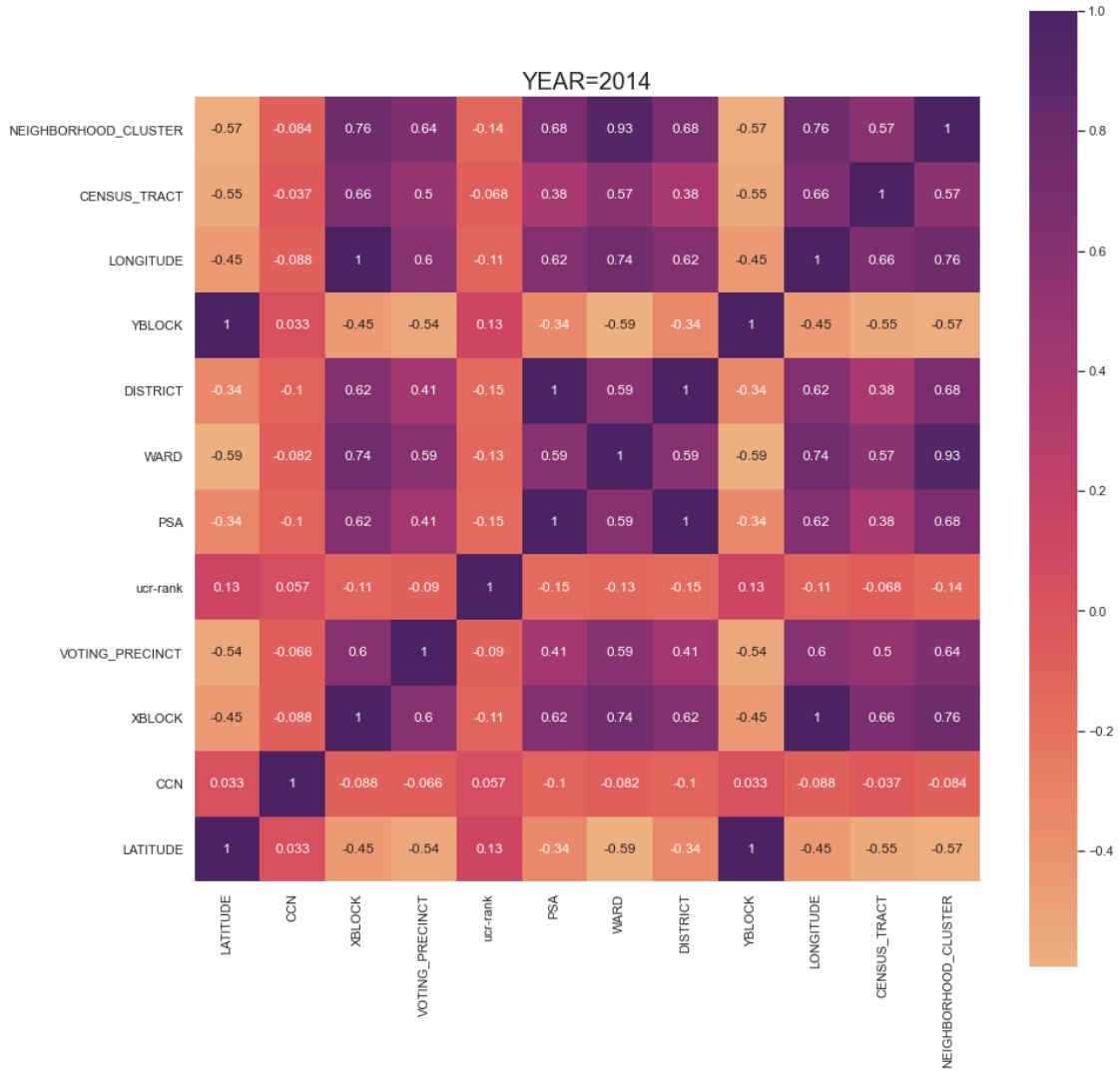

YEAR=2009
NEIGHBORHOOD_CLUSTER -0.62 0.0023 0.74 0.67 -0.046 0.7 0.94 0.7 -0.62 0.74 0.57 1
CENSUS_TRACT -0.59 0.0095 0.66 0.53 -0.018 0.4 0.58 0.4 -0.59 0.66 1 0.57
LONGITUDE -0.46 0.0053 1 0.62 -0.042 0.61 0.73 0.61 -0.46 1 0.66 0.74
YBLOCK 1 -0.00081 -0.46 -0.58 0.041 -0.43 -0.63 -0.43 1 -0.46 -0.59 -0.62
DISTRICT -0.43 0.00086 0.61 0.46 -0.065 1 0.62 1 -0.43 0.61 0.4 0.7
WARD -0.63 0.0021 0.73 0.64 -0.049 0.62 1 0.62 -0.63 0.73 0.58 0.94
PSA -0.43 0.0008 0.62 0.46 -0.065 1 0.62 1 -0.43 0.61 0.4 0.7
ucr-rank 0.041 0.0086 -0.042 -0.03 1 -0.065 -0.049 -0.065 0.041 -0.042 -0.018 -0.046
VOTING_PRECINCT -0.58 0.009 0.62 1 -0.03 0.46 0.64 0.46 -0.58 0.62 0.53 0.67
XBLOCK -0.46 0.0053 1 0.62 -0.042 0.62 0.73 0.61 -0.46 1 0.66 0.74
CCN -0.00081 1 0.0053 0.009 0.0086 0.0008 0.0021 0.00086 -0.00081 0.0053 0.0095 0.0023
LATITUDE 1 -0.00081 -0.46 -0.58 0.041 -0.43 -0.63 -0.43 1 -0.46 -0.59 -0.62
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

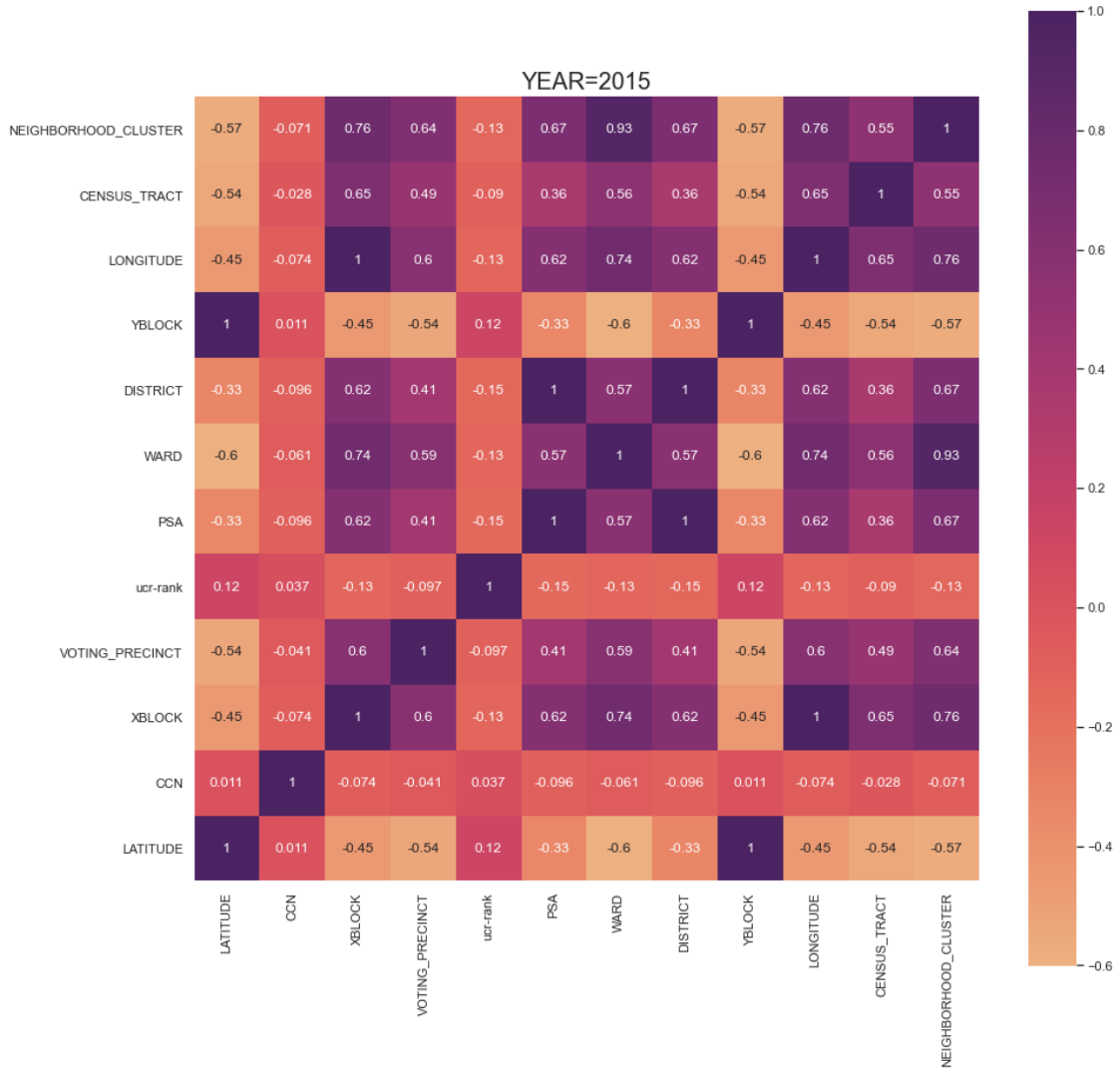

YEAR=2010
NEIGHBORHOOD_CLUSTER -0.62 0.0075 0.74 0.69 -0.073 0.72 0.94 0.72 -0.62 0.74 0.56 1
CENSUS_TRACT -0.57 0.01 0.65 0.53 -0.031 0.41 0.58 0.41 -0.57 0.65 1 0.56
LONGITUDE -0.45 0.0075 1 0.62 -0.052 0.62 0.73 0.63 -0.45 1 0.65 0.74
YBLOCK 1 -0.021 -0.45 -0.59 0.087 -0.44 -0.64 -0.44 1 -0.45 -0.57 -0.62
DISTRICT -0.44 0.0043 0.63 0.48 -0.089 1 0.64 1 -0.44 0.63 0.41 0.72
WARD -0.64 0.012 0.73 0.66 -0.082 0.64 1 0.64 -0.64 0.73 0.58 0.94
PSA -0.44 0.004 0.63 0.48 -0.089 1 0.64 1 -0.44 0.62 0.41 0.72
ucr-rank 0.087 0.0081 -0.052 -0.053 1 -0.089 -0.082 -0.089 0.087 -0.052 -0.031 -0.073
VOTING_PRECINCT -0.59 0.016 0.62 1 -0.053 0.48 0.66 0.48 -0.59 0.62 0.53 0.69
XBLOCK -0.45 0.0075 1 0.62 -0.052 0.63 0.73 0.63 -0.45 1 0.65 0.74
CCN -0.021 1 0.0075 0.016 0.0081 0.004 0.012 0.0043 -0.021 0.0075 0.01 0.0075
LATITUDE 1 -0.021 -0.45 -0.59 0.087 -0.44 -0.64 -0.44 1 -0.45 -0.57 -0.62
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

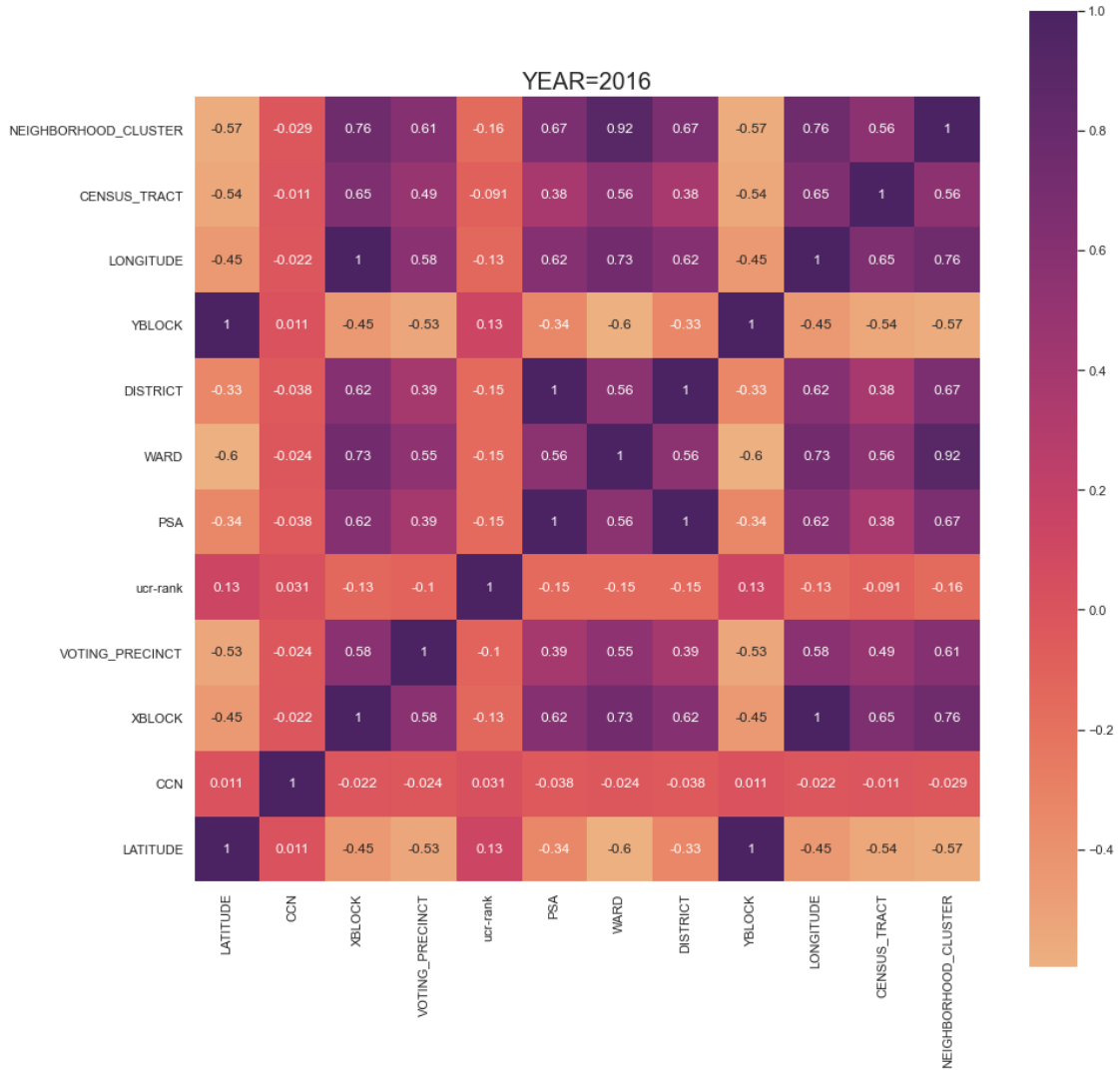

YEAR=2011
NEIGHBORHOOD_CLUSTER -0.61 0.0063 0.75 0.66 -0.1 0.7 0.94 0.7 -0.61 0.75 0.57 1
CENSUS_TRACT -0.56 -0.00059 0.66 0.51 -0.057 0.4 0.58 0.39 -0.56 0.66 1 0.57
LONGITUDE -0.45 0.01 1 0.6 -0.082 0.62 0.74 0.62 -0.45 1 0.66 0.75
YBLOCK 1 -0.00064 -0.45 -0.56 0.12 -0.42 -0.63 -0.41 1 -0.45 -0.56 -0.61
DISTRICT -0.41 -0.013 0.62 0.45 -0.13 1 0.62 1 -0.41 0.62 0.39 0.7
WARD -0.63 0.0085 0.74 0.63 -0.1 0.62 1 0.62 -0.63 0.74 0.58 0.94
PSA -0.42 -0.013 0.62 0.45 -0.13 1 0.62 1 -0.42 0.62 0.4 0.7
ucr-rank 0.12 0.021 -0.082 -0.083 1 -0.13 -0.1 -0.13 0.12 -0.082 -0.057 -0.1
VOTING_PRECINCT -0.56 0.0031 0.6 1 -0.083 0.45 0.63 0.45 -0.56 0.6 0.51 0.66
XBLOCK -0.45 0.01 1 0.6 -0.082 0.62 0.74 0.62 -0.45 1 0.66 0.75
CCN -0.00064 1 0.01 0.0031 0.021 -0.013 0.0085 -0.013 -0.00064 0.01 -0.00059 0.0063
LATITUDE 1 -0.00064 -0.45 -0.56 0.12 -0.42 -0.63 -0.41 1 -0.45 -0.56 -0.61
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

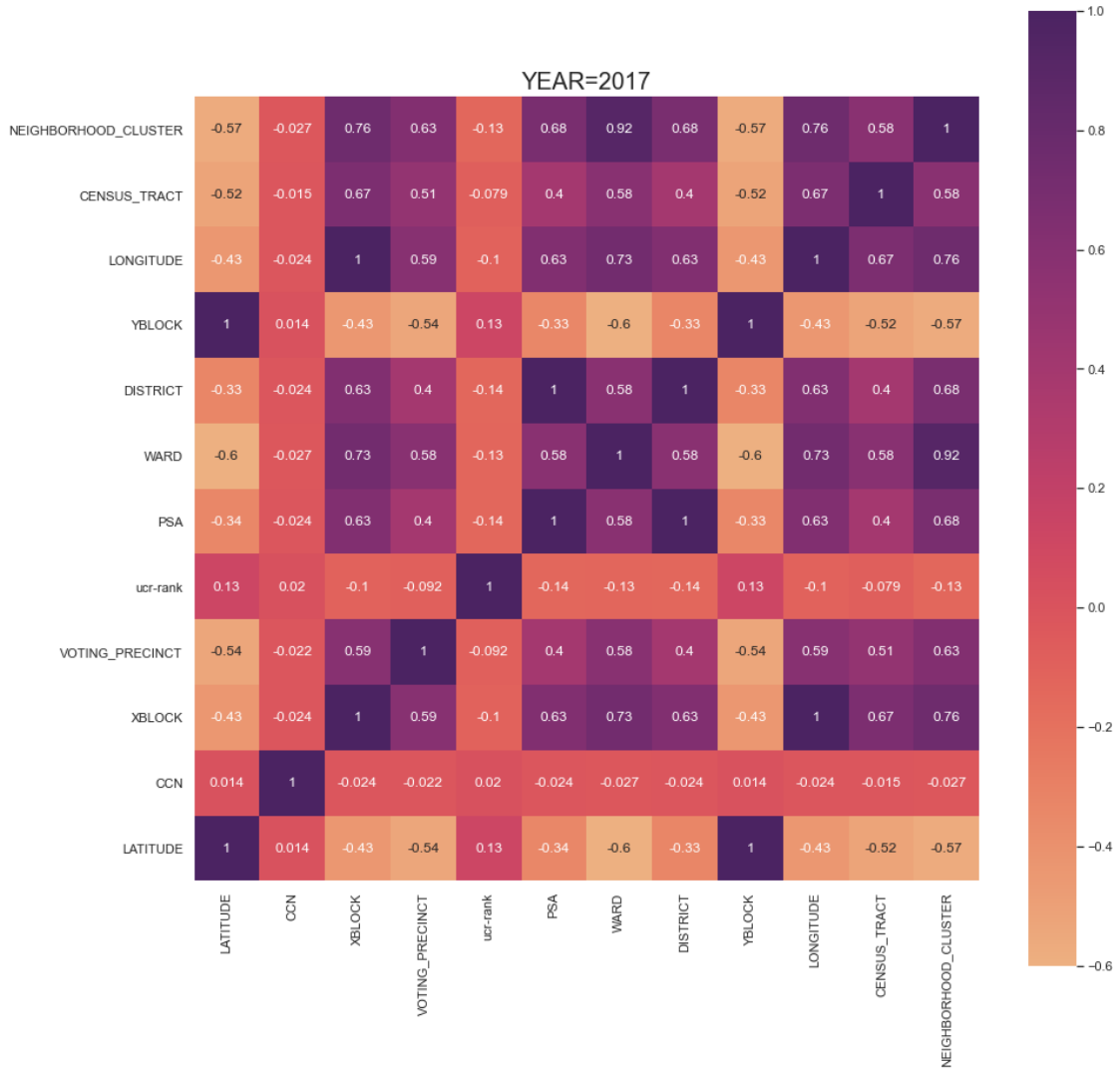

YEAR=2012
NEIGHBORHOOD_CLUSTER -0.61 -0.054 0.75 0.65 -0.13 0.69 0.94 0.69 -0.61 0.75 0.56 1
CENSUS_TRACT -0.55 -0.023 0.65 0.49 -0.066 0.38 0.57 0.37 -0.55 0.65 1 0.56
LONGITUDE -0.46 -0.051 1 0.6 -0.1 0.62 0.74 0.62 -0.46 1 0.65 0.75
YBLOCK 1 0.022 -0.46 -0.56 0.13 -0.39 -0.63 -0.39 1 -0.46 -0.55 -0.61
DISTRICT -0.39 -0.071 0.62 0.42 -0.14 1 0.61 1 -0.39 0.62 0.37 0.69
WARD -0.63 -0.053 0.74 0.61 -0.12 0.61 1 0.61 -0.63 0.74 0.57 0.94
PSA -0.39 -0.071 0.62 0.42 -0.14 1 0.61 1 -0.39 0.62 0.38 0.69
ucr-rank 0.13 0.05 -0.1 -0.092 1 -0.14 -0.12 -0.14 0.13 -0.1 -0.066 -0.13
VOTING_PRECINCT -0.56 -0.031 0.6 1 -0.092 0.42 0.61 0.42 -0.56 0.6 0.49 0.65
XBLOCK -0.46 -0.051 1 0.6 -0.1 0.62 0.74 0.62 -0.46 1 0.65 0.75
CCN 0.022 1 -0.051 -0.031 0.05 -0.071 -0.053 -0.071 0.022 -0.051 -0.023 -0.054
LATITUDE 1 0.022 -0.46 -0.56 0.13 -0.39 -0.63 -0.39 1 -0.46 -0.55 -0.61
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

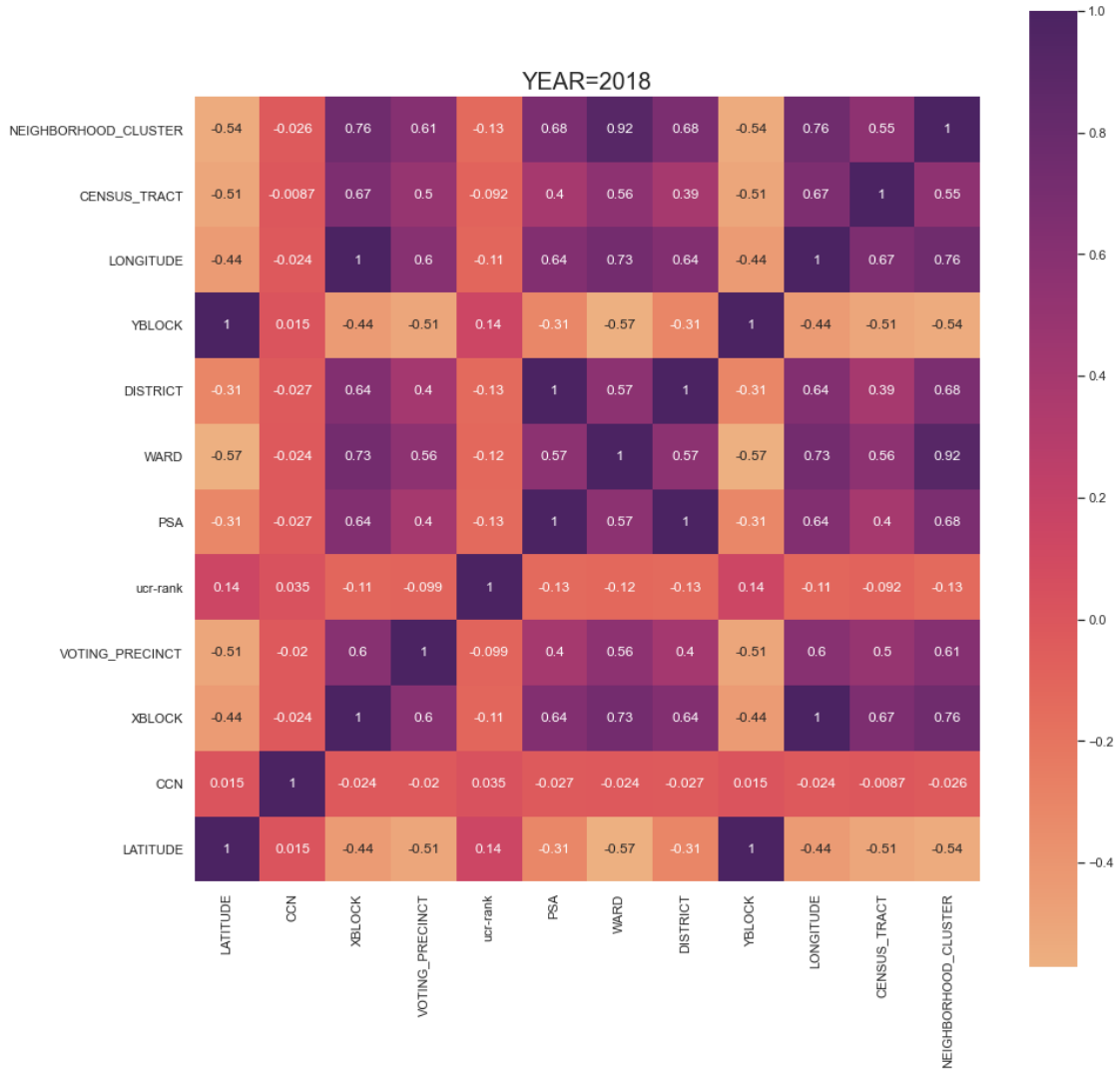

YEAR=2013
NEIGHBORHOOD_CLUSTER -0.61 -0.079 0.75 0.66 -0.14 0.68 0.94 0.68 -0.61 0.75 0.58 1
CENSUS_TRACT -0.58 -0.051 0.67 0.53 -0.086 0.39 0.59 0.39 -0.58 0.67 1 0.58
LONGITUDE -0.47 -0.089 1 0.61 -0.13 0.61 0.73 0.61 -0.47 1 0.67 0.75
YBLOCK 1 0.033 -0.47 -0.58 0.14 -0.39 -0.63 -0.39 1 -0.47 -0.58 -0.61
DISTRICT -0.39 -0.1 0.61 0.42 -0.16 1 0.59 1 -0.39 0.61 0.39 0.68
WARD -0.63 -0.075 0.73 0.62 -0.14 0.59 1 0.59 -0.63 0.73 0.59 0.94
PSA -0.39 -0.1 0.61 0.42 -0.16 1 0.59 1 -0.39 0.61 0.39 0.68
ucr-rank 0.14 0.081 -0.13 -0.11 1 -0.16 -0.14 -0.16 0.14 -0.13 -0.086 -0.14
VOTING_PRECINCT -0.58 -0.057 0.61 1 -0.11 0.42 0.62 0.42 -0.58 0.61 0.53 0.66
XBLOCK -0.47 -0.089 1 0.61 -0.13 0.61 0.73 0.61 -0.47 1 0.67 0.75
CCN 0.033 1 -0.089 -0.057 0.081 -0.1 -0.075 -0.1 0.033 -0.089 -0.051 -0.079
LATITUDE 1 0.033 -0.47 -0.58 0.14 -0.39 -0.63 -0.39 1 -0.47 -0.58 -0.61
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

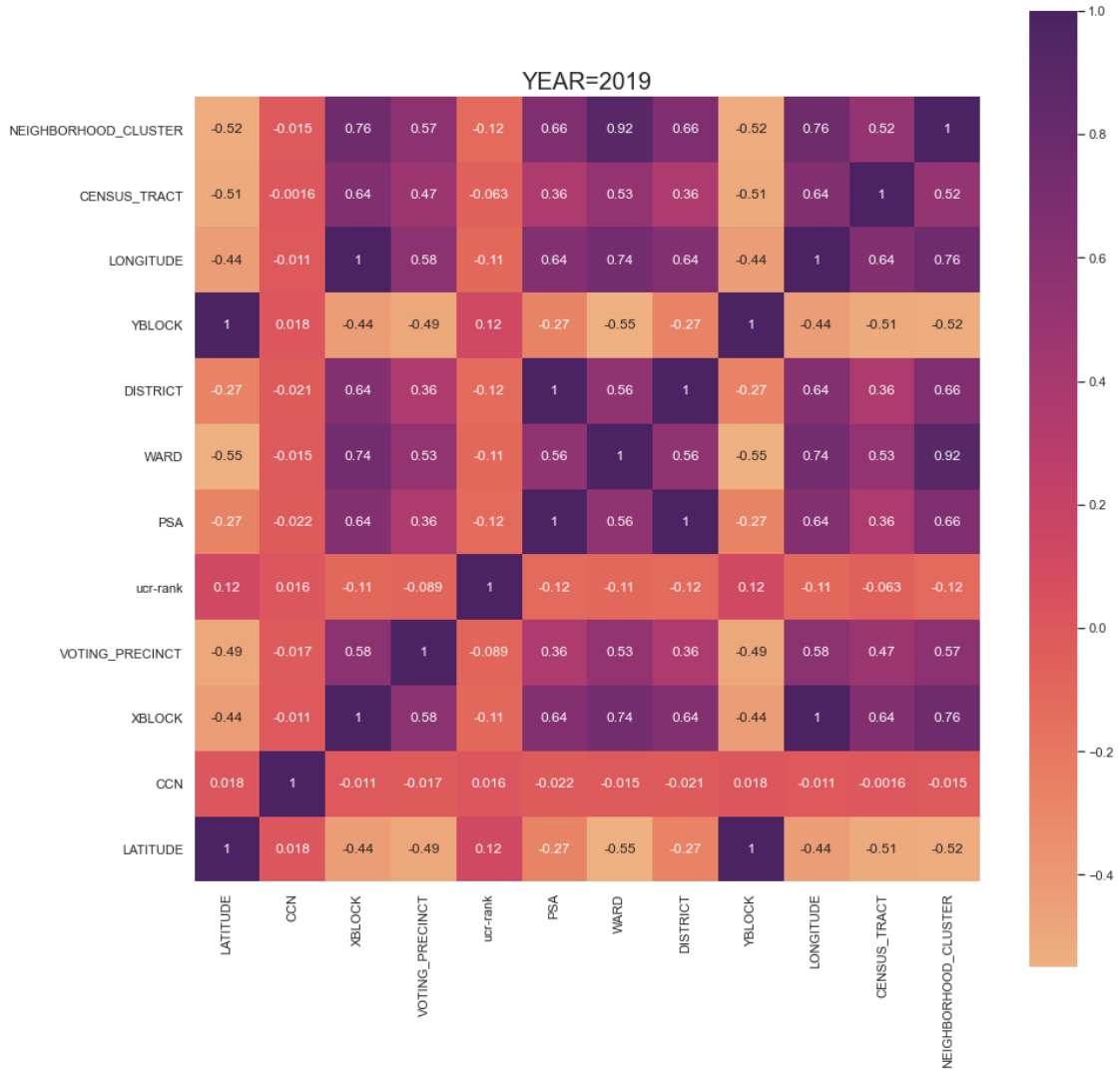

YEAR=2014
NEIGHBORHOOD_CLUSTER -0.57 -0.084 0.76 0.64 -0.14 0.68 0.93 0.68 -0.57 0.76 0.57 1
CENSUS_TRACT -0.55 -0.037 0.66 0.5 -0.068 0.38 0.57 0.38 -0.55 0.66 1 0.57
LONGITUDE -0.45 -0.088 1 0.6 -0.11 0.62 0.74 0.62 -0.45 1 0.66 0.76
YBLOCK 1 0.033 -0.45 -0.54 0.13 -0.34 -0.59 -0.34 1 -0.45 -0.55 -0.57
DISTRICT -0.34 -0.1 0.62 0.41 -0.15 1 0.59 1 -0.34 0.62 0.38 0.68
WARD -0.59 -0.082 0.74 0.59 -0.13 0.59 1 0.59 -0.59 0.74 0.57 0.93
PSA -0.34 -0.1 0.62 0.41 -0.15 1 0.59 1 -0.34 0.62 0.38 0.68
ucr-rank 0.13 0.057 -0.11 -0.09 1 -0.15 -0.13 -0.15 0.13 -0.11 -0.068 -0.14
VOTING_PRECINCT -0.54 -0.066 0.6 1 -0.09 0.41 0.59 0.41 -0.54 0.6 0.5 0.64
XBLOCK -0.45 -0.088 1 0.6 -0.11 0.62 0.74 0.62 -0.45 1 0.66 0.76
CCN 0.033 1 -0.088 -0.066 0.057 -0.1 -0.082 -0.1 0.033 -0.088 -0.037 -0.084
LATITUDE 1 0.033 -0.45 -0.54 0.13 -0.34 -0.59 -0.34 1 -0.45 -0.55 -0.57
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

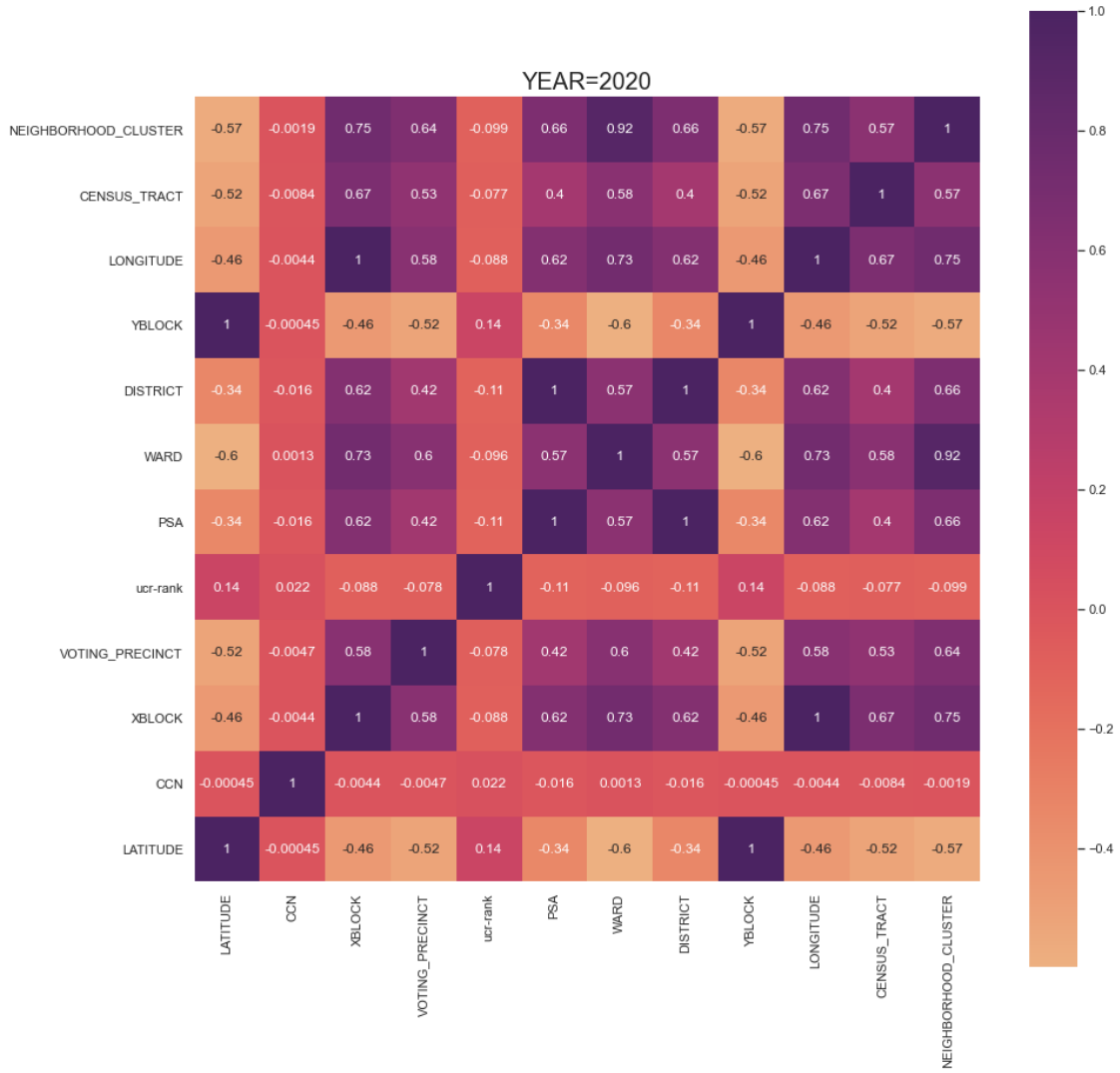

YEAR=2015
NEIGHBORHOOD_CLUSTER -0.57 -0.071 0.76 0.64 -0.13 0.67 0.93 0.67 -0.57 0.76 0.55 1
CENSUS_TRACT -0.54 -0.028 0.65 0.49 -0.09 0.36 0.56 0.36 -0.54 0.65 1 0.55
LONGITUDE -0.45 -0.074 1 0.6 -0.13 0.62 0.74 0.62 -0.45 1 0.65 0.76
YBLOCK 1 0.011 -0.45 -0.54 0.12 -0.33 -0.6 -0.33 1 -0.45 -0.54 -0.57
DISTRICT -0.33 -0.096 0.62 0.41 -0.15 1 0.57 1 -0.33 0.62 0.36 0.67
WARD -0.6 -0.061 0.74 0.59 -0.13 0.57 1 0.57 -0.6 0.74 0.56 0.93
PSA -0.33 -0.096 0.62 0.41 -0.15 1 0.57 1 -0.33 0.62 0.36 0.67
ucr-rank 0.12 0.037 -0.13 -0.097 1 -0.15 -0.13 -0.15 0.12 -0.13 -0.09 -0.13
VOTING_PRECINCT -0.54 -0.041 0.6 1 -0.097 0.41 0.59 0.41 -0.54 0.6 0.49 0.64
XBLOCK -0.45 -0.074 1 0.6 -0.13 0.62 0.74 0.62 -0.45 1 0.65 0.76
CCN 0.011 1 -0.074 -0.041 0.037 -0.096 -0.061 -0.096 0.011 -0.074 -0.028 -0.071
LATITUDE 1 0.011 -0.45 -0.54 0.12 -0.33 -0.6 -0.33 1 -0.45 -0.54 -0.57
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

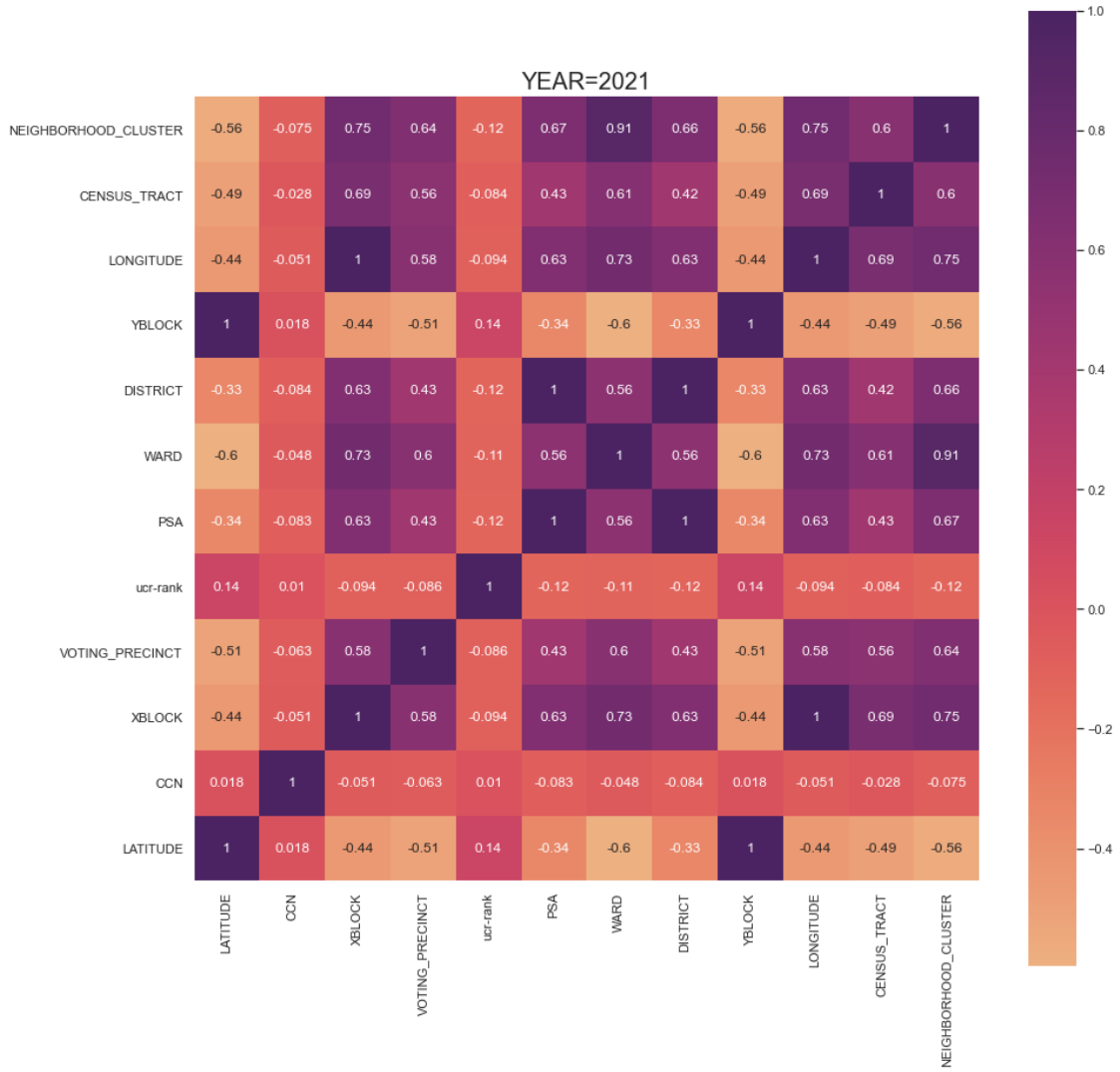

YEAR=2016
NEIGHBORHOOD_CLUSTER -0.57 -0.029 0.76 0.61 -0.16 0.67 0.92 0.67 -0.57 0.76 0.56 1
CENSUS_TRACT -0.54 -0.011 0.65 0.49 -0.091 0.38 0.56 0.38 -0.54 0.65 1 0.56
LONGITUDE -0.45 -0.022 1 0.58 -0.13 0.62 0.73 0.62 -0.45 1 0.65 0.76
YBLOCK 1 0.011 -0.45 -0.53 0.13 -0.34 -0.6 -0.33 1 -0.45 -0.54 -0.57
DISTRICT -0.33 -0.038 0.62 0.39 -0.15 1 0.56 1 -0.33 0.62 0.38 0.67
WARD -0.6 -0.024 0.73 0.55 -0.15 0.56 1 0.56 -0.6 0.73 0.56 0.92
PSA -0.34 -0.038 0.62 0.39 -0.15 1 0.56 1 -0.34 0.62 0.38 0.67
ucr-rank 0.13 0.031 -0.13 -0.1 1 -0.15 -0.15 -0.15 0.13 -0.13 -0.091 -0.16
VOTING_PRECINCT -0.53 -0.024 0.58 1 -0.1 0.39 0.55 0.39 -0.53 0.58 0.49 0.61
XBLOCK -0.45 -0.022 1 0.58 -0.13 0.62 0.73 0.62 -0.45 1 0.65 0.76
CCN 0.011 1 -0.022 -0.024 0.031 -0.038 -0.024 -0.038 0.011 -0.022 -0.011 -0.029
LATITUDE 1 0.011 -0.45 -0.53 0.13 -0.34 -0.6 -0.33 1 -0.45 -0.54 -0.57
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


YEAR=2017
NEIGHBORHOOD_CLUSTER -0.57 -0.027 0.76 0.63 -0.13 0.68 0.92 0.68 -0.57 0.76 0.58 1
CENSUS_TRACT -0.52 -0.015 0.67 0.51 -0.079 0.4 0.58 0.4 -0.52 0.67 1 0.58
LONGITUDE -0.43 -0.024 1 0.59 -0.1 0.63 0.73 0.63 -0.43 1 0.67 0.76
YBLOCK 1 0.014 -0.43 -0.54 0.13 -0.33 -0.6 -0.33 1 -0.43 -0.52 -0.57
DISTRICT -0.33 -0.024 0.63 0.4 -0.14 1 0.58 1 -0.33 0.63 0.4 0.68
WARD -0.6 -0.027 0.73 0.58 -0.13 0.58 1 0.58 -0.6 0.73 0.58 0.92
PSA -0.34 -0.024 0.63 0.4 -0.14 1 0.58 1 -0.33 0.63 0.4 0.68
ucr-rank 0.13 0.02 -0.1 -0.092 1 -0.14 -0.13 -0.14 0.13 -0.1 -0.079 -0.13
VOTING_PRECINCT -0.54 -0.022 0.59 1 -0.092 0.4 0.58 0.4 -0.54 0.59 0.51 0.63
XBLOCK -0.43 -0.024 1 0.59 -0.1 0.63 0.73 0.63 -0.43 1 0.67 0.76
CCN 0.014 1 -0.024 -0.022 0.02 -0.024 -0.027 -0.024 0.014 -0.024 -0.015 -0.027
LATITUDE 1 0.014 -0.43 -0.54 0.13 -0.34 -0.6 -0.33 1 -0.43 -0.52 -0.57
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


YEAR=2018
NEIGHBORHOOD_CLUSTER -0.54 -0.026 0.76 0.61 -0.13 0.68 0.92 0.68 -0.54 0.76 0.55 1
CENSUS_TRACT -0.51 -0.0087 0.67 0.5 -0.092 0.4 0.56 0.39 -0.51 0.67 1 0.55
LONGITUDE -0.44 -0.024 1 0.6 -0.11 0.64 0.73 0.64 -0.44 1 0.67 0.76
YBLOCK 1 0.015 -0.44 -0.51 0.14 -0.31 -0.57 -0.31 1 -0.44 -0.51 -0.54
DISTRICT -0.31 -0.027 0.64 0.4 -0.13 1 0.57 1 -0.31 0.64 0.39 0.68
WARD -0.57 -0.024 0.73 0.56 -0.12 0.57 1 0.57 -0.57 0.73 0.56 0.92
PSA -0.31 -0.027 0.64 0.4 -0.13 1 0.57 1 -0.31 0.64 0.4 0.68
ucr-rank 0.14 0.035 -0.11 -0.099 1 -0.13 -0.12 -0.13 0.14 -0.11 -0.092 -0.13
VOTING_PRECINCT -0.51 -0.02 0.6 1 -0.099 0.4 0.56 0.4 -0.51 0.6 0.5 0.61
XBLOCK -0.44 -0.024 1 0.6 -0.11 0.64 0.73 0.64 -0.44 1 0.67 0.76
CCN 0.015 1 -0.024 -0.02 0.035 -0.027 -0.024 -0.027 0.015 -0.024 -0.0087 -0.026
LATITUDE 1 0.015 -0.44 -0.51 0.14 -0.31 -0.57 -0.31 1 -0.44 -0.51 -0.54
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


YEAR=2019
NEIGHBORHOOD_CLUSTER -0.52 -0.015 0.76 0.57 -0.12 0.66 0.92 0.66 -0.52 0.76 0.52 1
CENSUS_TRACT -0.51 -0.0016 0.64 0.47 -0.063 0.36 0.53 0.36 -0.51 0.64 1 0.52
LONGITUDE -0.44 -0.011 1 0.58 -0.11 0.64 0.74 0.64 -0.44 1 0.64 0.76
YBLOCK 1 0.018 -0.44 -0.49 0.12 -0.27 -0.55 -0.27 1 -0.44 -0.51 -0.52
DISTRICT -0.27 -0.021 0.64 0.36 -0.12 1 0.56 1 -0.27 0.64 0.36 0.66
WARD -0.55 -0.015 0.74 0.53 -0.11 0.56 1 0.56 -0.55 0.74 0.53 0.92
PSA -0.27 -0.022 0.64 0.36 -0.12 1 0.56 1 -0.27 0.64 0.36 0.66
ucr-rank 0.12 0.016 -0.11 -0.089 1 -0.12 -0.11 -0.12 0.12 -0.11 -0.063 -0.12
VOTING_PRECINCT -0.49 -0.017 0.58 1 -0.089 0.36 0.53 0.36 -0.49 0.58 0.47 0.57
XBLOCK -0.44 -0.011 1 0.58 -0.11 0.64 0.74 0.64 -0.44 1 0.64 0.76
CCN 0.018 1 -0.011 -0.017 0.016 -0.022 -0.015 -0.021 0.018 -0.011 -0.0016 -0.015
LATITUDE 1 0.018 -0.44 -0.49 0.12 -0.27 -0.55 -0.27 1 -0.44 -0.51 -0.52
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


YEAR=2020
NEIGHBORHOOD_CLUSTER -0.57 -0.0019 0.75 0.64 -0.099 0.66 0.92 0.66 -0.57 0.75 0.57 1
CENSUS_TRACT -0.52 -0.0084 0.67 0.53 -0.077 0.4 0.58 0.4 -0.52 0.67 1 0.57
LONGITUDE -0.46 -0.0044 1 0.58 -0.088 0.62 0.73 0.62 -0.46 1 0.67 0.75
YBLOCK 1 -0.00045 -0.46 -0.52 0.14 -0.34 -0.6 -0.34 1 -0.46 -0.52 -0.57
DISTRICT -0.34 -0.016 0.62 0.42 -0.11 1 0.57 1 -0.34 0.62 0.4 0.66
WARD -0.6 0.0013 0.73 0.6 -0.096 0.57 1 0.57 -0.6 0.73 0.58 0.92
PSA -0.34 -0.016 0.62 0.42 -0.11 1 0.57 1 -0.34 0.62 0.4 0.66
ucr-rank 0.14 0.022 -0.088 -0.078 1 -0.11 -0.096 -0.11 0.14 -0.088 -0.077 -0.099
VOTING_PRECINCT -0.52 -0.0047 0.58 1 -0.078 0.42 0.6 0.42 -0.52 0.58 0.53 0.64
XBLOCK -0.46 -0.0044 1 0.58 -0.088 0.62 0.73 0.62 -0.46 1 0.67 0.75
CCN -0.00045 1 -0.0044 -0.0047 0.022 -0.016 0.0013 -0.016 -0.00045 -0.0044 -0.0084 -0.0019
LATITUDE 1 -0.00045 -0.46 -0.52 0.14 -0.34 -0.6 -0.34 1 -0.46 -0.52 -0.57
LATITUDE
CCN
XBLOCK
VOTING_PRECINCT
ucr-rank
PSA
WARD
DISTRICT
YBLOCK
LONGITUDE
CENSUS_TRACT
NEIGHBORHOOD_CLUSTER
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



*Conclusion: The correlations change between features above are not evident with the time changes.

3.0.3 2-3 Correlation Analysis-2(Task 2-3)

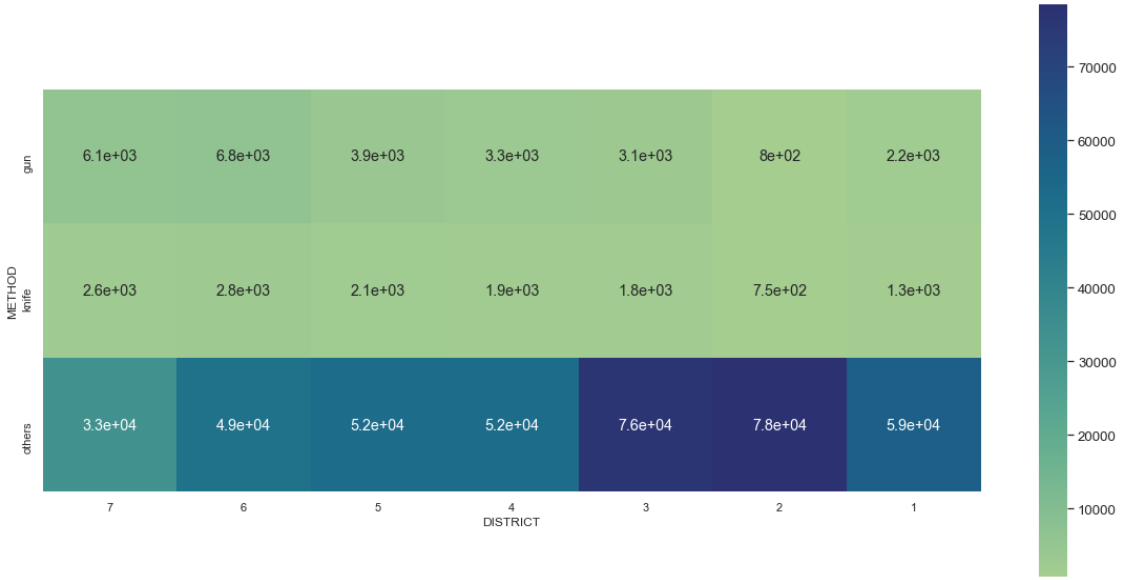
@task:Moreover, you are asked to find the correlation between the crime events and geographic locations. You are also encouraged to analyze the changes of crime events in both time and space.

method & geographic location

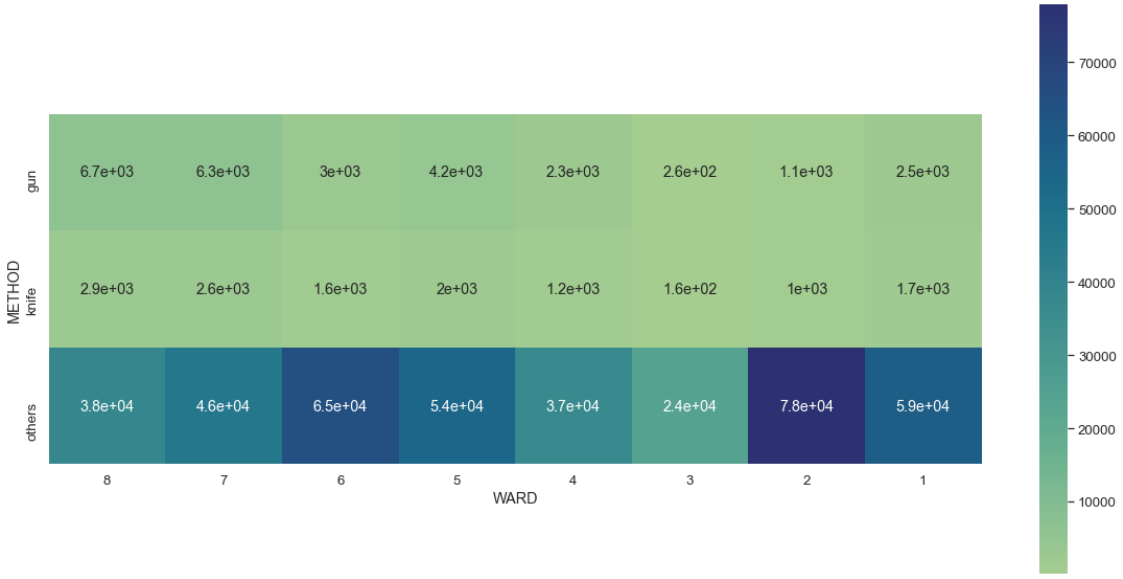
[130]:
```
# define a function to visualize the correlation between crimes and locations
def correl(cross,a,b,font,x,y):
    f, ax=plt.subplots(figsize=(x, y))
    sns.set(font_scale=font)
    sns.heatmap(cross, cmap = 'crest',annot = True,square=True)
    ax.set_ylim([a,0])
```

```
    ax.set_xlim([b,0])
    plt.show()
```

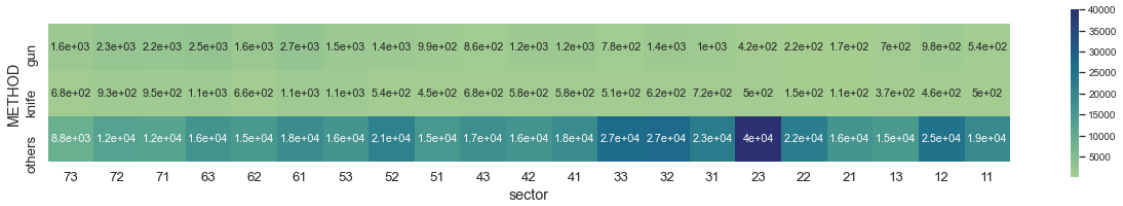
[131]:
```
cross3=pd.crosstab(crime_filtered['METHOD'], crime_filtered['DISTRICT'])
correl(cross3,3,7,1.2,20,10)
#### method and district
```



[132]:
```
cross4=pd.crosstab(crime_filtered['METHOD'], crime_filtered['WARD'])
correl(cross4,3,8,1.2,20,10)
#### method and ward
```

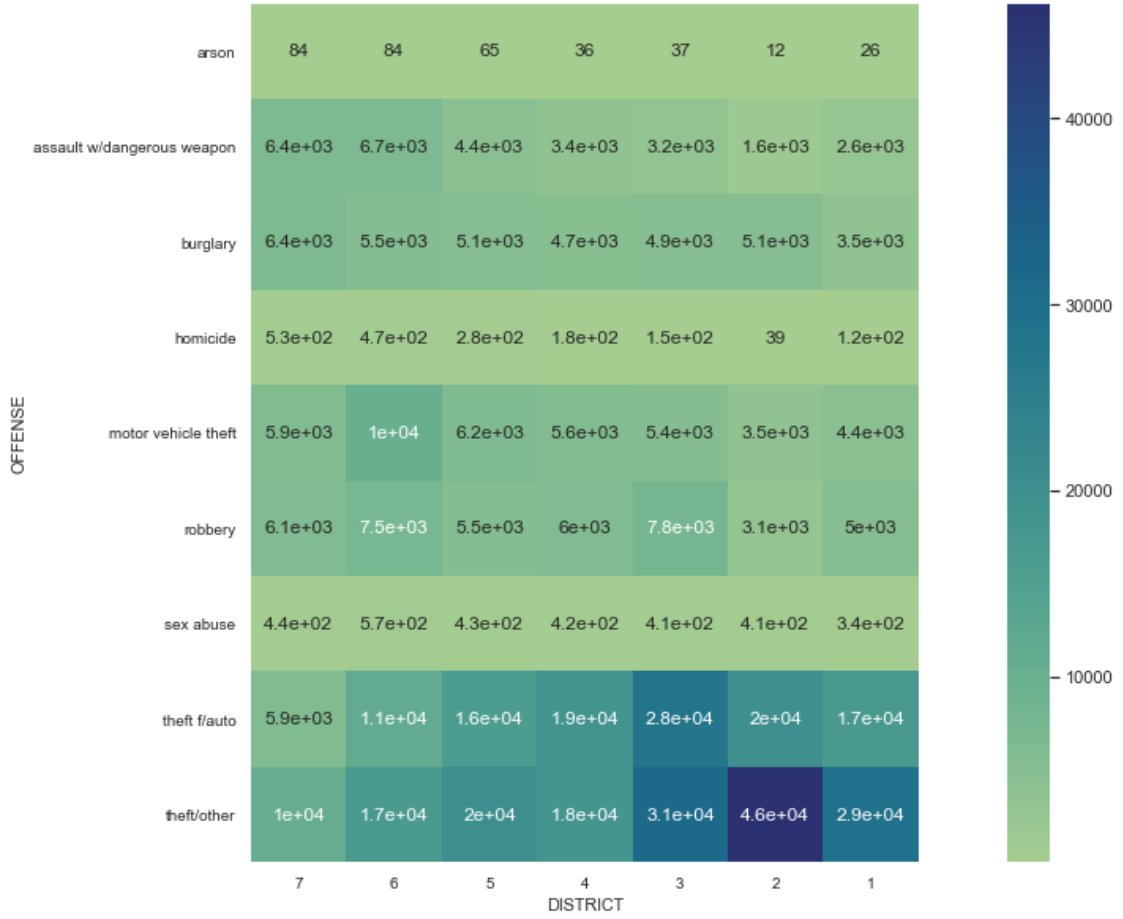


[133]:
```
cross5=pd.crosstab(crime_filtered['METHOD'], crime_filtered['sector'])
correl(cross5,3,21,0.9,21,3)
#### method and sector                sector: 7D3 --- 73
```

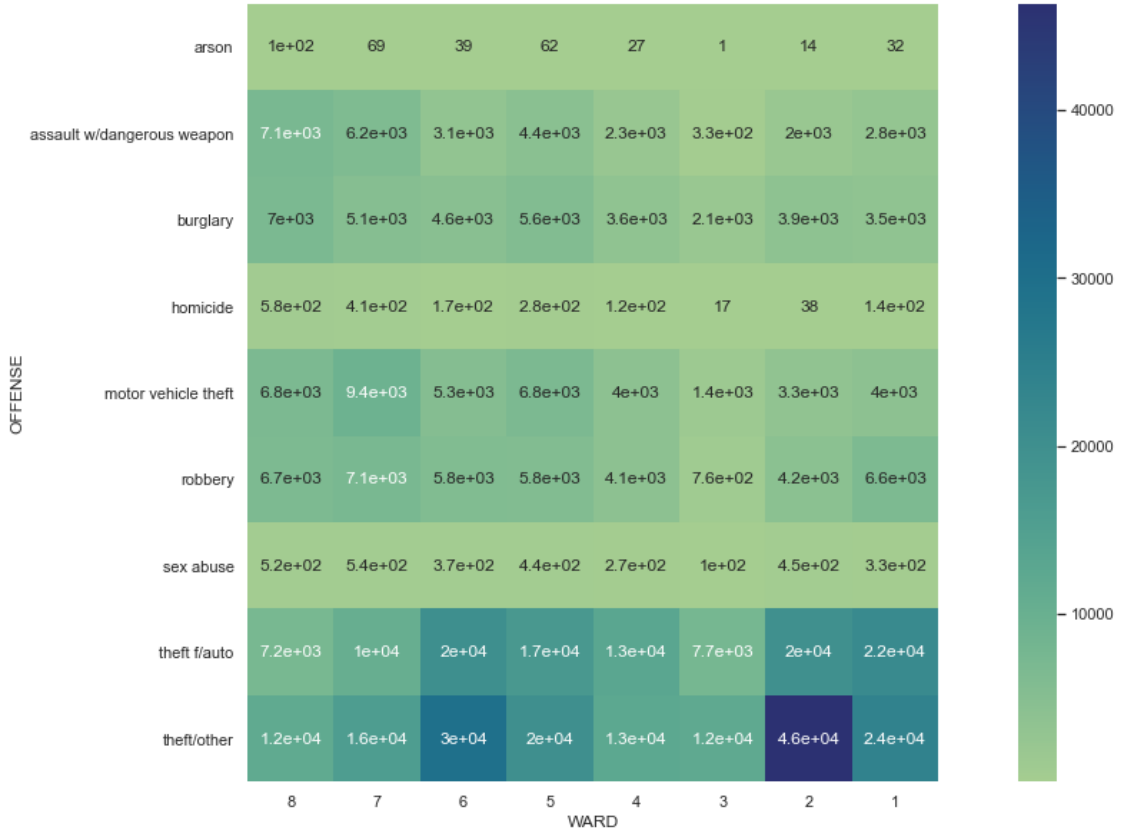


offense & geographic location

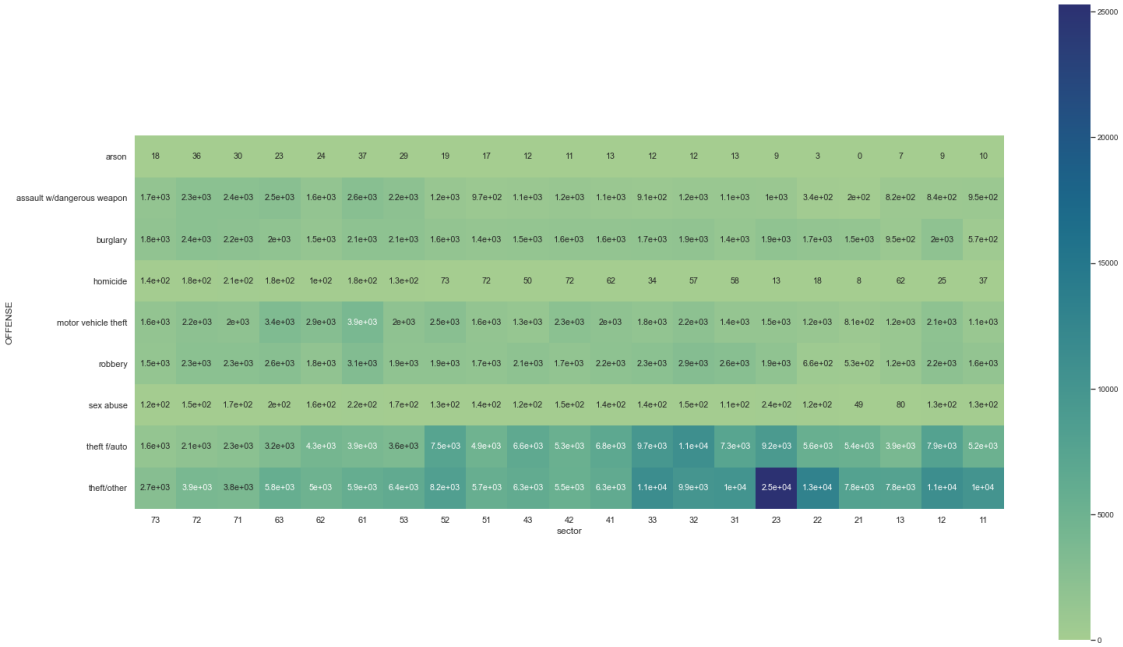
[134]:
```
cross6=pd.crosstab(crime_filtered['OFFENSE'], crime_filtered['DISTRICT'])
correl(cross6,9,7,1,20,10)
#### offense and district
```



[135]:
```
cross7=pd.crosstab(crime_filtered['OFFENSE'], crime_filtered['WARD'])
correl(cross7,9,8,1,20,10)
#### offense and ward
```

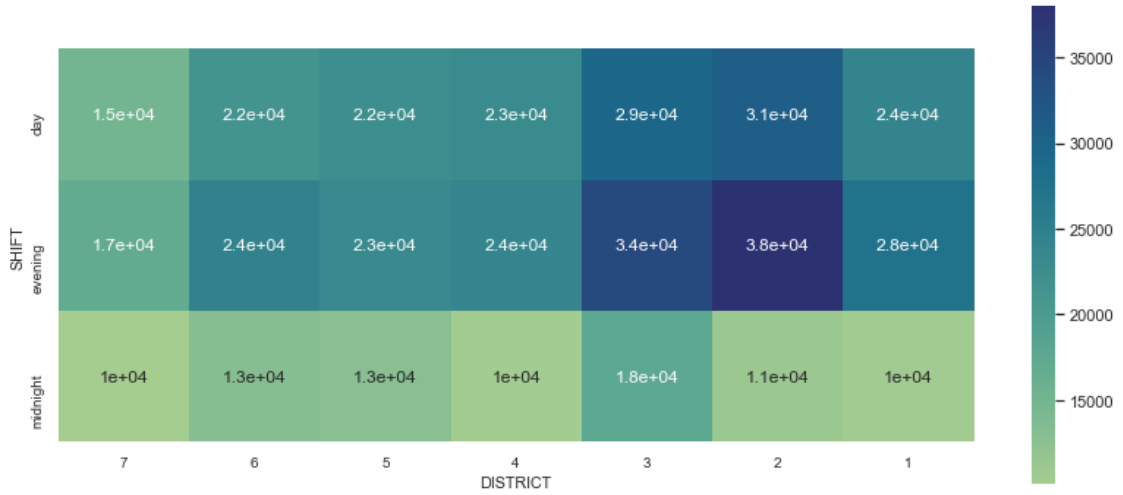


[136]:
```
cross8=pd.crosstab(crime_filtered['OFFENSE'], crime_filtered['sector'])
correl(cross8,9,21,0.9,25,15)
#### offense and sector  / sector: 7D3--->73
```

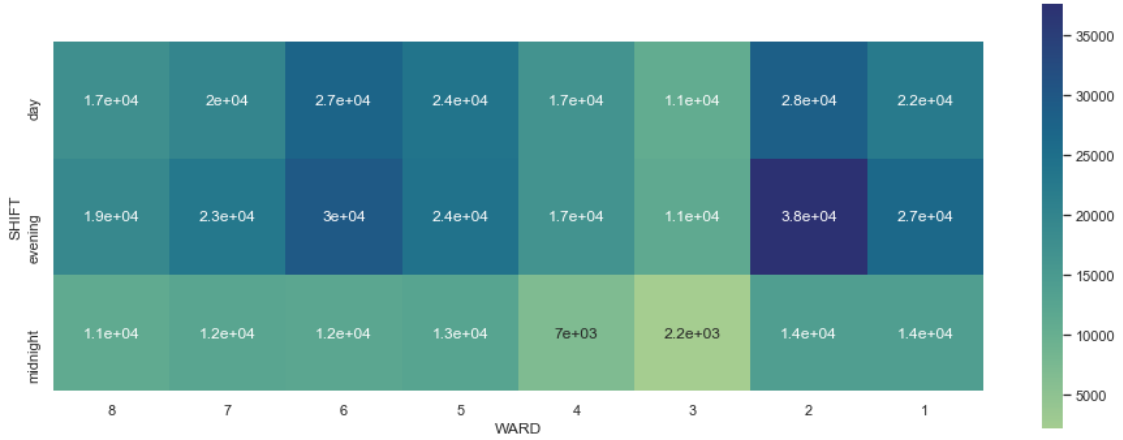


shift & geographic loation

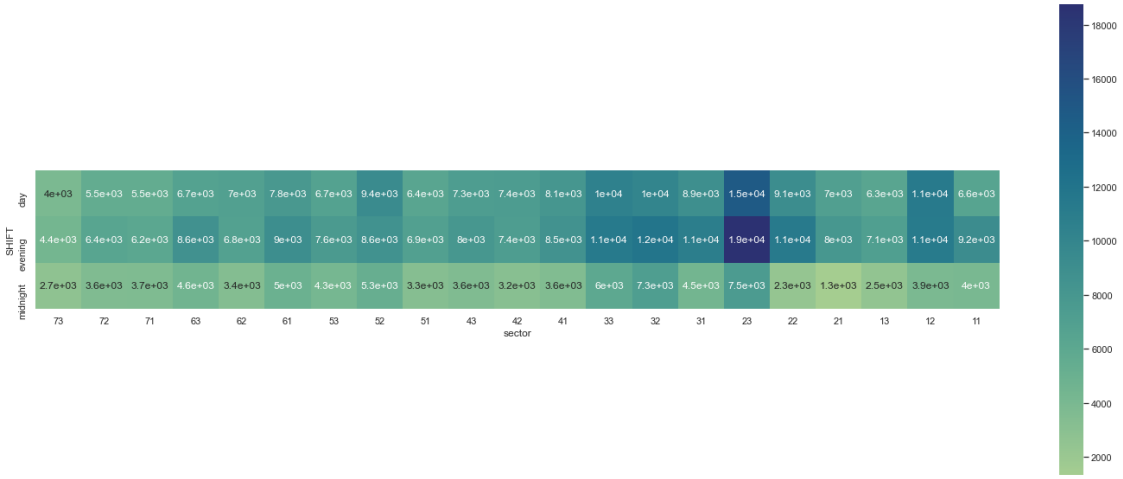
[137]:
```
cross9=pd.crosstab(crime_filtered['SHIFT'], crime_filtered['DISTRICT'])
correl(cross9,3,7,1,14,6)
#### shift and district
```



[138]:
```
cross10=pd.crosstab(crime_filtered['SHIFT'], crime_filtered['WARD'])
correl(cross10,3,8,1,16,6)
#### shift and ward
```

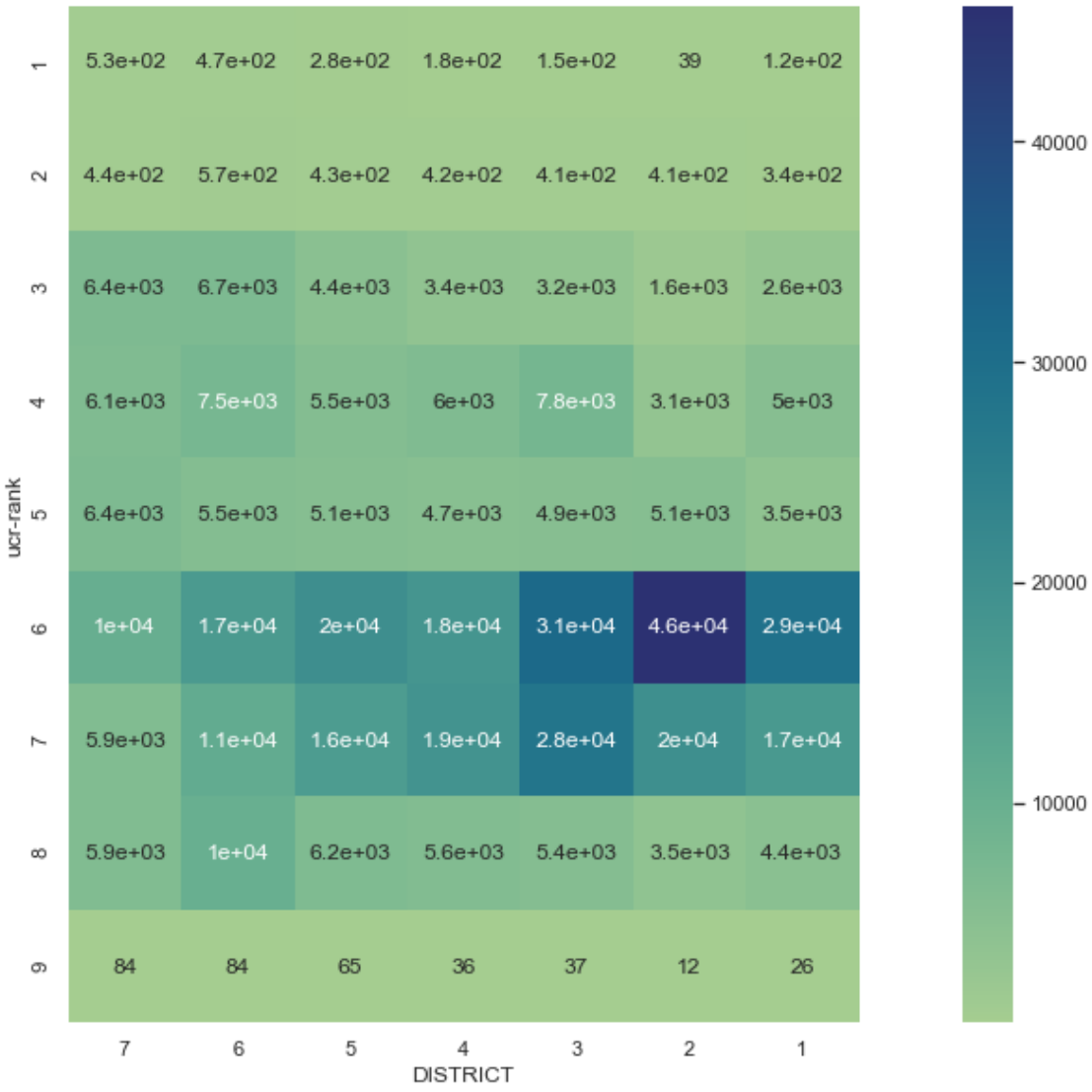


[139]:
```
cross11=pd.crosstab(crime_filtered['SHIFT'], crime_filtered['sector'])
correl(cross11,3,21,1,25,10)
#### shift and sector
```

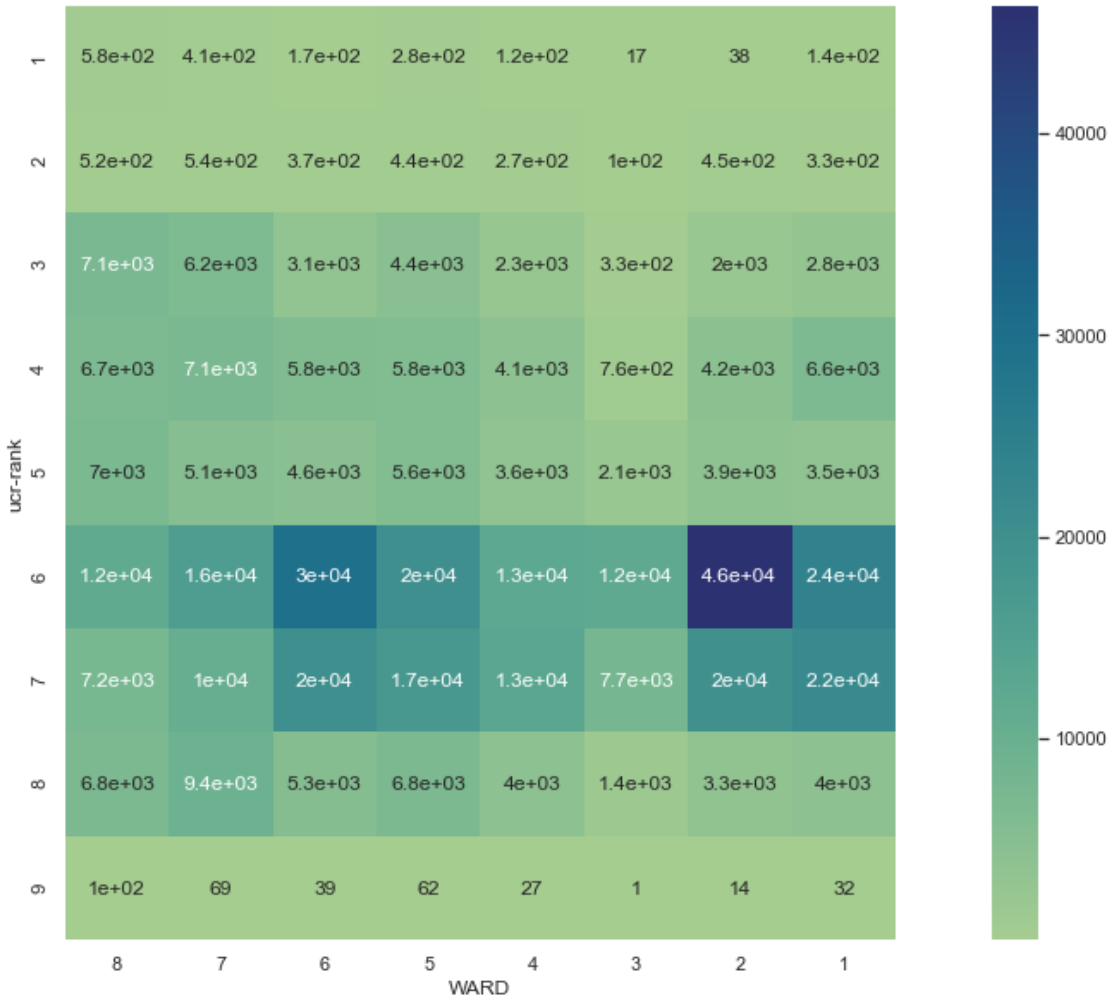


ucr-rank & geographic loation

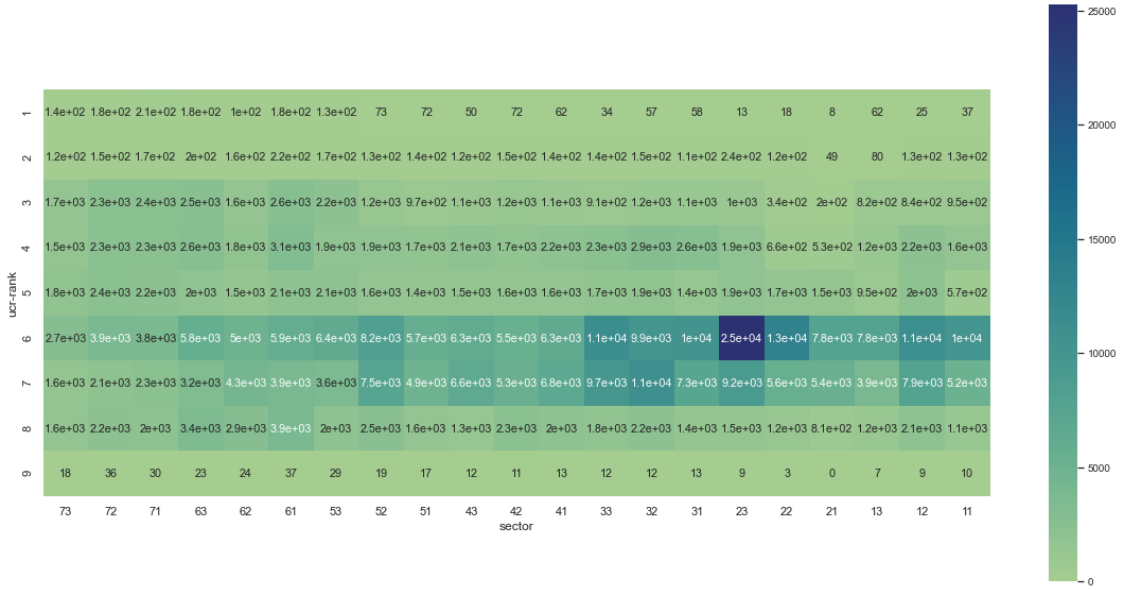
[140]:
```
cross12=pd.crosstab(crime_filtered['ucr-rank'], crime_filtered['DISTRICT'])
correl(cross12,9,7,1,20,10)
#### ucr-rank and district
```



[141]:
```
cross13=pd.crosstab(crime_filtered['ucr-rank'], crime_filtered['WARD'])
correl(cross13,9,8,1,20,10)
#### ucr-rank and ward
```

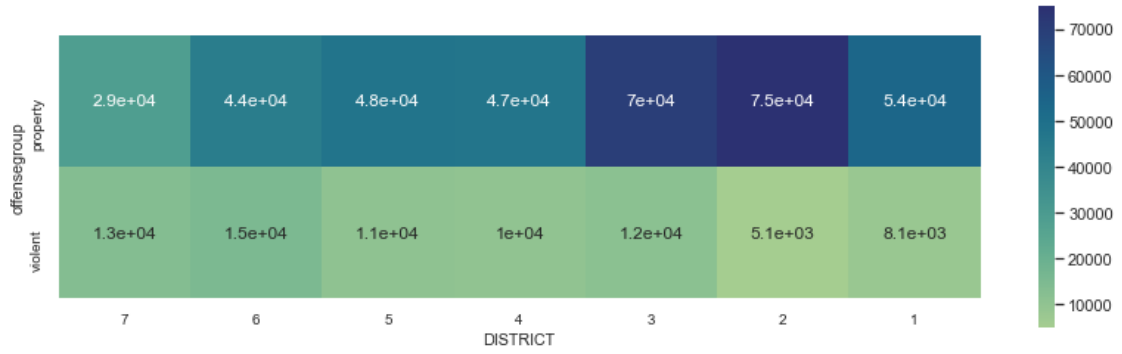


[142]:
```
cross14=pd.crosstab(crime_filtered['ucr-rank'], crime_filtered['sector'])
correl(cross14,9,21,0.9,20,10)
#### ucr-rank and sector
```

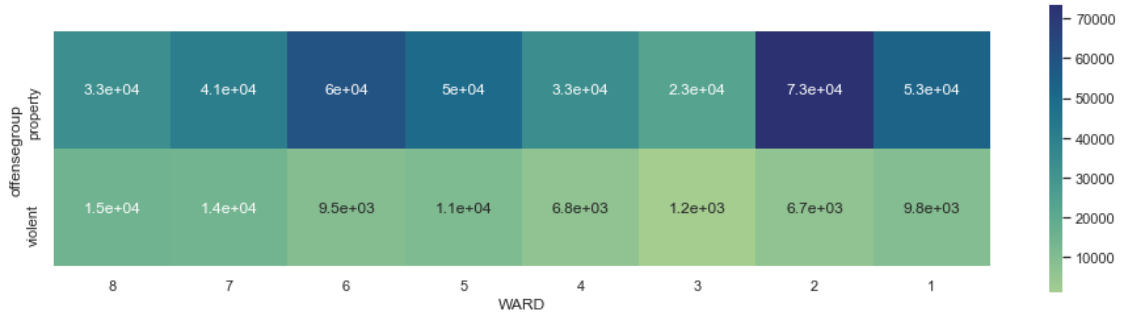


offensegroup & geographic location

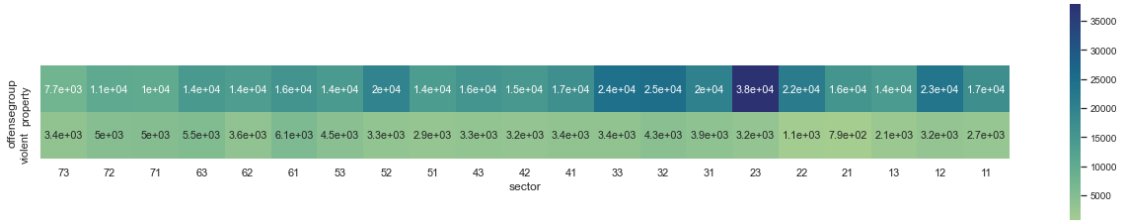
[143]:
```
cross15=pd.crosstab(crime_filtered['offensegroup'], crime_filtered['DISTRICT'])
correl(cross15,2,7,1,14,4)
#### offensegroup and district
```



[144]:
```
cross16=pd.crosstab(crime_filtered['offensegroup'], crime_filtered['WARD'])
correl(cross16,2,8,1,16,4)
#### offensegroup and ward
```



[145]:
```
cross17=pd.crosstab(crime_filtered['offensegroup'], crime_filtered['sector'])
correl(cross17,2,21,0.9,22,4)
#### offensegroup and sector
```

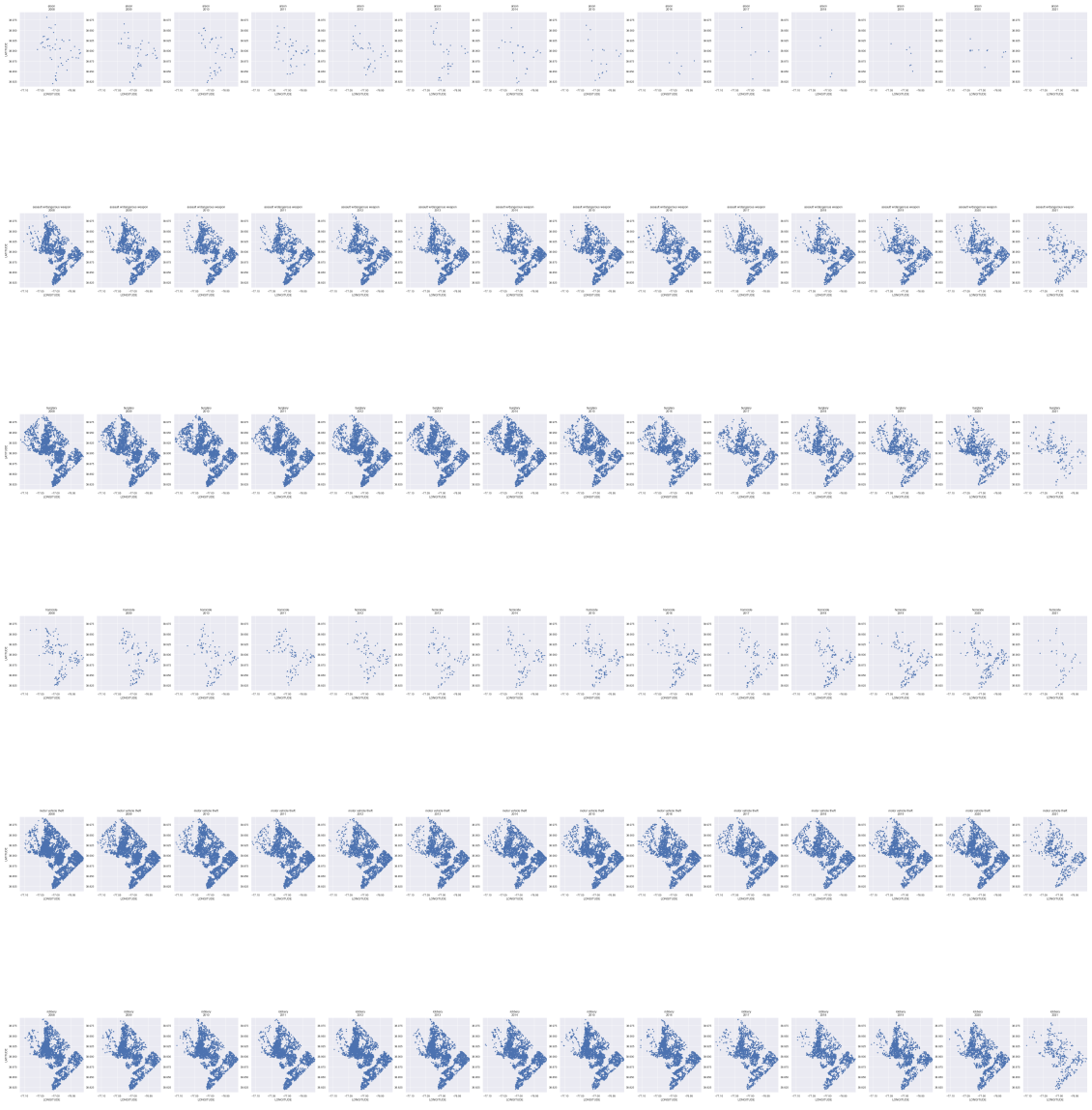


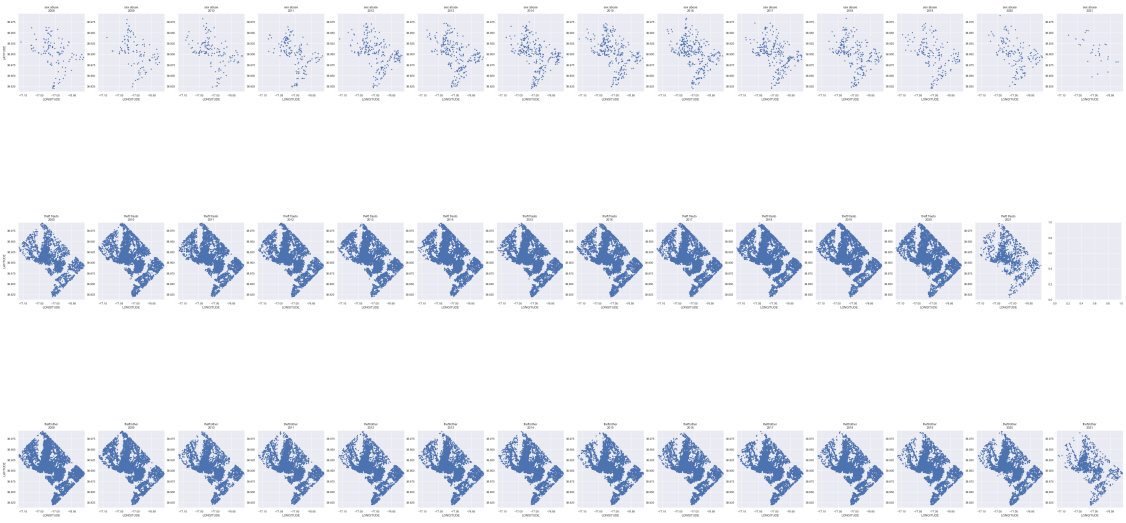
The changes of crime events with different type in time

[146]:
```
# define a function to plot the location changes by year
def changes(data,name):
    data=crime_filtered[data]
    Groups = crime_filtered.groupby(data[name])
    Groups = dict(list(Groups))
    Keys = list(Groups.keys())
    for key in Keys:
        i = 0
        Date_Groups = Groups[key].groupby(Groups[key].YEAR)
        Date_Groups = dict(list(Date_Groups))
        Date_Keys = list(Date_Groups.keys())
        f, ax = plt.subplots(1,14)
        f.set_figheight(5)
        f.set_figwidth(70)
        for dkeys in Date_Keys:
            ax[i].scatter(Date_Groups[dkeys].LONGITUDE, Date_Groups[dkeys].
↪LATITUDE, marker = '.')
            ax[i].set_ylim(38.813,38.995)
            ax[i].set_xlim(-77.115,-76.91)
            f.subplots_adjust(hspace = 3)
```

```
            s = str(key) + "\n" + str(dkeys)
            ax[i].set_title(s)
            ax[0].set_ylabel('LATITUDE')
            ax[i].set_xlabel('LONGITUDE')
            f.subplots_adjust(hspace = 3)
            i=i+1
        plt.show()
```

[147]:
```
changes(['LATITUDE','LONGITUDE','OFFENSE'],'OFFENSE')
# each row is a kind of OFFENE; location changes by year from left to right
```





*Conclusion: geography changes are not evident. Amount of each type crime is decreasing by time.

*Other analysis for the changes of crime events in time are in 2-4. And some results are in the map file.

3.0.4 2-4 Other Analysis(Task 2-4)

@task:How does the number of crimes vary geographically and temporally? As a warm-up for the next task, we suggest you to visualize the number of crimes according to the geographical districts. You can first divide the DC area into several disjoint subareas according to either administrative district or other types of geographical districts defined by yourselves. Then you may count the number of crimes in each subarea in a certain time period and plot them on the geographical map (e.g., you may plot heatmap). Please also show your plot for some different time, seasons, years, etc.

Visualize the number of crimes according to districts.

[148]:
```
plt.figure(figsize=(20, 20))
sns.set(font_scale=2)
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==1],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==1], s=50, alpha=0.3,␣
 ↪color=[0,0,0], lw=0, label='district 1')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==2],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==2], s=50, alpha=0.3,␣
 ↪color=[0,0.3,0.7], lw=0, label='district 2')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==3],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==3], s=50, alpha=0.3,␣
 ↪color=[0,0.7,0.3], lw=0, label='district 3')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==4],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==4], s=50, alpha=0.3,␣
 ↪color=[0.3,0,0.7], lw=0, label='district 4')
```

```
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==5],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==5], s=50, alpha=0.3,␣
 ↪color=[0.7,0,0.3], lw=0, label='district 5')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==6],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==6], s=50, alpha=0.3,␣
 ↪color=[0.7,0.3,0], lw=0, label='district 6')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['DISTRICT']==7],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['DISTRICT']==7], s=50, alpha=0.3,␣
 ↪color=[0.3,0.7,0], lw=0, label='district 7')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
```

[148]: <Figure size 1440x1440 with 0 Axes>

[148]: <matplotlib.collections.PathCollection at 0x2c402401520>

[148]: <matplotlib.collections.PathCollection at 0x2c5169e13a0>

[148]: <matplotlib.collections.PathCollection at 0x2c55fd6ab20>

[148]: <matplotlib.collections.PathCollection at 0x2c402babc40>

[148]: <matplotlib.collections.PathCollection at 0x2c55fd6a250>

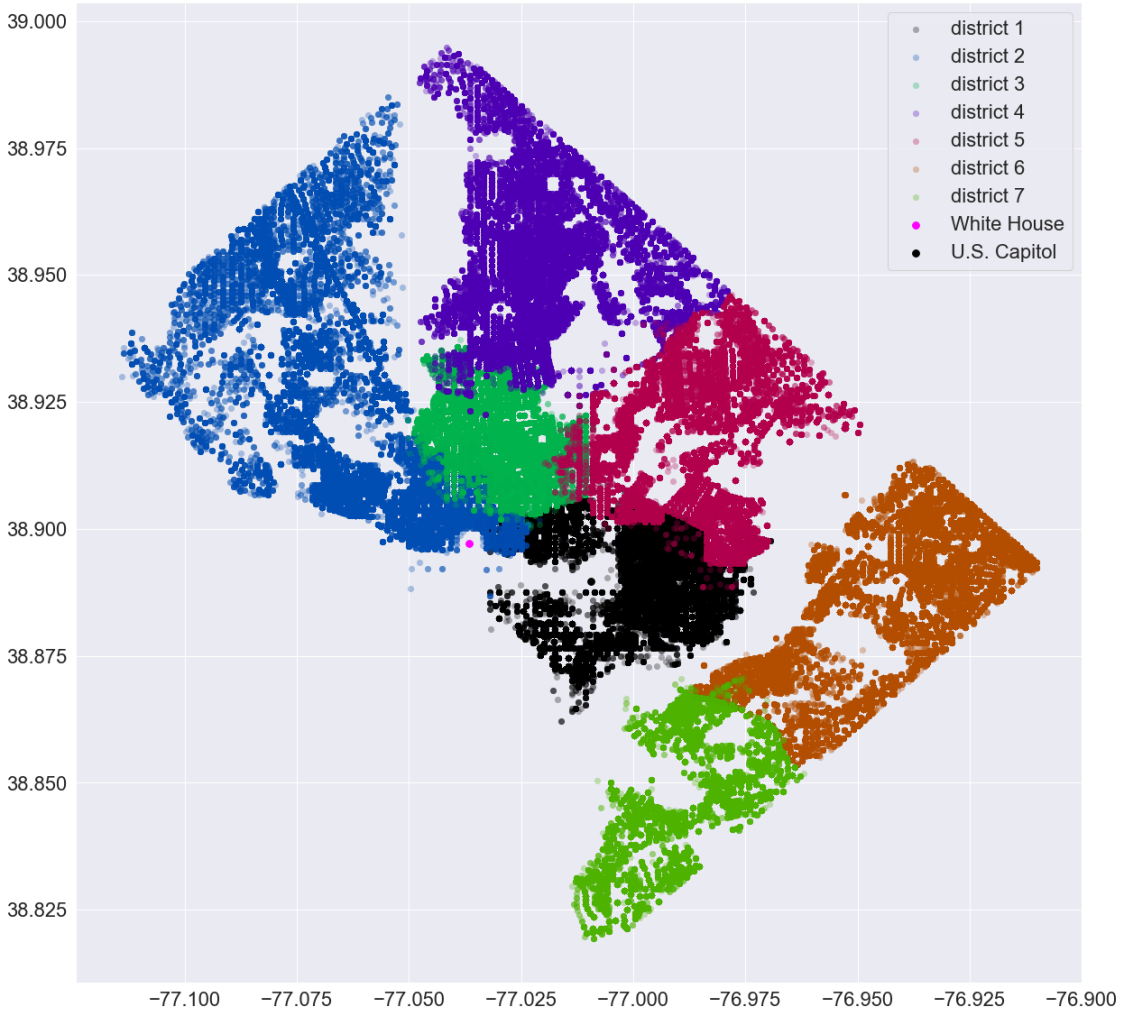
[148]: <matplotlib.collections.PathCollection at 0x2c402401970>

[148]: <matplotlib.collections.PathCollection at 0x2c40995a6d0>

[148]: <matplotlib.collections.PathCollection at 0x2c4691b3910>

[148]: <matplotlib.collections.PathCollection at 0x2c402401d60>

[148]: <matplotlib.legend.Legend at 0x2c4024014f0>



Visualize the number of crimes according to wards.

[149]:
```
plt.figure(figsize=(20, 20))
sns.set(font_scale=2)
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==1],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==1], s=50, alpha=0.3,␣
 ↪color=[0,0,0], lw=0, label='ward 1')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==2],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==2], s=50, alpha=0.3,␣
 ↪color=[0,0.3,0.7], lw=0, label='ward 2')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==3],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==3], s=50, alpha=0.3,␣
 ↪color=[0,0.7,0.3], lw=0, label='ward 3')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==4],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==4], s=50, alpha=0.3,␣
 ↪color=[0.3,0,0.7], lw=0, label='ward 4')
```

```
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==5],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==5], s=50, alpha=0.3,␣
 ↪color=[0.7,0,0.3], lw=0, label='ward 5')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==6],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==6], s=50, alpha=0.3,␣
 ↪color=[0.7,0.3,0], lw=0, label='ward 6')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==7],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==7], s=50, alpha=0.3,␣
 ↪color=[0.3,0.7,0], lw=0, label='ward 7')
plt.scatter(crime_filtered['LONGITUDE'][crime_filtered['WARD']==8],␣
 ↪crime_filtered['LATITUDE'][crime_filtered['WARD']==8], s=50, alpha=0.3,␣
 ↪color=[1,0,0], lw=0, label='ward 8')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
```

[149]: <Figure size 1440x1440 with 0 Axes>

[149]: <matplotlib.collections.PathCollection at 0x2c3bf292ac0>

[149]: <matplotlib.collections.PathCollection at 0x2c503c127f0>

[149]: <matplotlib.collections.PathCollection at 0x2c503bed610>

[149]: <matplotlib.collections.PathCollection at 0x2c3bf2923a0>

[149]: <matplotlib.collections.PathCollection at 0x2c3bf451040>

[149]: <matplotlib.collections.PathCollection at 0x2c503beda90>

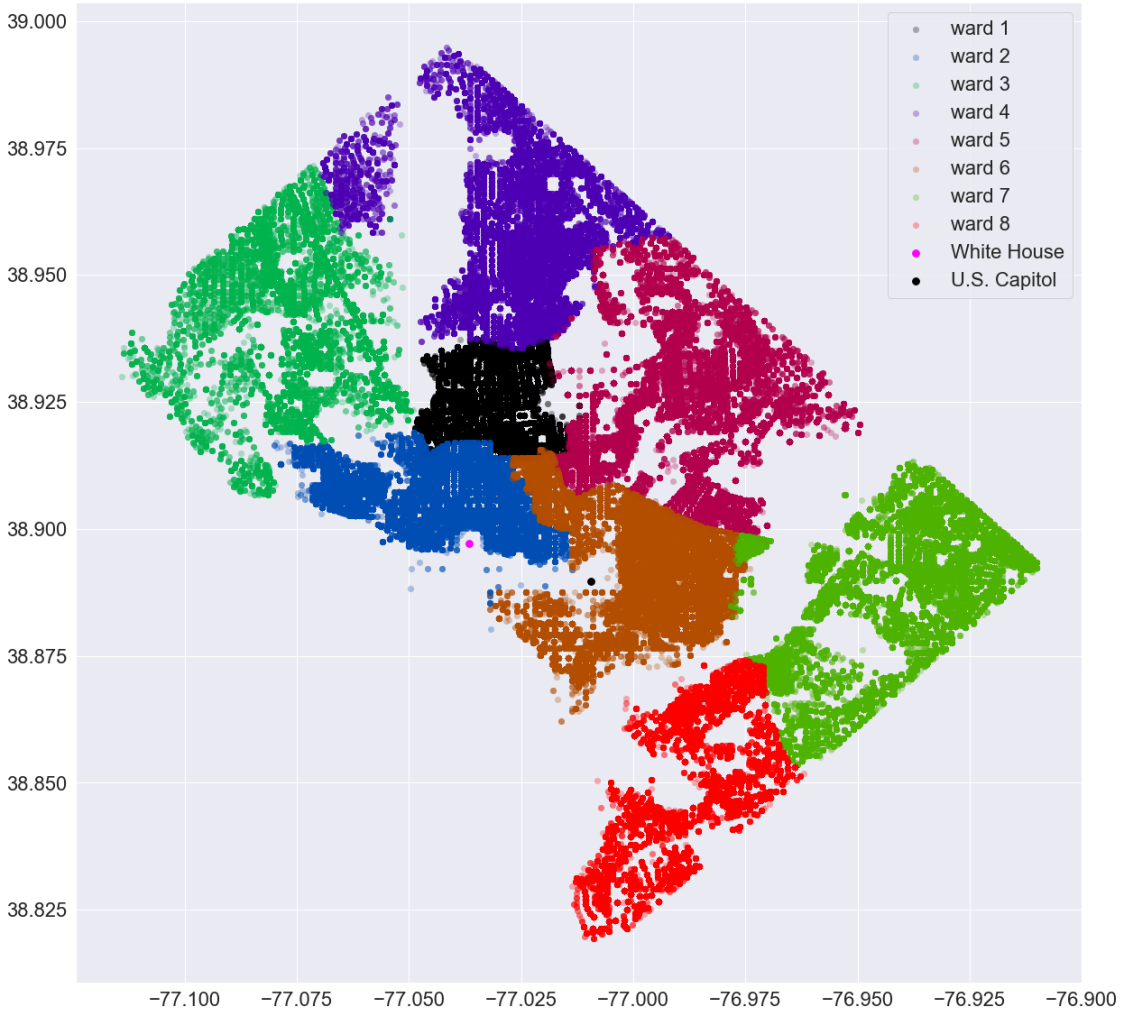
[149]: <matplotlib.collections.PathCollection at 0x2c402461160>

[149]: <matplotlib.collections.PathCollection at 0x2c3bf292880>

[149]: <matplotlib.collections.PathCollection at 0x2c4022e1a60>

[149]: <matplotlib.collections.PathCollection at 0x2c3bf292190>

[149]: <matplotlib.legend.Legend at 0x2c402432dc0>



*Following results are in the map file.

Geographical map(general).

[150]:
```
draw_heatmap(crime_filtered,'general')
```

Conditional geographical heatmap.

[151]:
```
# define a function to draw geographical heatmap by district, ward, year,␣
 ↪month, etc.
def heatmapby(cat):
    if cat == 'DISTRICT':
        for i in range(1,8):
            temp = crime_filtered[crime_filtered[cat]==i]
            draw_heatmap(temp,cat.lower()+str(i))
    if cat == 'WARD':
        for i in range(1,9):
```

```
            temp = crime_filtered[crime_filtered[cat]==i]
            draw_heatmap(temp,cat.lower()+str(i))
    if cat == 'YEAR':
        for i in range(2008,2022):
            temp = crime_filtered[crime_filtered[cat]==i]
            draw_heatmap(temp,str(i))
    if cat == 'MONTH':
        for i in range(1,13):
            temp = crime_filtered[crime_filtered_newfeature[cat]==i]
            draw_heatmap(temp,'month_'+str(i))
```

[152]:
```
# draw heatmaps
heatmapby('DISTRICT')
heatmapby('WARD')
heatmapby('YEAR')
heatmapby('MONTH')
```

[153]:
```
# geographical heatmap by season
a=crime_filtered[(crime_filtered_newfeature.MONTH>2)&(crime_filtered_newfeature.
 ↪MONTH<=5)]
b=crime_filtered[(crime_filtered_newfeature.MONTH>5)&(crime_filtered_newfeature.
 ↪MONTH<=8)]
c=crime_filtered[(crime_filtered_newfeature.MONTH>8)&(crime_filtered_newfeature.
 ↪MONTH<=11)]
d=crime_filtered[(crime_filtered_newfeature.
 ↪MONTH==1)|(crime_filtered_newfeature.MONTH==2)
                 |(crime_filtered_newfeature.MONTH==12)]
draw_heatmap(a,'season_spring')
draw_heatmap(b,'season_summer')
draw_heatmap(c,'season_autumn')
draw_heatmap(d,'season_winnter')
```

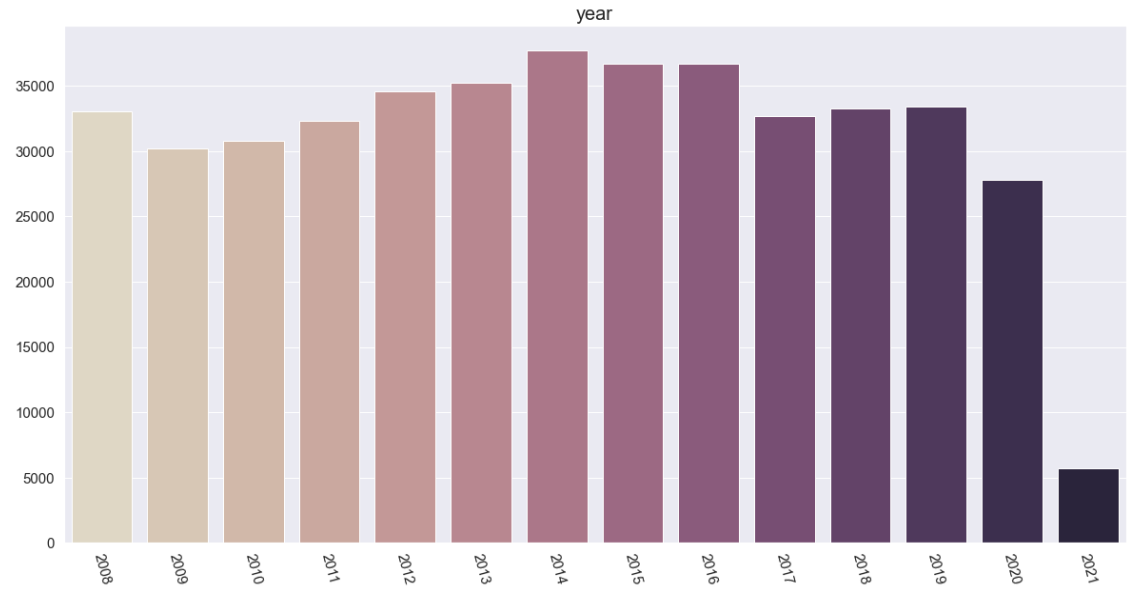
[154]:
```
# season data
a['season']='spring'
b['season']='summer'
c['season']='autumn'
d['season']='winter'
season_data=pd.concat([a,b,c,d],axis=0)
```

Plots for some different time, seasons, years, etc.

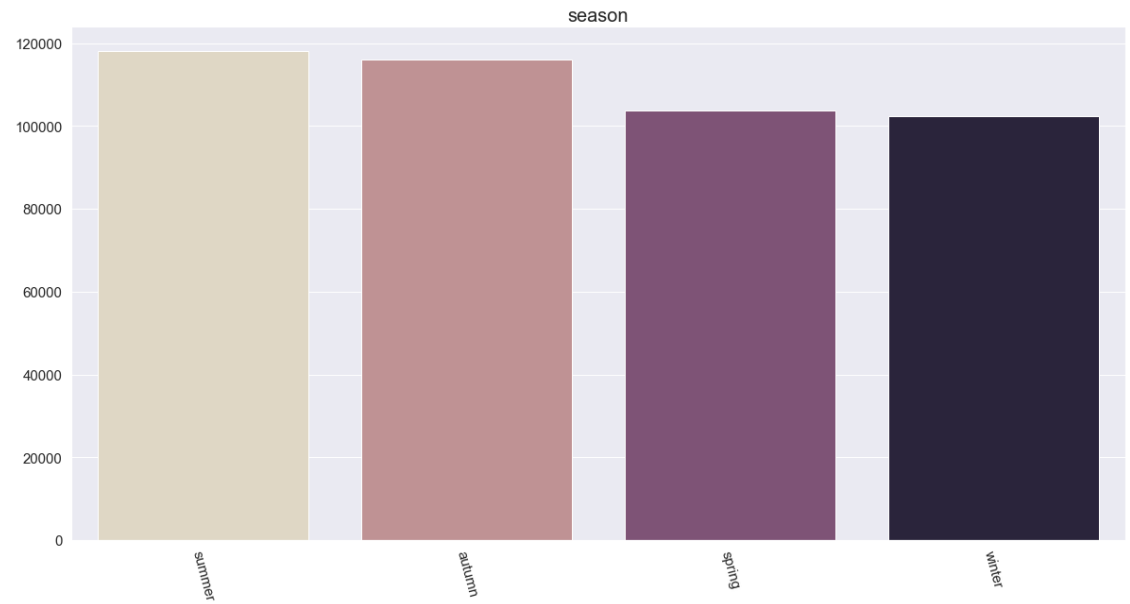
[155]:
```
yearplot=pd.DataFrame(crime_filtered['YEAR'].value_counts()).reset_index()
seasonplot=pd.DataFrame(season_data['season'].value_counts()).reset_index()
monthplot=pd.DataFrame(crime_filtered_newfeature['MONTH'].value_counts()).
 ↪reset_index()
weekplot=pd.DataFrame(crime_filtered_newfeature['week'].value_counts()).
 ↪reset_index()
```

```
hourplot=pd.DataFrame(crime_filtered_newfeature['hour'].value_counts()).
↪reset_index()
```

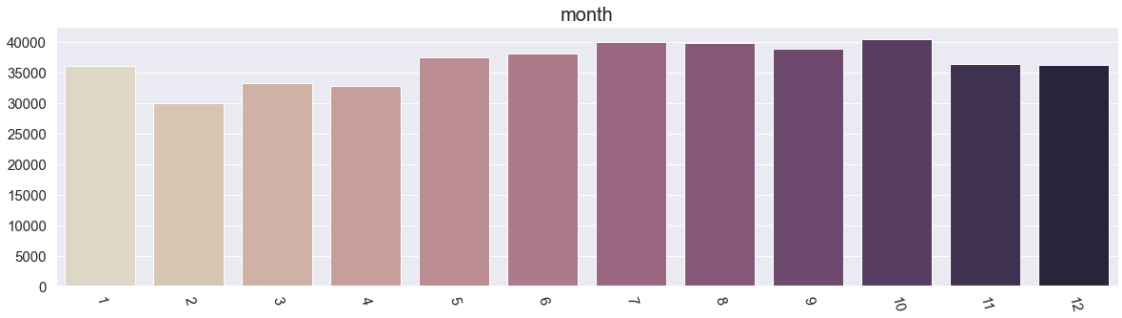
[156]:
```
bar_plot(yearplot,'year',20,15,75,20,10,"ch:s=-.2,r=.6",0)
bar_plot(seasonplot,'season',20,15,75,20,10,"ch:s=-.2,r=.6",0)
bar_plot(monthplot,'month',20,15,75,20,5,"ch:s=-.2,r=.6",0)
bar_plot(weekplot,'week',20,15,75,20,5,"ch:s=-.2,r=.6",0)
bar_plot(hourplot,'hour',20,15,75,20,5,"ch:s=-.2,r=.6",0)
```



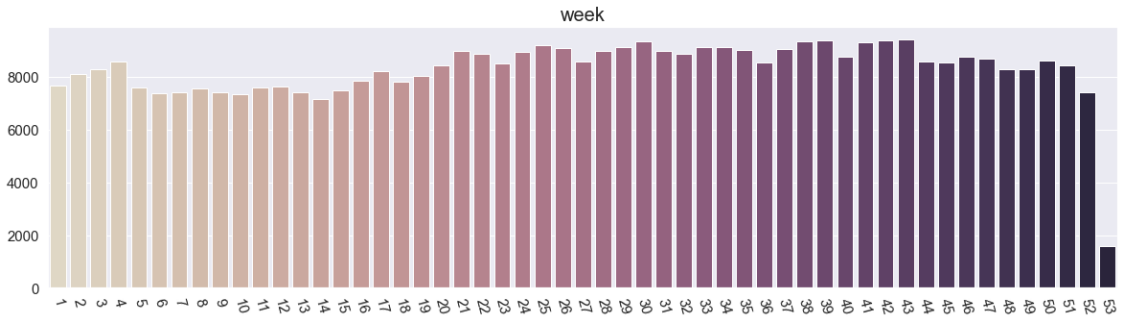
[156]: ()



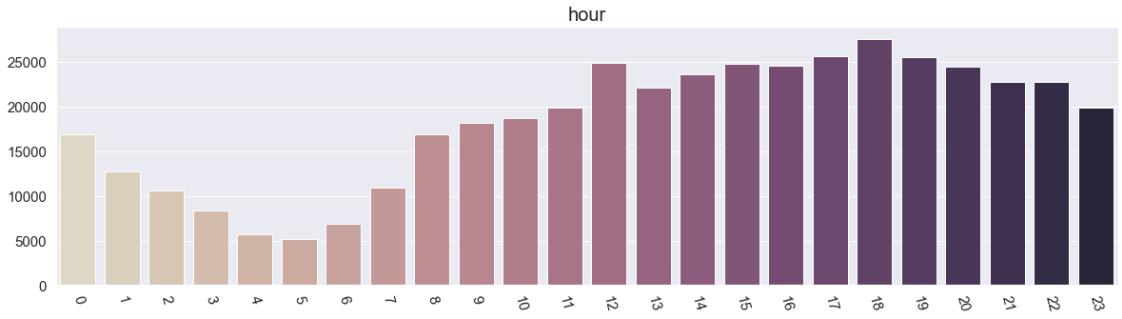
[156]: ()



[156]: ()



[156]: ()

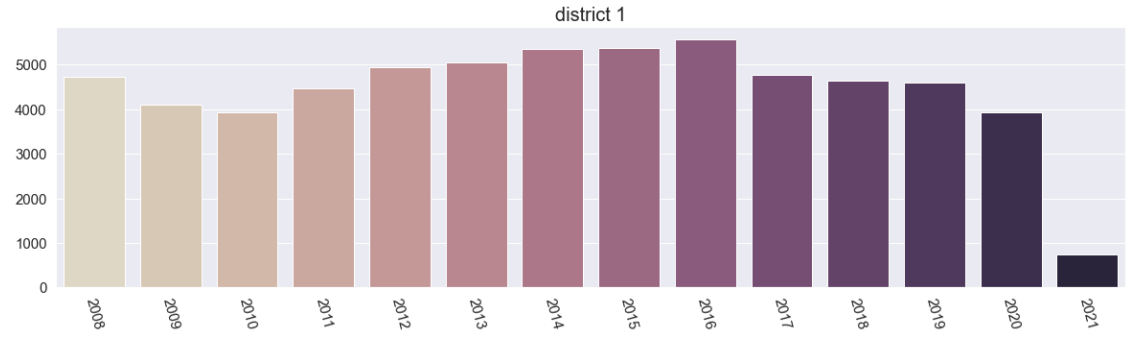


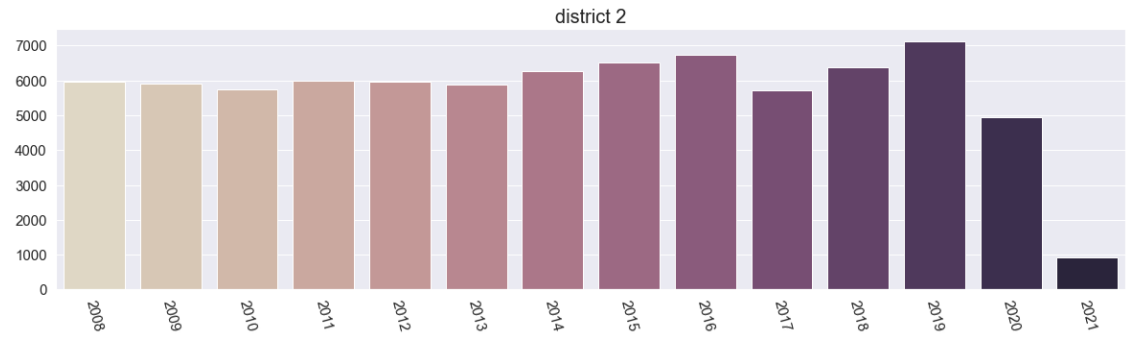
[156]: ()

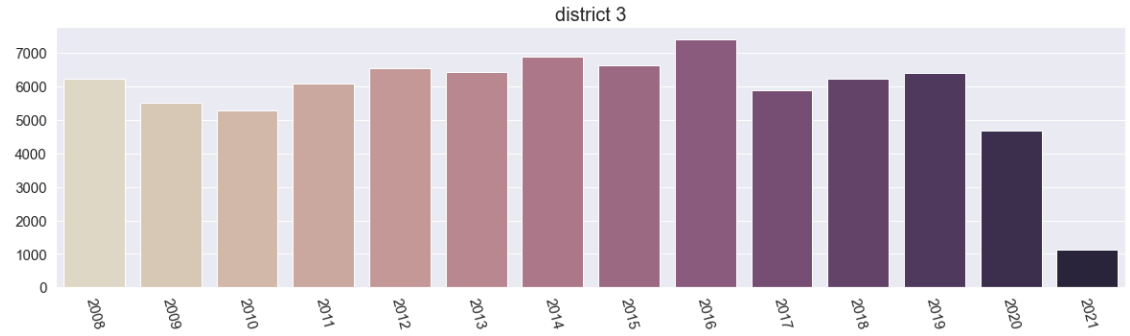
*Conclusion: crimes had greater proportions in summer and autumn. 12 o'clock, 17 o'clock, 18 o'clock and 19 o'clock are the peak periods for crime.

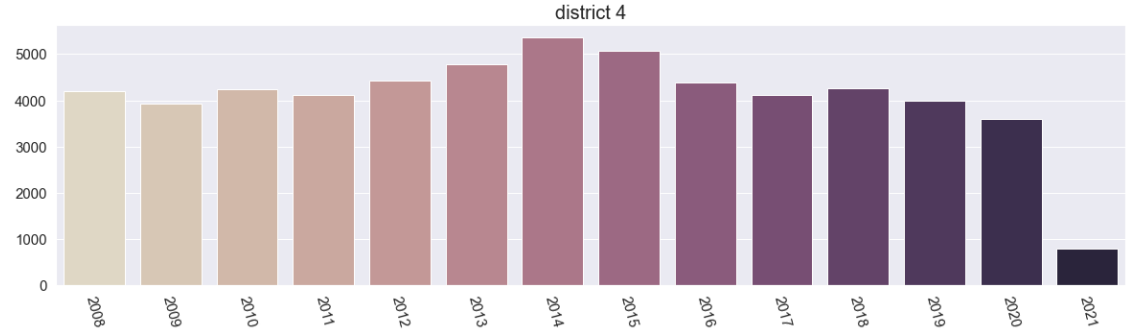
[157]:
```
# define a function to plot the number of crimes in each district by time
def districtplot(time):
    for i in range(1,8):
        templot = pd.
 ↪DataFrame(crime_filtered_newfeature[crime_filtered_newfeature.
 ↪DISTRICT==i][time].value_counts()).reset_index()
        bar_plot(templot,'district '+str(i),20,15,75,20,5,"ch:s=-.2,r=.6",0)

# define a function to plot the number of crimes in each ward by time
def wardplot(time):
    for i in range(1,9):
        templot = pd.
 ↪DataFrame(crime_filtered_newfeature[crime_filtered_newfeature.WARD==i][time].
 ↪value_counts()).reset_index()
        bar_plot(templot,'ward '+str(i),20,15,75,20,5,"ch:s=-.2,r=.6",0)
```

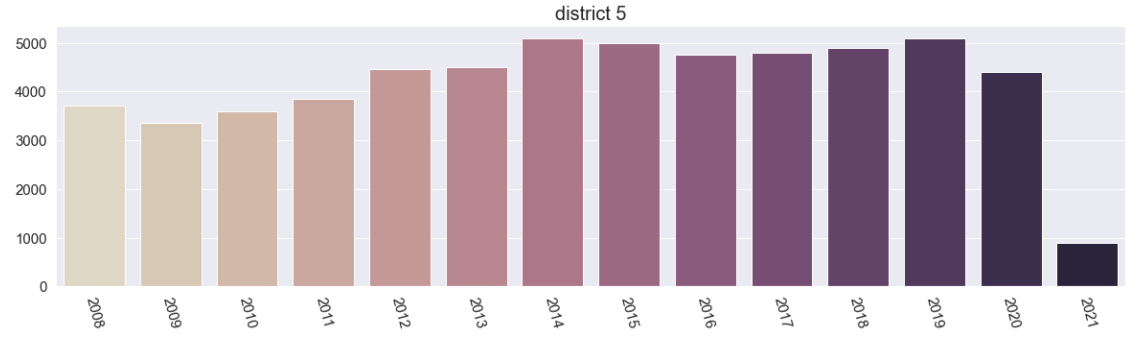
[158]:
```
districtplot('YEAR')
wardplot('YEAR')
# number of crimes in district and ward by year
```

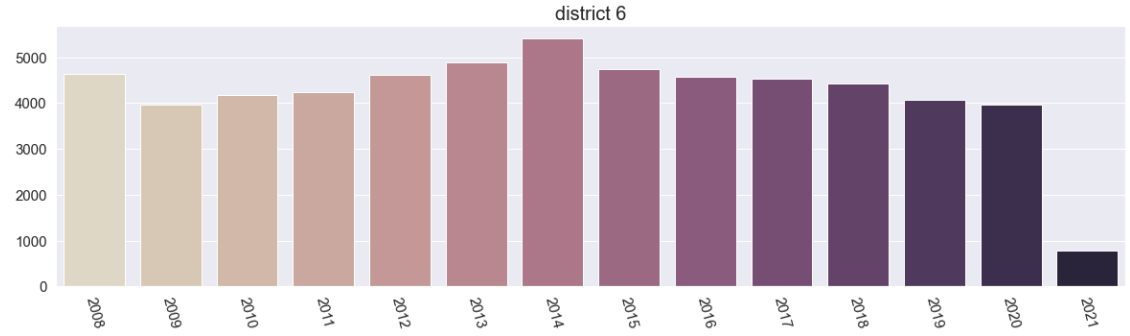


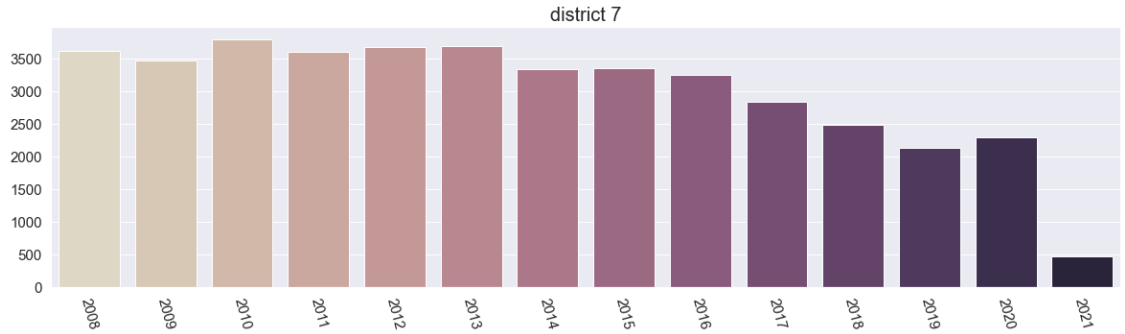


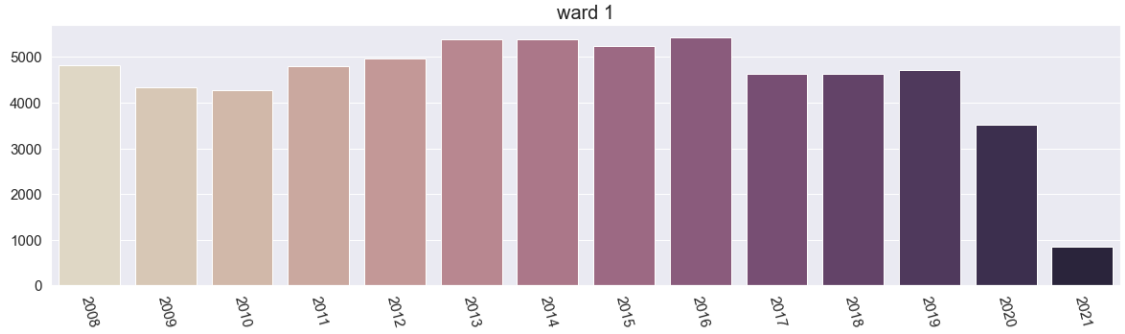

district 3
7000
6000
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

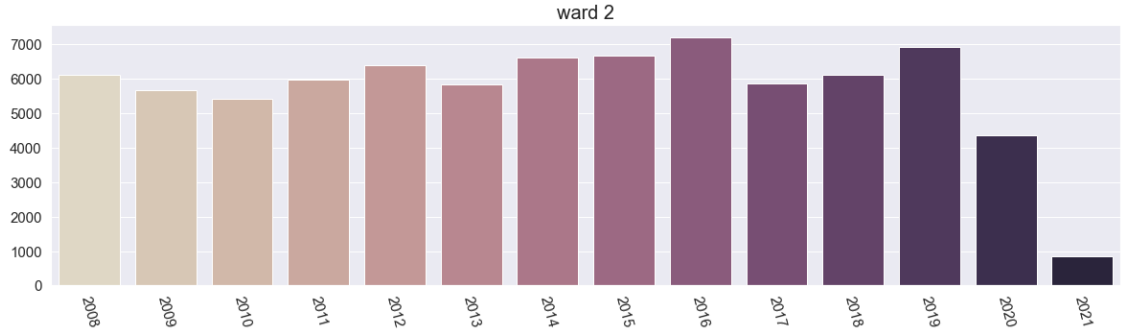

district 4
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

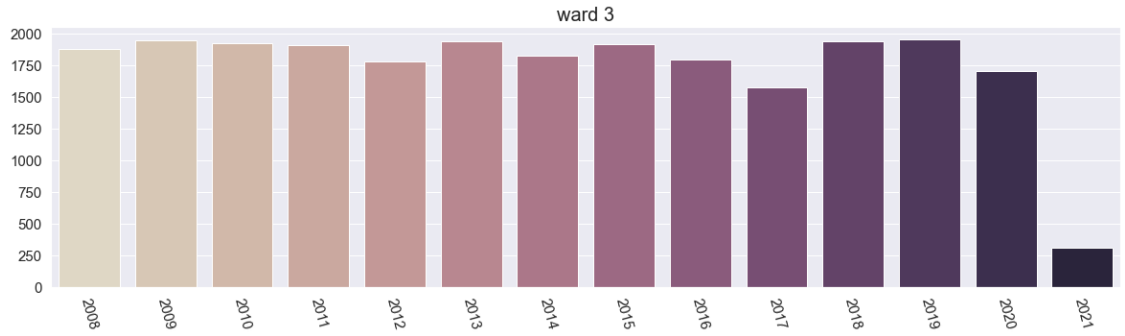

district 5
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

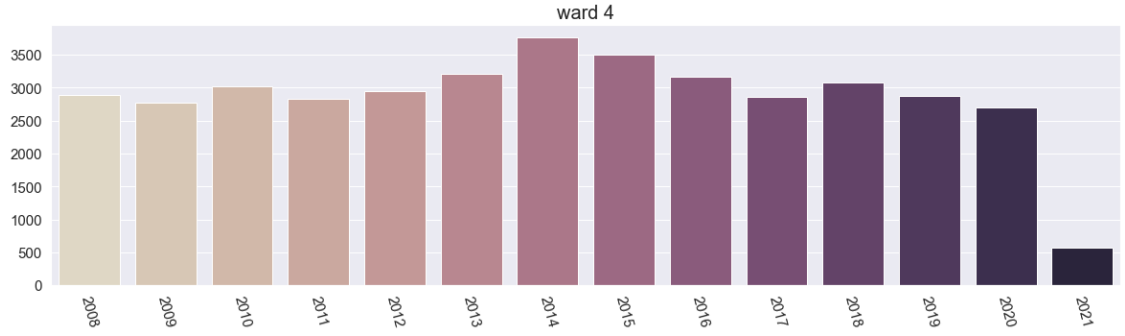

district 6
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

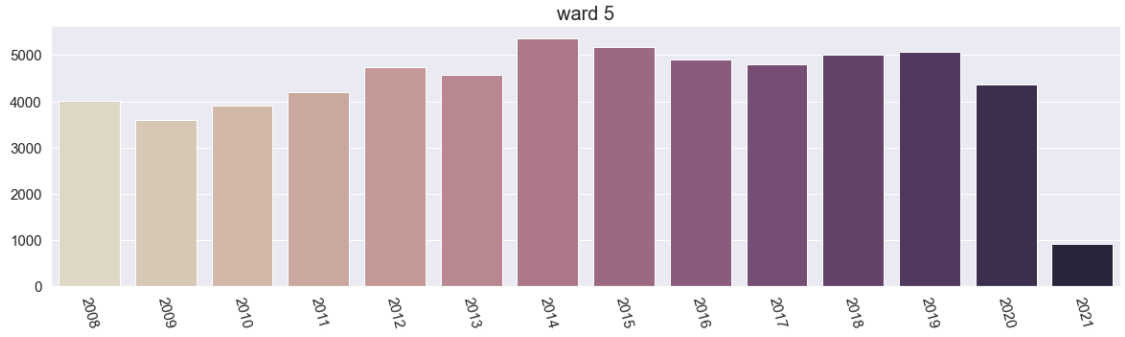

district 7
3500
3000
2500
2000
1500
1000
500
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

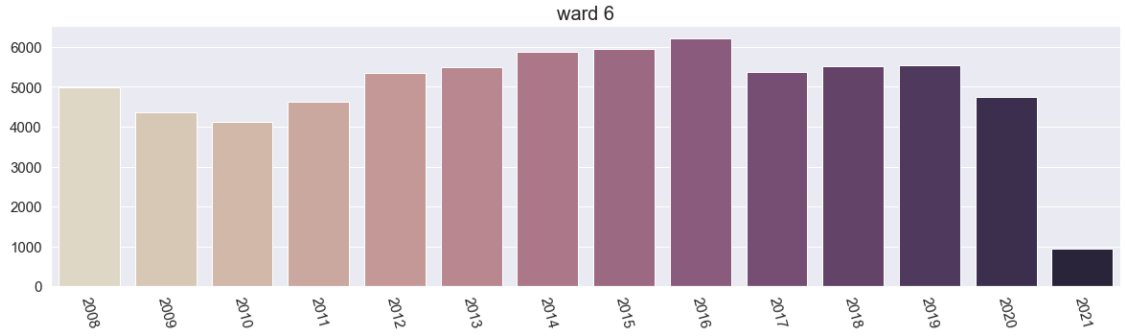

ward 1
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

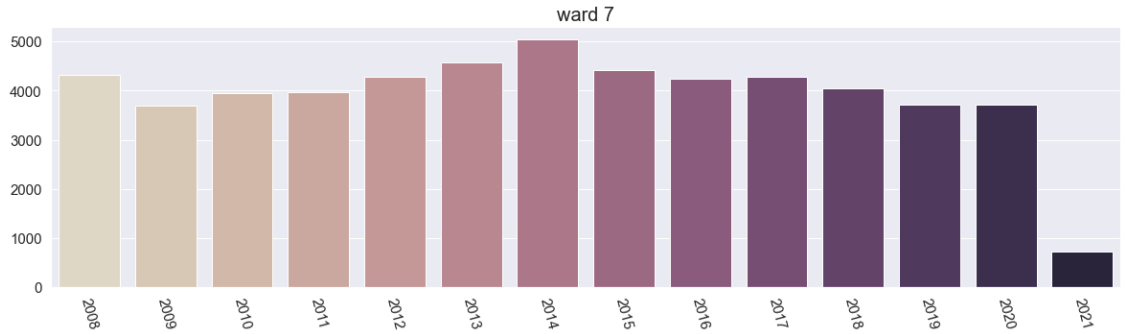

ward 2
7000
6000
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

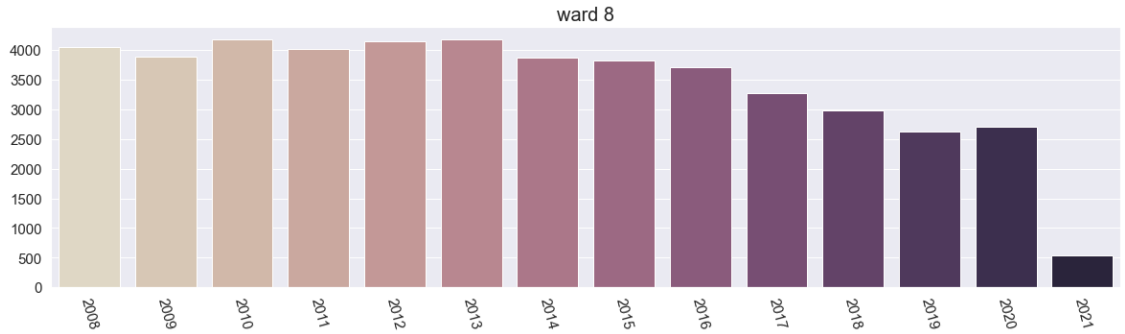

ward 3
2000
1750
1500
1250
1000
750
500
250
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


ward 4
3500
3000
2500
2000
1500
1000
500
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

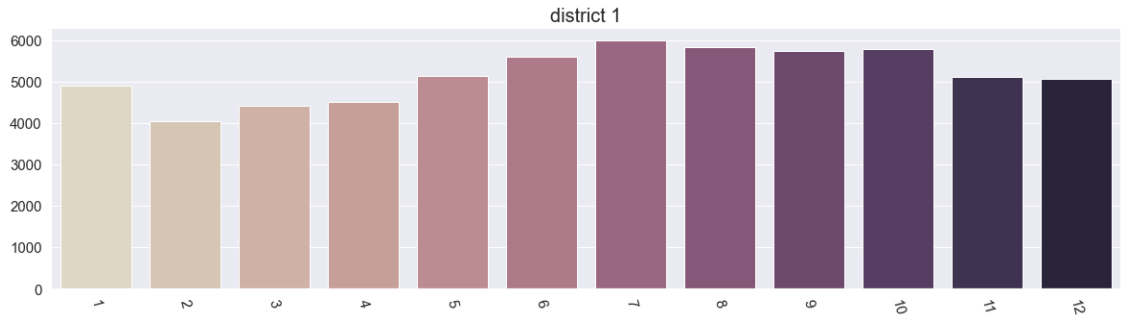

ward 5
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

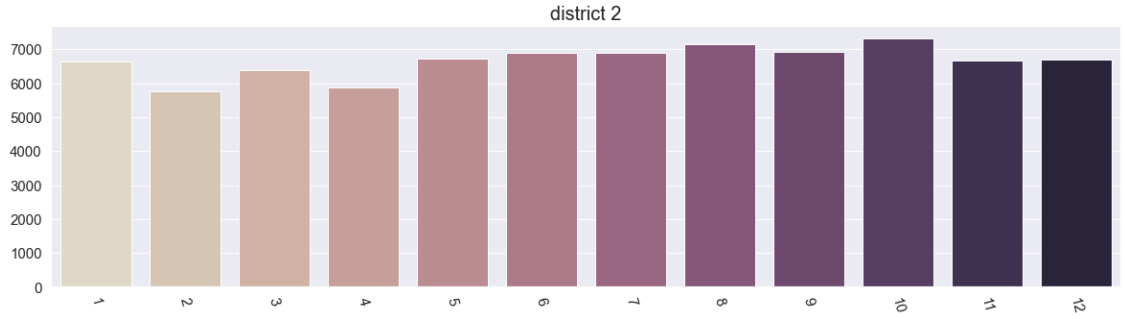

ward 6
6000
5000
4000
3000
2000
1000
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

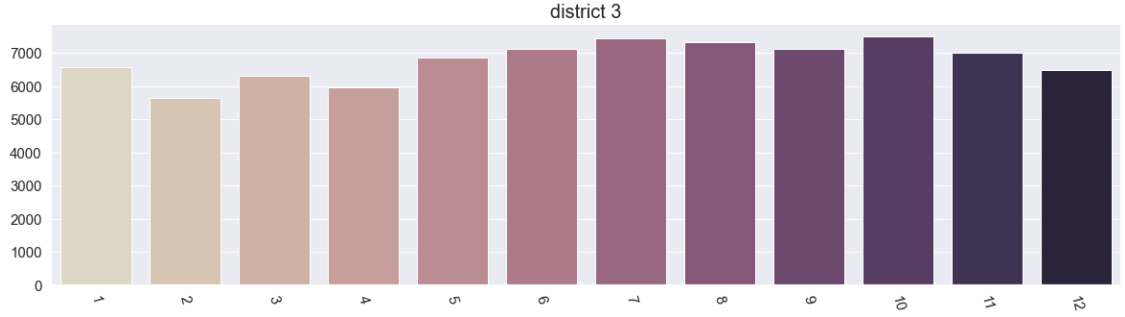


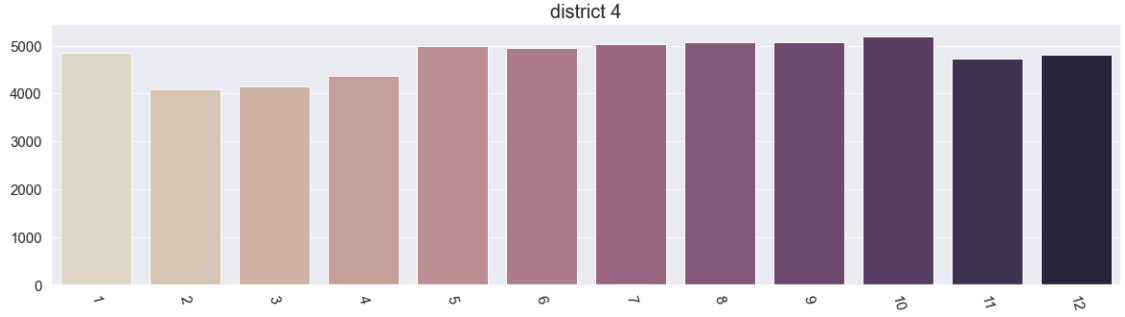


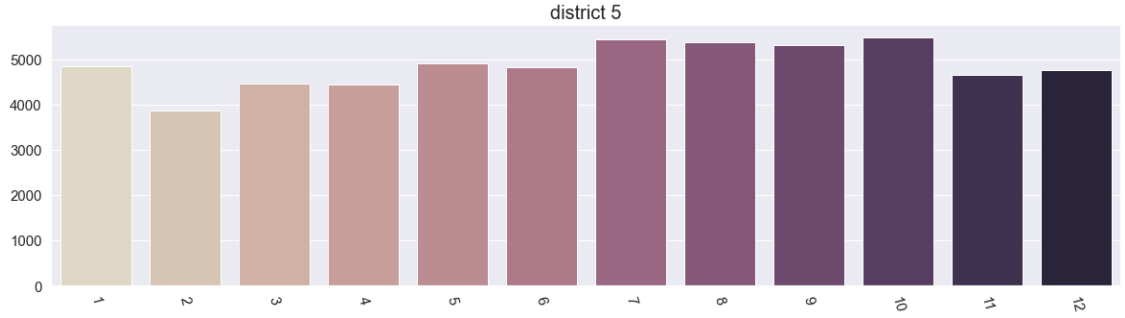
[159]:
```
districtplot('MONTH')
wardplot('MONTH')
# number of crimes in district and ward by month
```

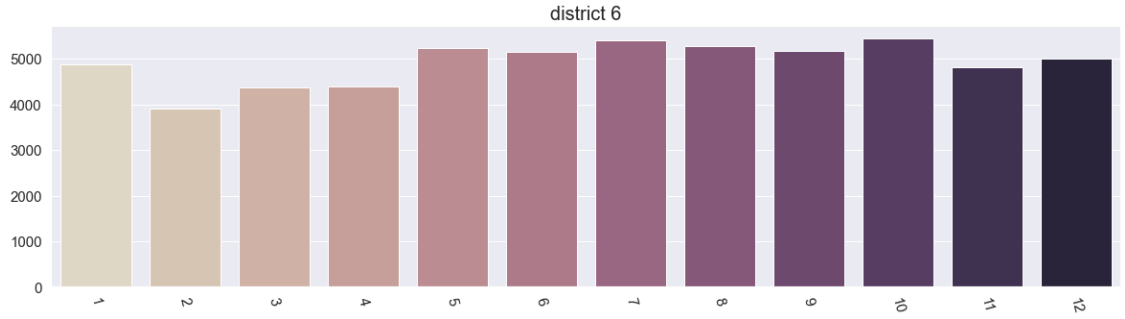


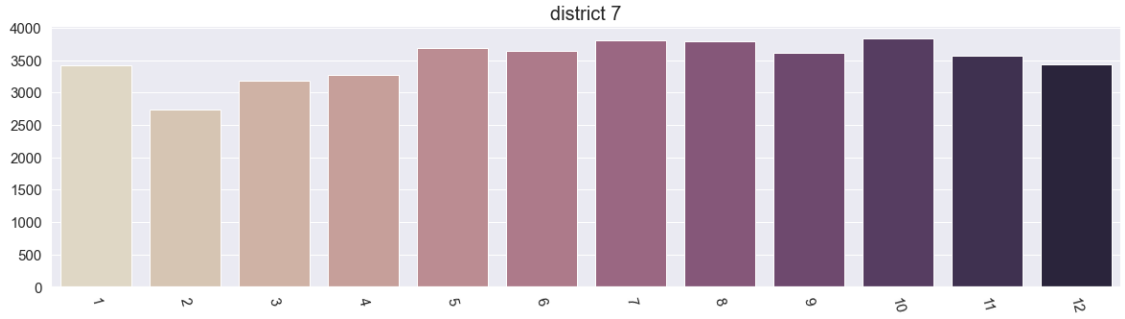

district 2
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

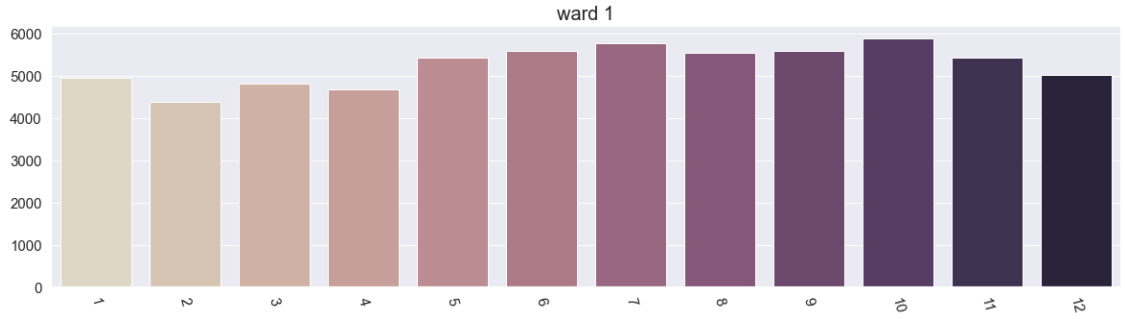

district 3
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

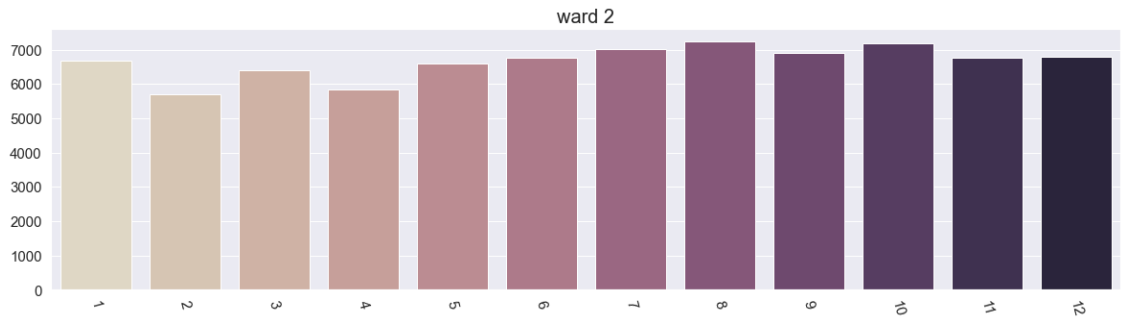

district 4
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

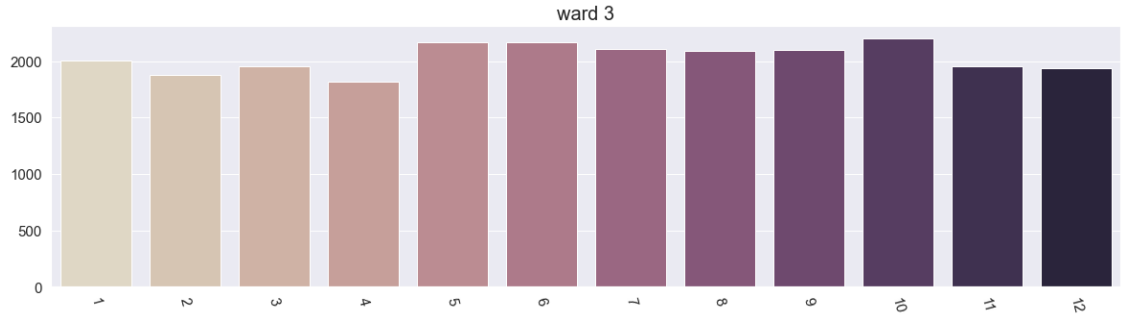

district 5
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

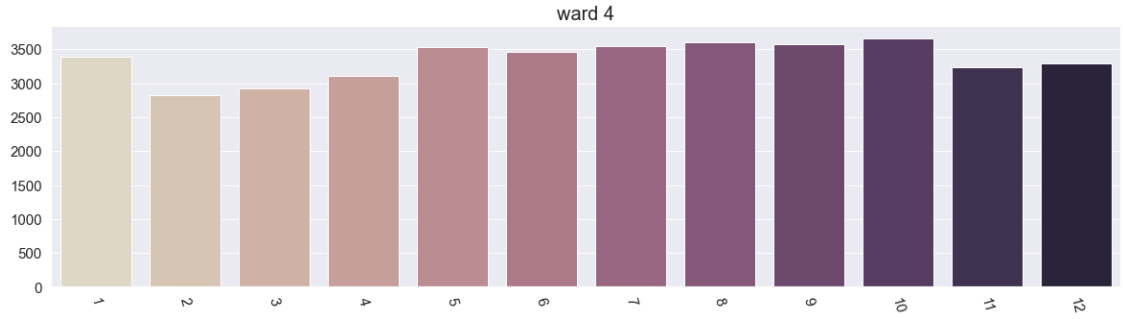

district 6
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

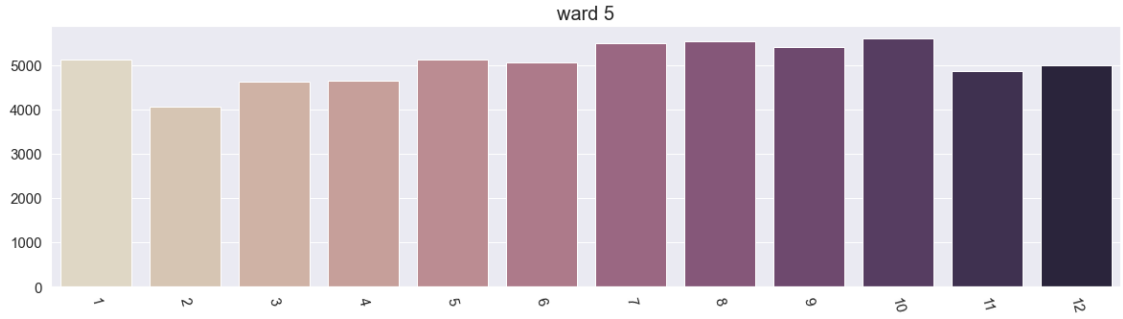

district 7
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
11
12

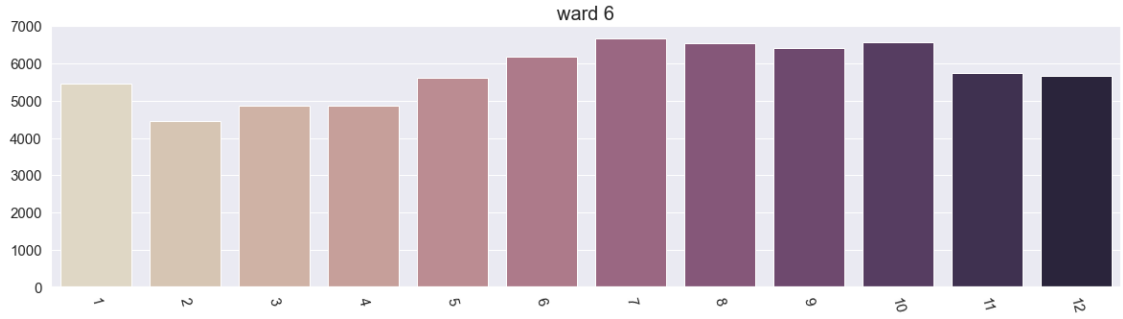

ward 1
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

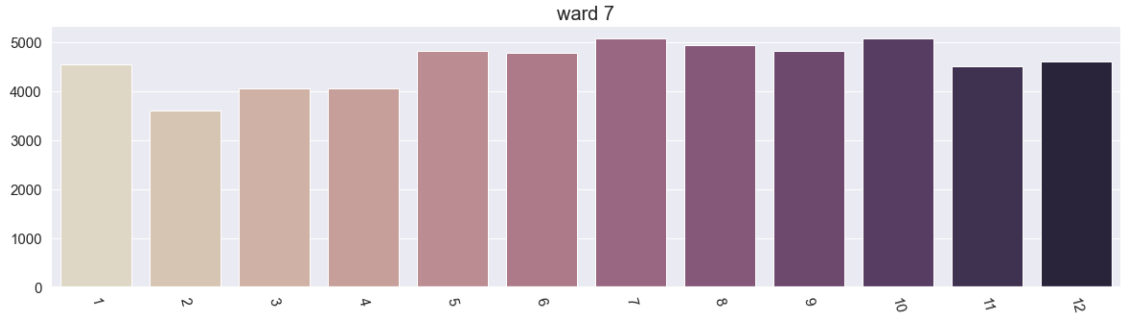

ward 2
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

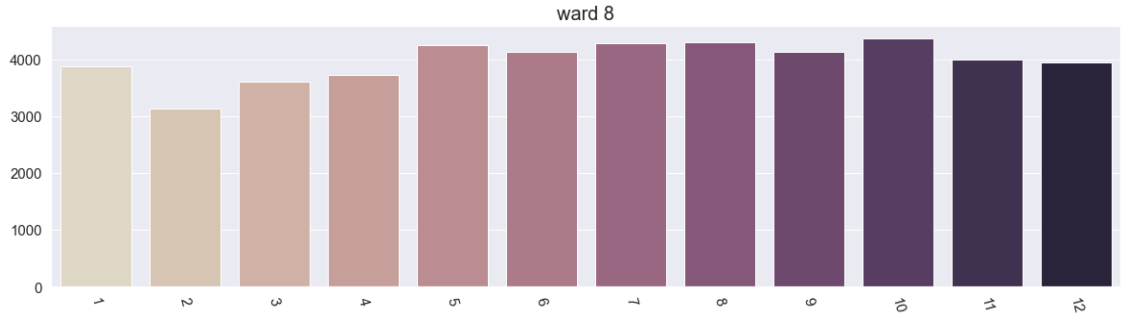

ward 3
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
11
12


ward 4
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
11
12

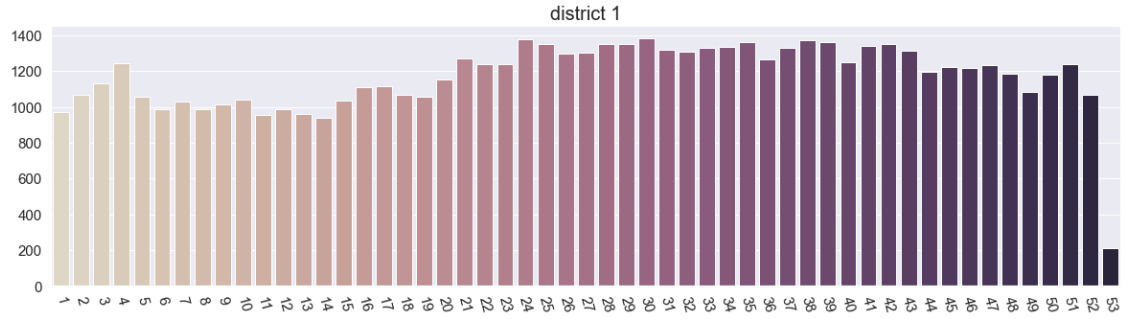

ward 5
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

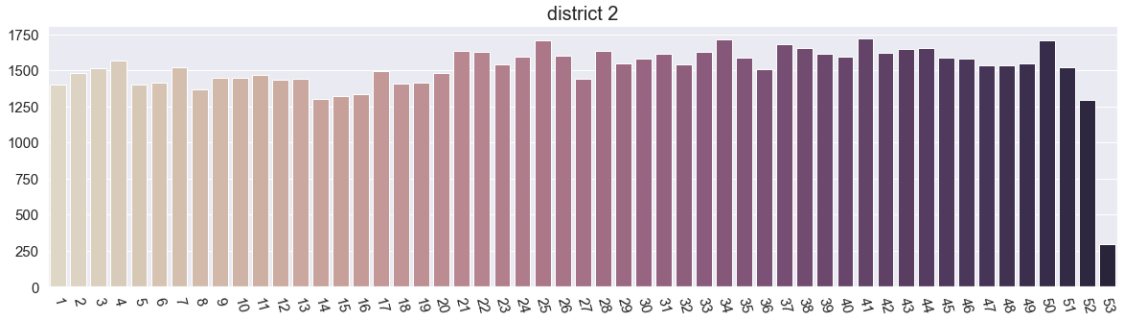

ward 6
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12

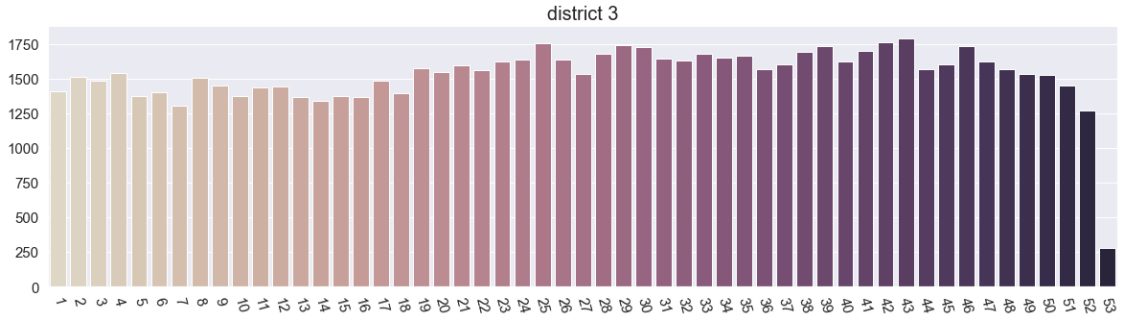


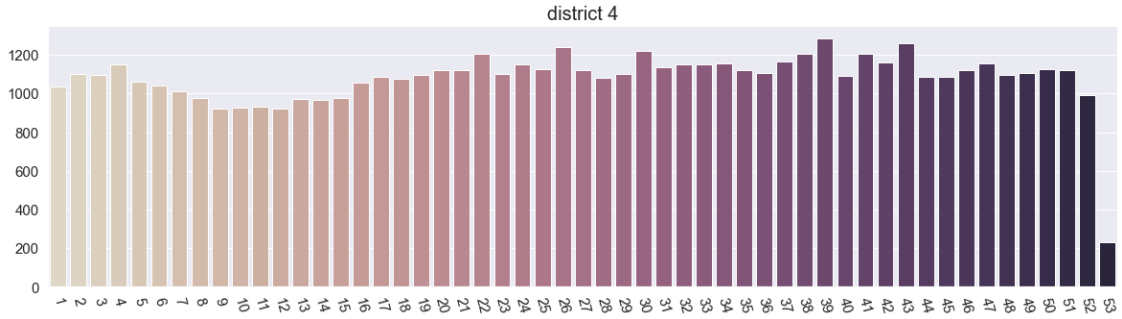


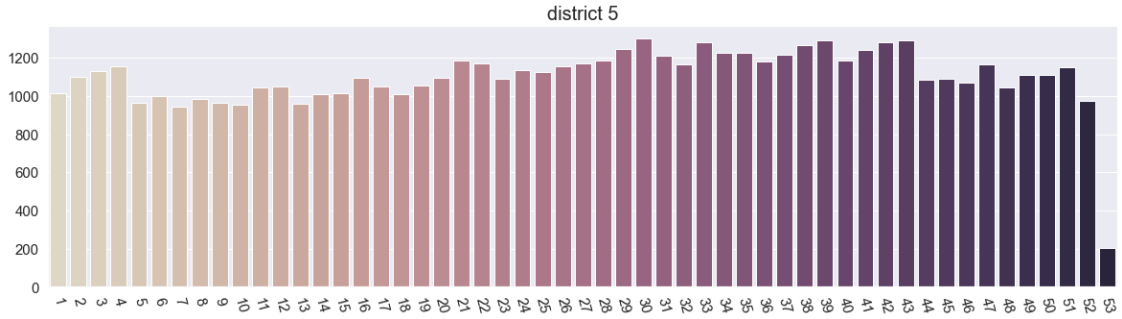
[160]:
```
districtplot('week')
wardplot('week')
# number of crimes in district and ward by week
```

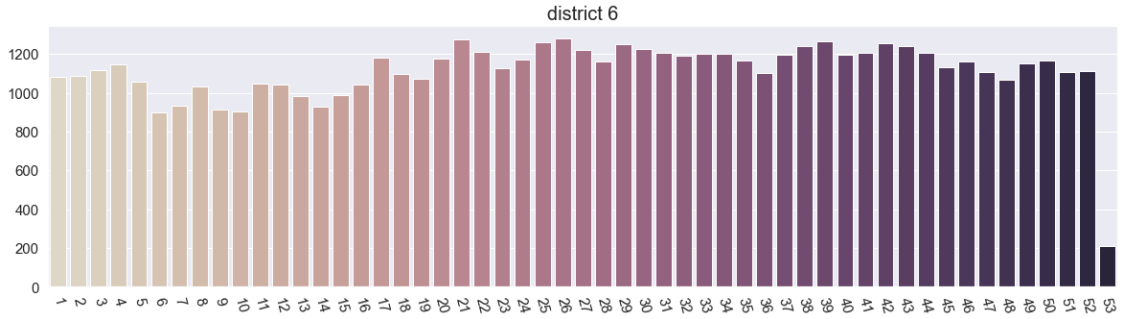


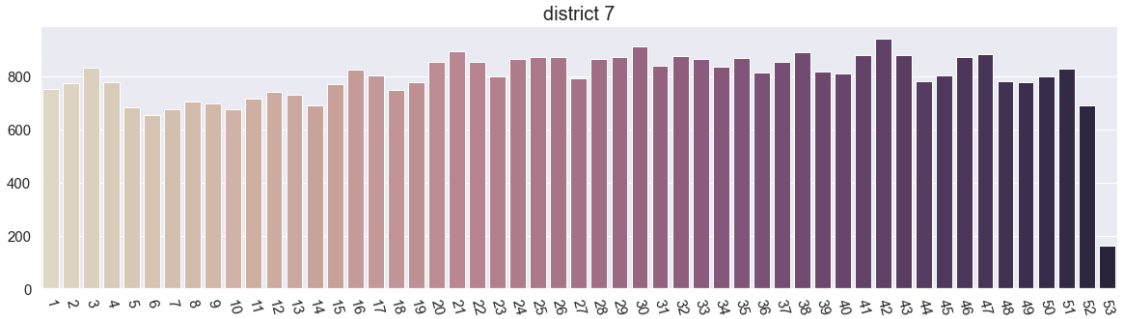

district 2
0
250
500
750
1000
1250
1500
1750
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

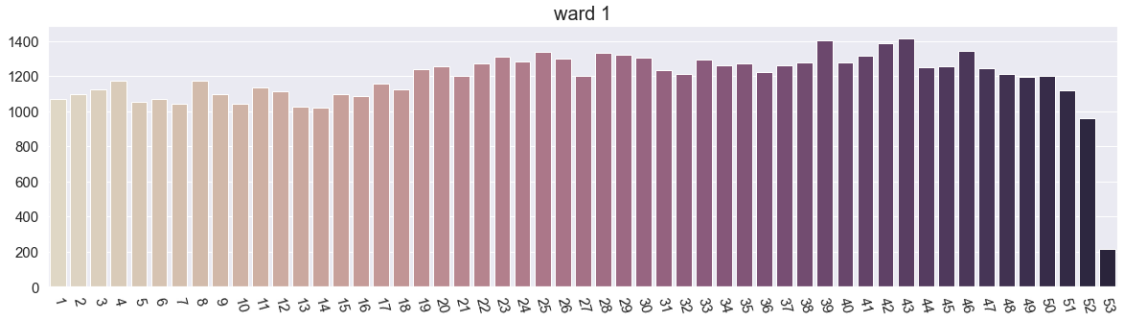

district 3
0
250
500
750
1000
1250
1500
1750
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

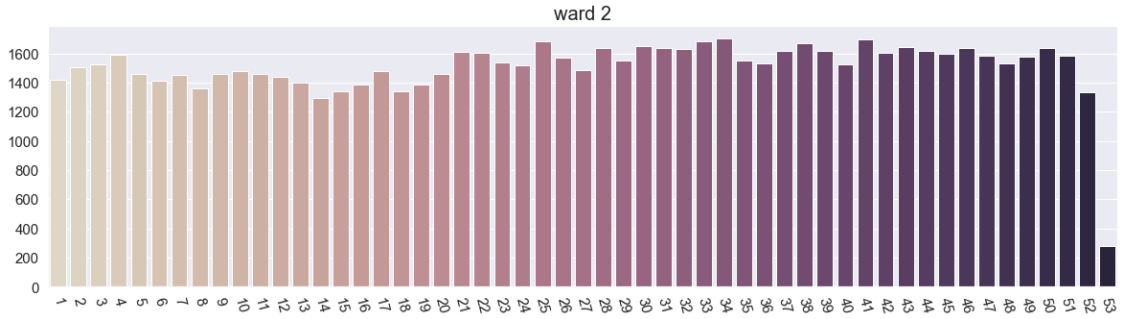

district 4
0
200
400
600
800
1000
1200
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

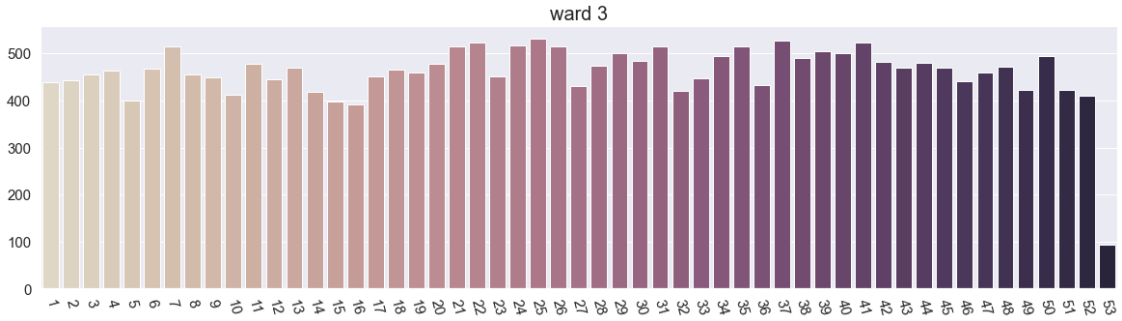

district 5
0
200
400
600
800
1000
1200
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

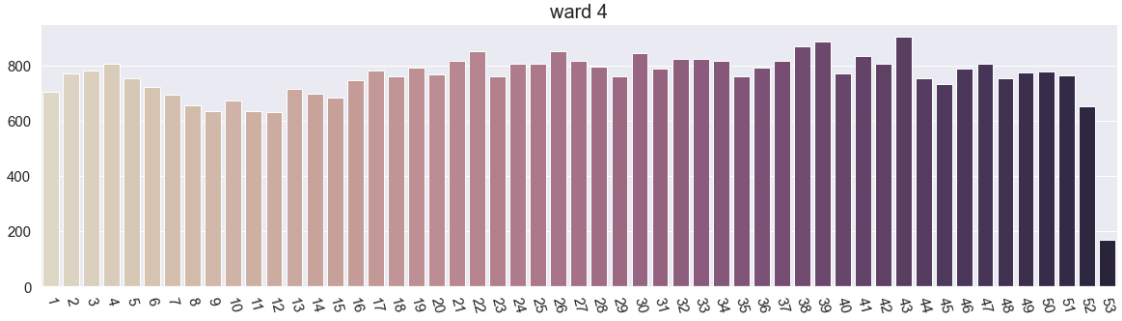

district 6
0
200
400
600
800
1000
1200
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

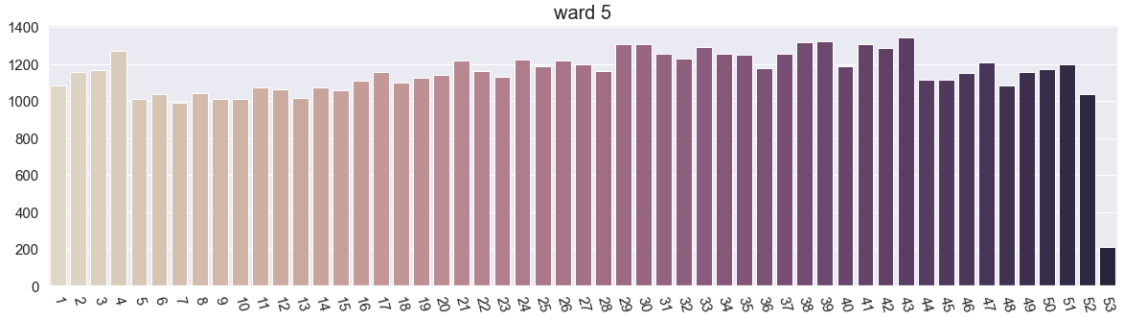

district 7
0
200
400
600
800
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

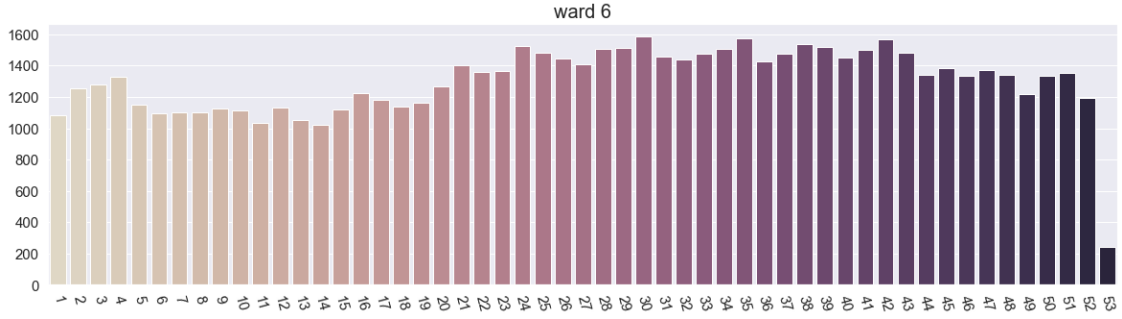

ward 1
0
200
400
600
800
1000
1200
1400
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

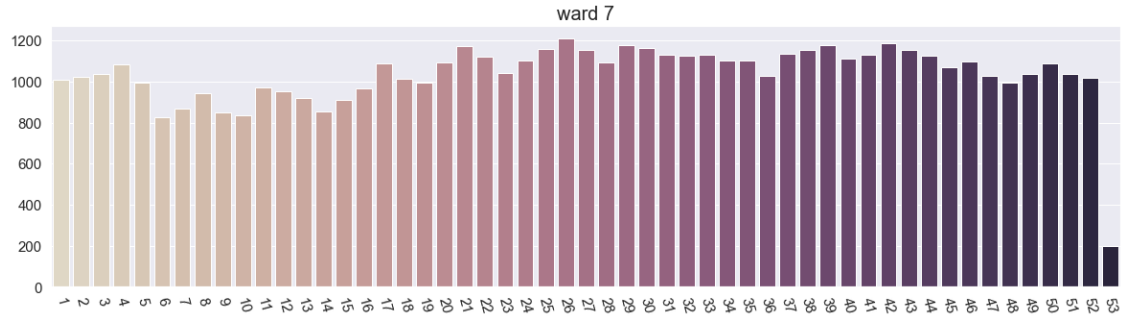

ward 2
0
200
400
600
800
1000
1200
1400
1600
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

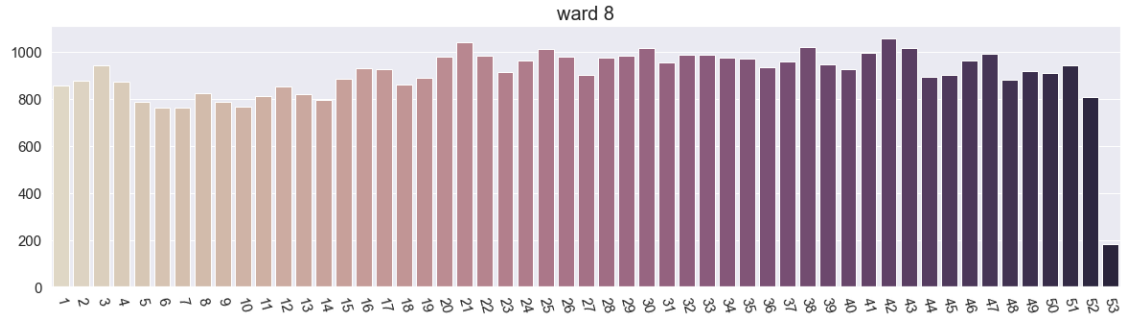

ward 3
0
100
200
300
400
500
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53


ward 4
0
200
400
600
800
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

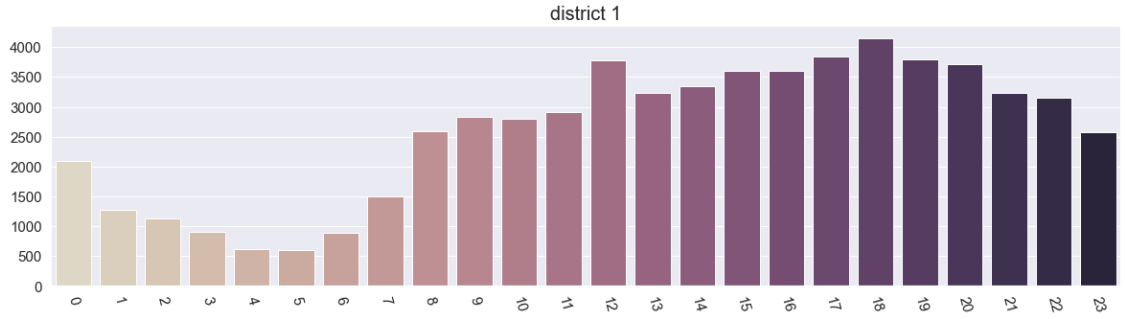

ward 5
0
200
400
600
800
1000
1200
1400
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

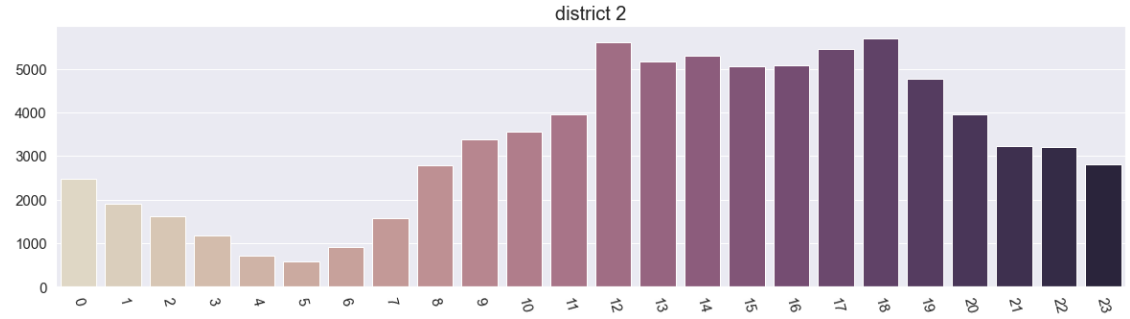

ward 6
0
200
400
600
800
1000
1200
1400
1600
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

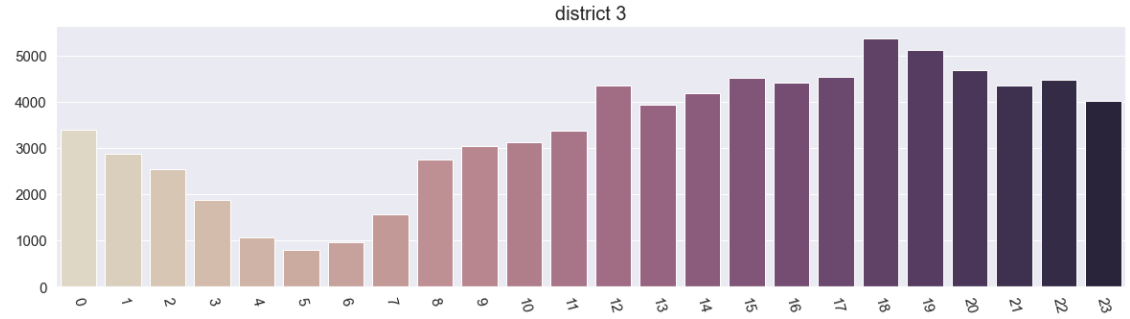


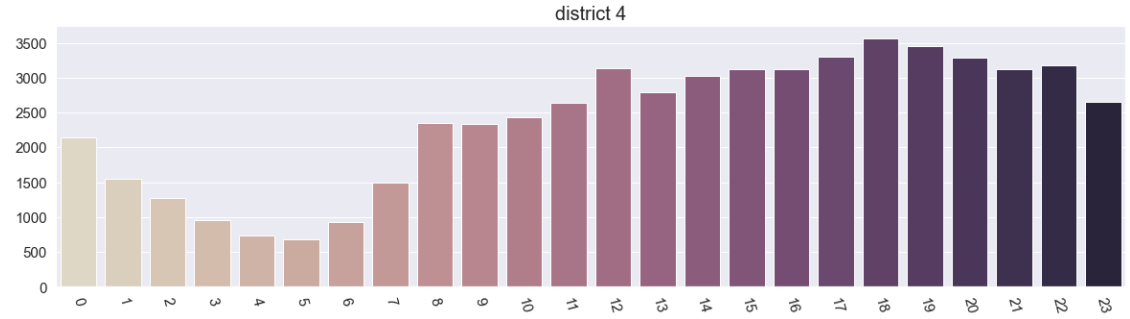


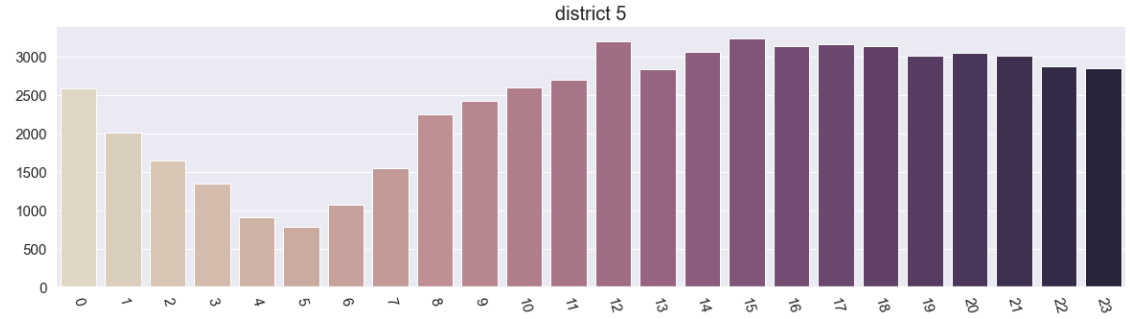
[161]:
```
districtplot('hour')
wardplot('hour')
# number of crimes in district and ward by hour
```

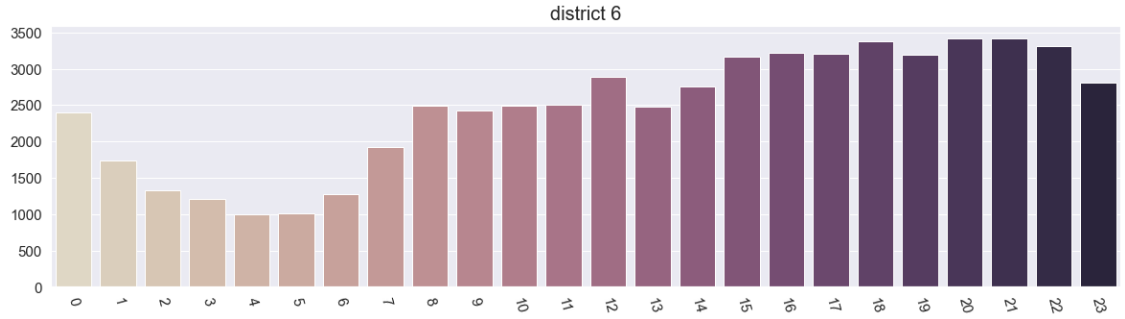


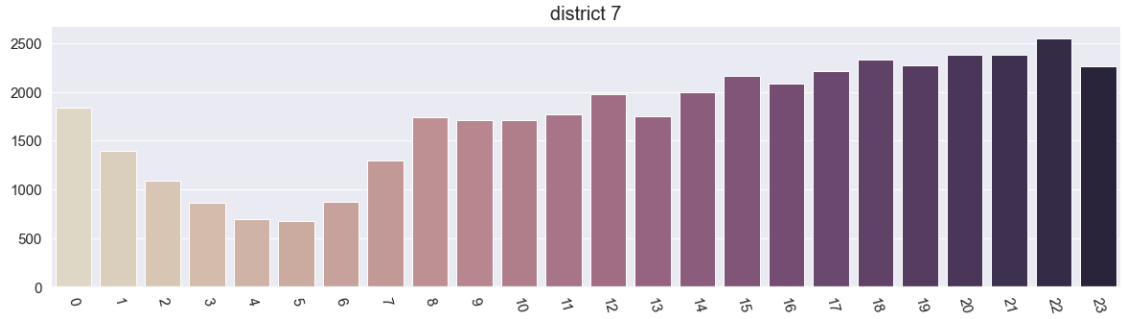

district 2
0
1000
2000
3000
4000
5000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

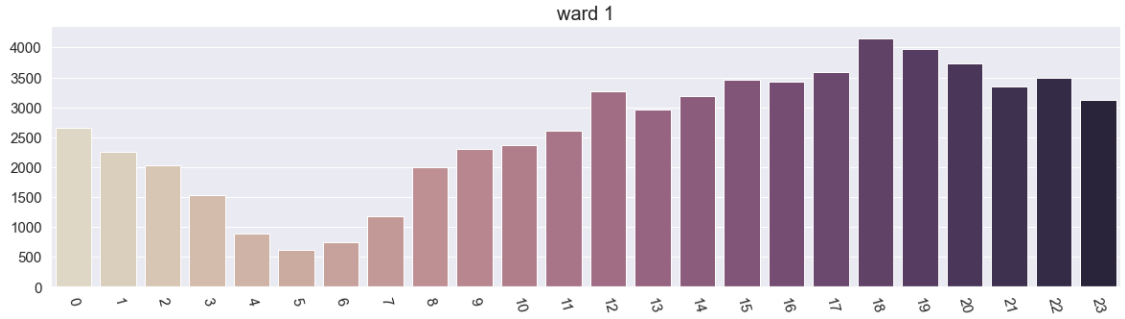

district 3
0
1000
2000
3000
4000
5000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

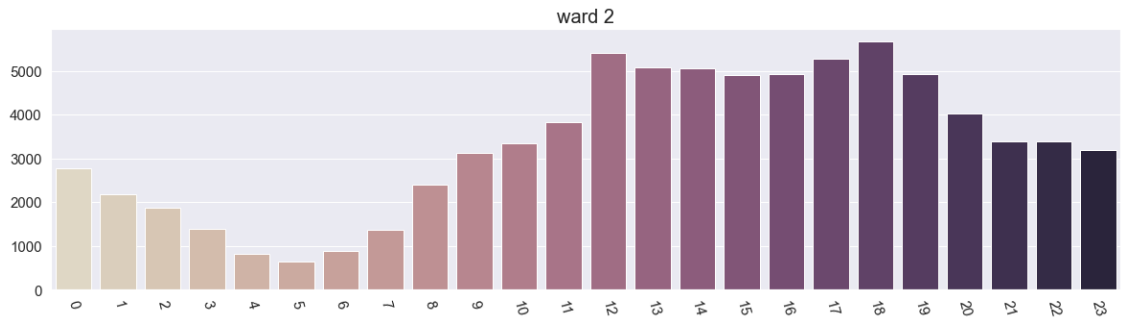

district 4
0
500
1000
1500
2000
2500
3000
3500
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

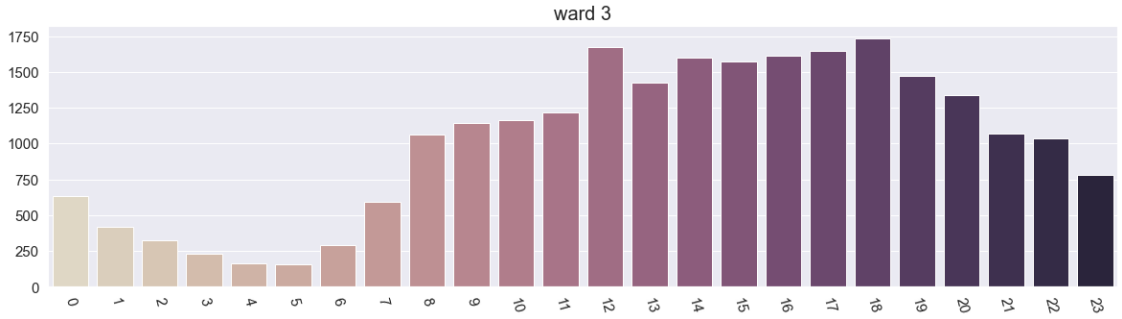

district 5
0
500
1000
1500
2000
2500
3000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

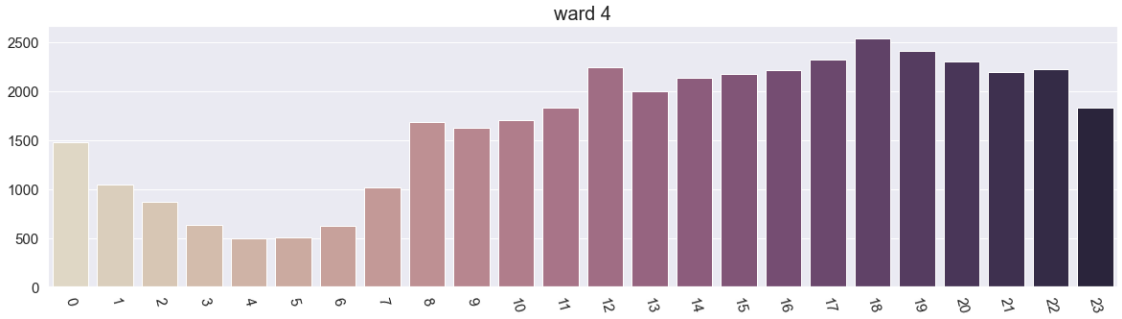

district 6
3500
3000
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

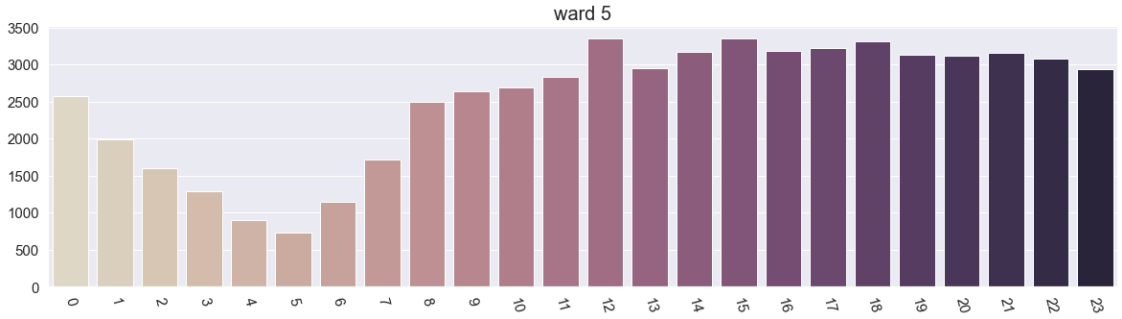

district 7
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

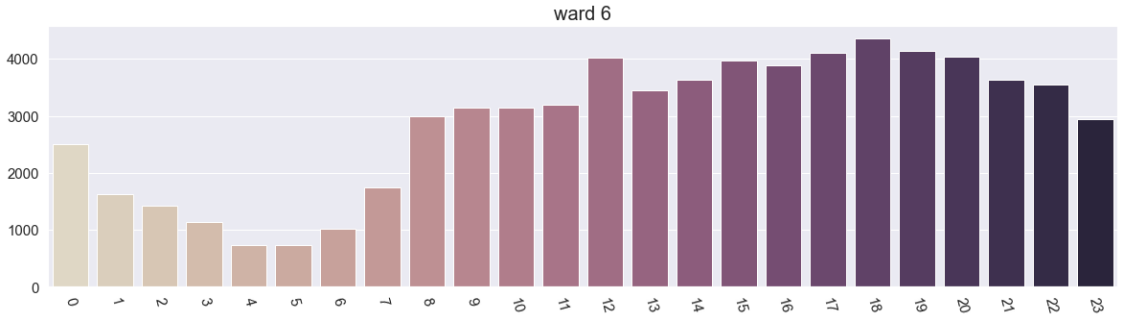

ward 1
4000
3500
3000
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

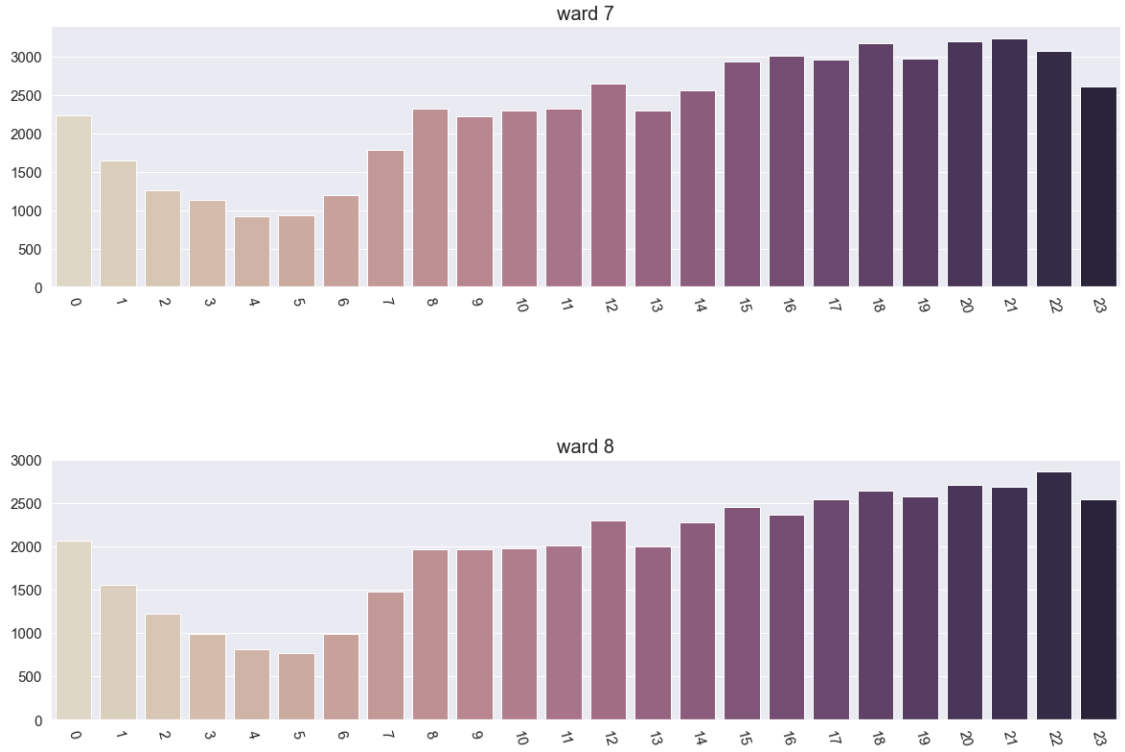

ward 2
5000
4000
3000
2000
1000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


ward 3
1750
1500
1250
1000
750
500
250
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


ward 4
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


ward 5
3500
3000
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


ward 6
4000
3000
2000
1000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23



Number of different crimes from 2020 to 2021

[162]:
```
# define a function to caculate the number of crimes by offense, offensegroup,␣
 ↪method and shift
def numof(type):
    num=[]
    for j in range(1,121):
        if type == 'property':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
 ↪offensegroup=='property'][crime_filtered_newfeature.
 ↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'violent':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
 ↪offensegroup=='violent'][crime_filtered_newfeature.
 ↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'gun':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
 ↪METHOD=='gun'][crime_filtered_newfeature.
 ↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'knife':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
 ↪METHOD=='knife'][crime_filtered_newfeature.
 ↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
```

```
        elif type == 'others':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
↪METHOD=='others'][crime_filtered_newfeature.
↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'day':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
↪SHIFT=='day'][crime_filtered_newfeature.
↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'evening':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
↪SHIFT=='evening'][crime_filtered_newfeature.
↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        elif type == 'midnight':
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
↪SHIFT=='midnight'][crime_filtered_newfeature.
↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]
        else:
            temp = crime_filtered_newfeature[crime_filtered_newfeature.
↪OFFENSE_code==type][crime_filtered_newfeature.
↪YEAR==2021][crime_filtered_newfeature.dayofyear==j].shape[0]

        if temp != 0:
            num.append(temp)
    return num
```

[163]:
```
#offense_code: theft/other:1, theft f/auto:2, burglary:3,
#               assault w/dangerous weapon:4, robbery:5,
#               motor vehicle theft:6, homicide:7, sex abuse:8, arson:9
num_1 = numof(1)
num_2 = numof(2)
num_3 = numof(3)
num_4 = numof(4)
num_5 = numof(5)
num_6 = numof(6)
num_7 = numof(7)
num_8 = numof(8)
num_9 = numof(9)
plt.figure(figsize=(40,10))
plt.plot(range(1,len(num_1)+1),num_1)
plt.plot(range(1,len(num_2)+1),num_2)
plt.plot(range(1,len(num_3)+1),num_3)
plt.plot(range(1,len(num_4)+1),num_4)
plt.plot(range(1,len(num_5)+1),num_5)
plt.plot(range(1,len(num_6)+1),num_6)
plt.plot(range(1,len(num_7)+1),num_7)
plt.plot(range(1,len(num_8)+1),num_8)
plt.plot(range(1,len(num_9)+1),num_9)
```

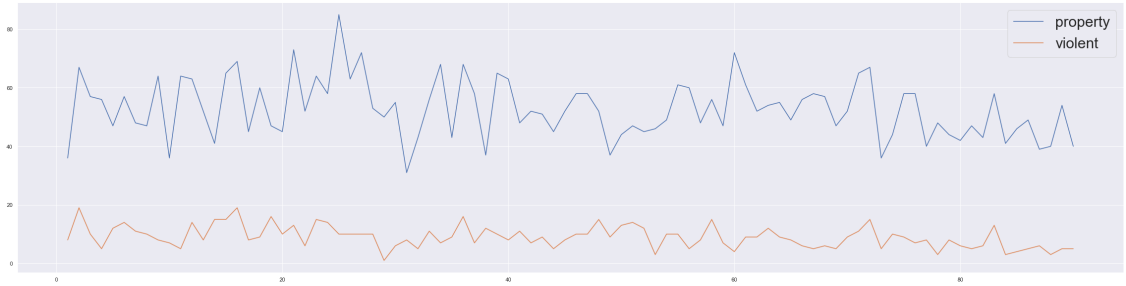
```
plt.legend(['theft/other','theft f/auto','burglary',
            'assault','robbery','motor theft','homicide',
            'sex abuse','arson'],loc='upper right',fontsize=30)
plt.show()
# from 2021-1-1 to 2021-4-30
```

[163]:

[164]:
```
num_pro=numof('property')
num_vio=numof('violent')
num_gun=numof('gun')
num_kni=numof('knife')
num_oth=numof('others')
num_day=numof('day')
num_eve=numof('evening')
num_mid=numof('midnight')
```

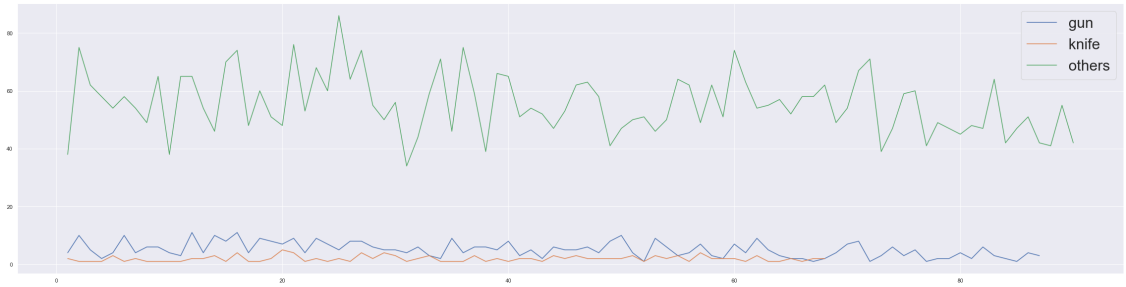
[165]:
```
plt.figure(figsize=(40,10))
plt.plot(range(1,len(num_pro)+1),num_pro)
plt.plot(range(1,len(num_vio)+1),num_vio)
plt.legend(['property','violent'],loc='upper right',fontsize=30)
plt.show()
# from 2021-1-1 to 2021-4-30
```

[165]:



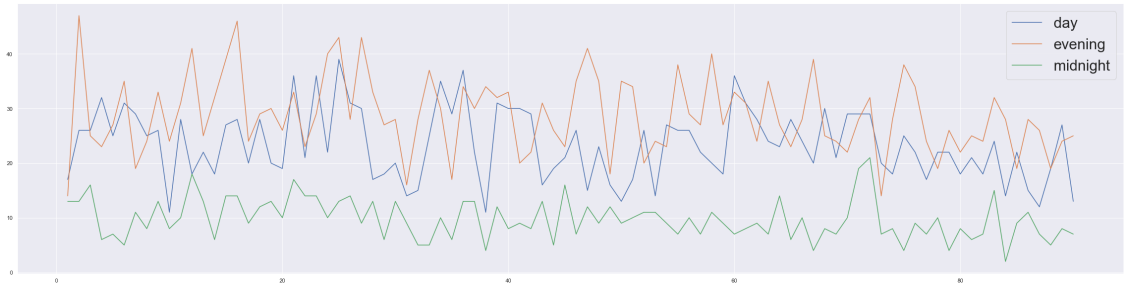
[166]:
```
plt.figure(figsize=(40,10))
plt.plot(range(1,len(num_gun)+1),num_gun)
plt.plot(range(1,len(num_kni)+1),num_kni)
plt.plot(range(1,len(num_oth)+1),num_oth)
plt.legend(['gun','knife','others'],loc='upper right',fontsize=30)
plt.show()
# from 2021-1-1 to 2021-4-30
```

[166]:



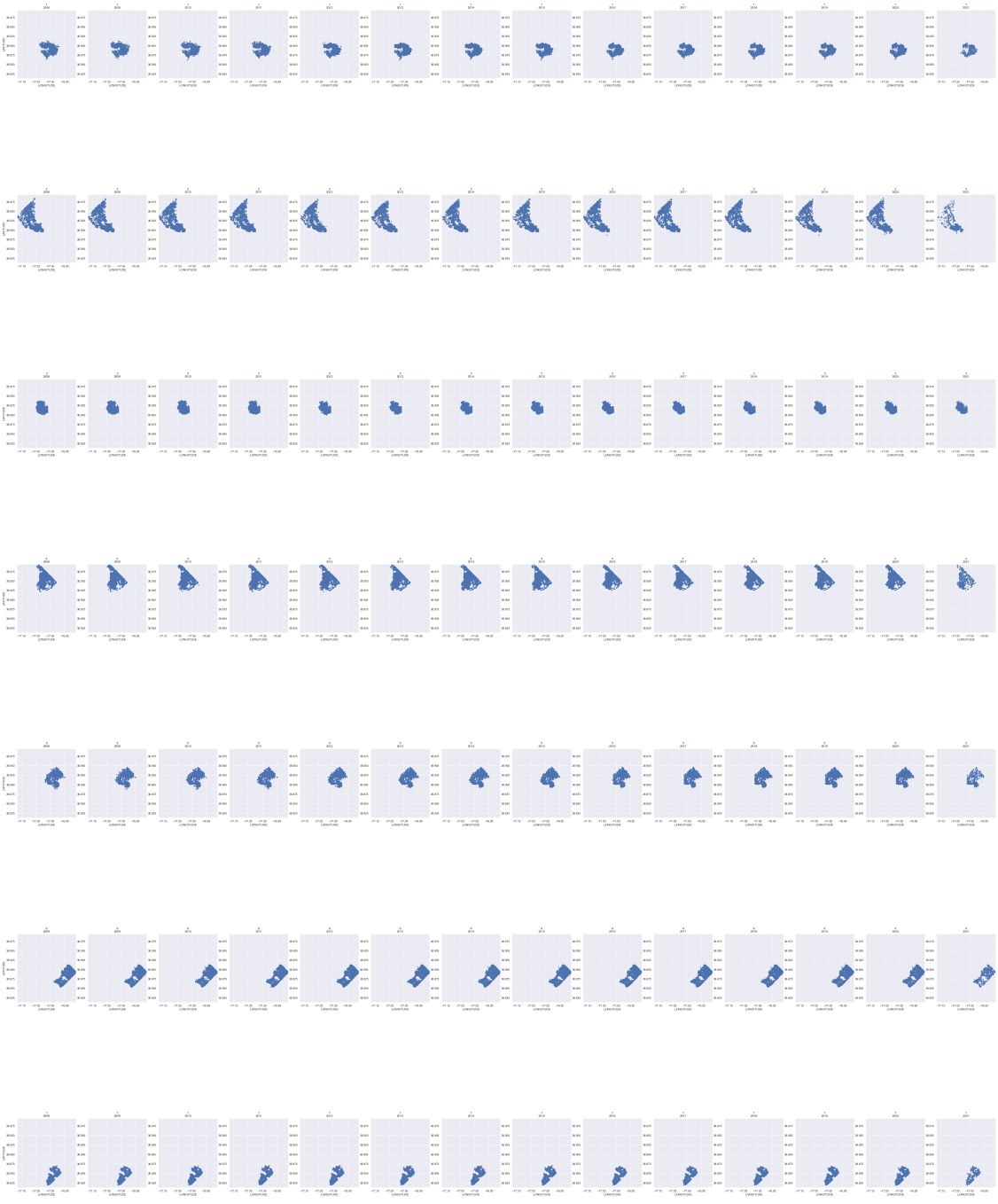
[167]:
```
plt.figure(figsize=(40,10))
plt.plot(range(1,len(num_day)+1),num_day)
plt.plot(range(1,len(num_eve)+1),num_eve)
plt.plot(range(1,len(num_mid)+1),num_mid)
plt.legend(['day','evening','midnight'],loc='upper right',fontsize=30)
plt.show()
# from 2021-1-1 to 2021-4-30
```

[167]:



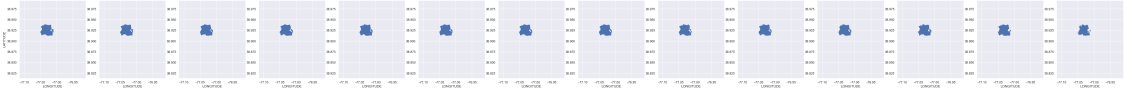
The distribution of crimes in each district by year.

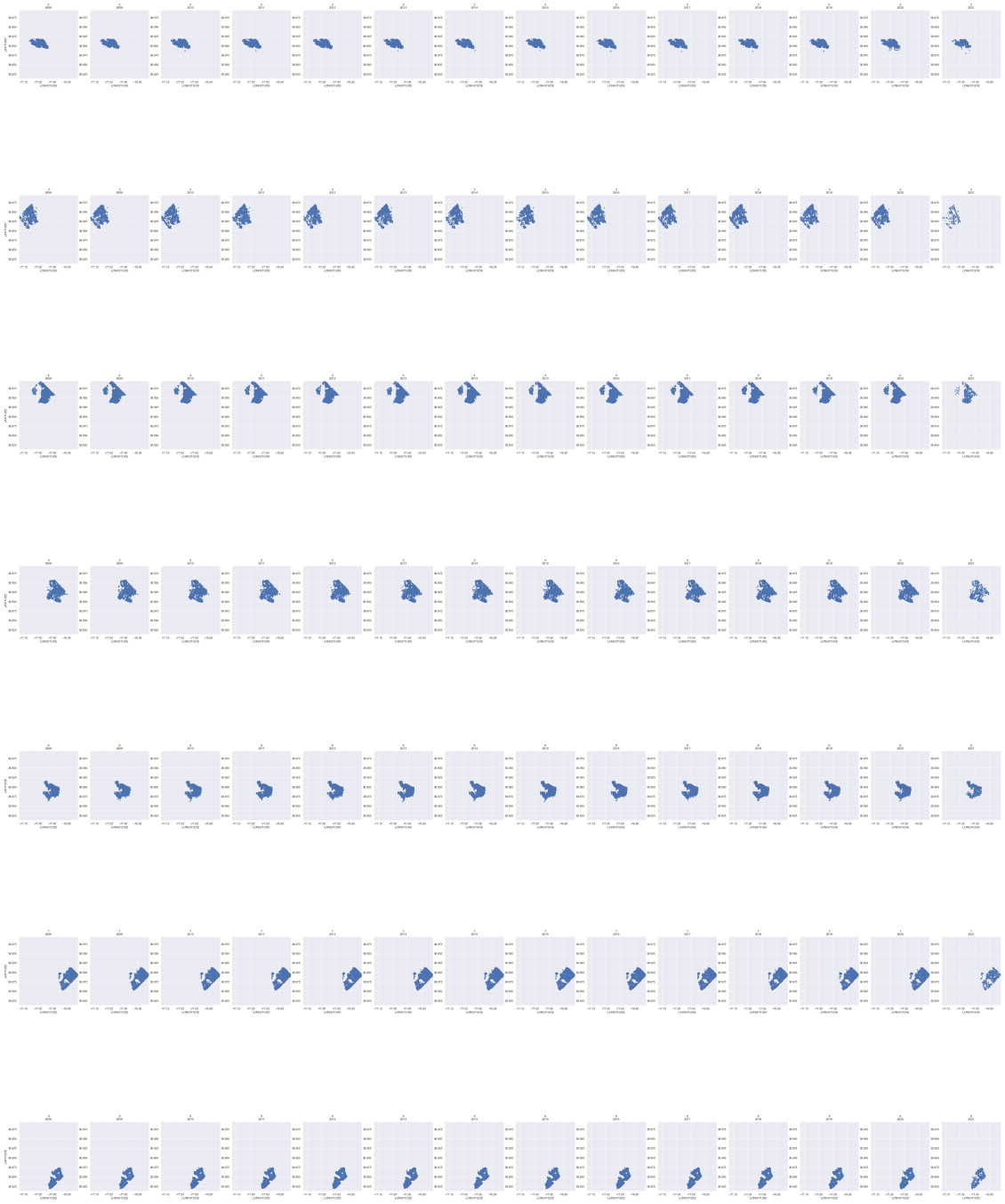
[168]:
```
changes(['LATITUDE','LONGITUDE','DISTRICT'],'DISTRICT')
```



The distribution of crimes in each ward by year.

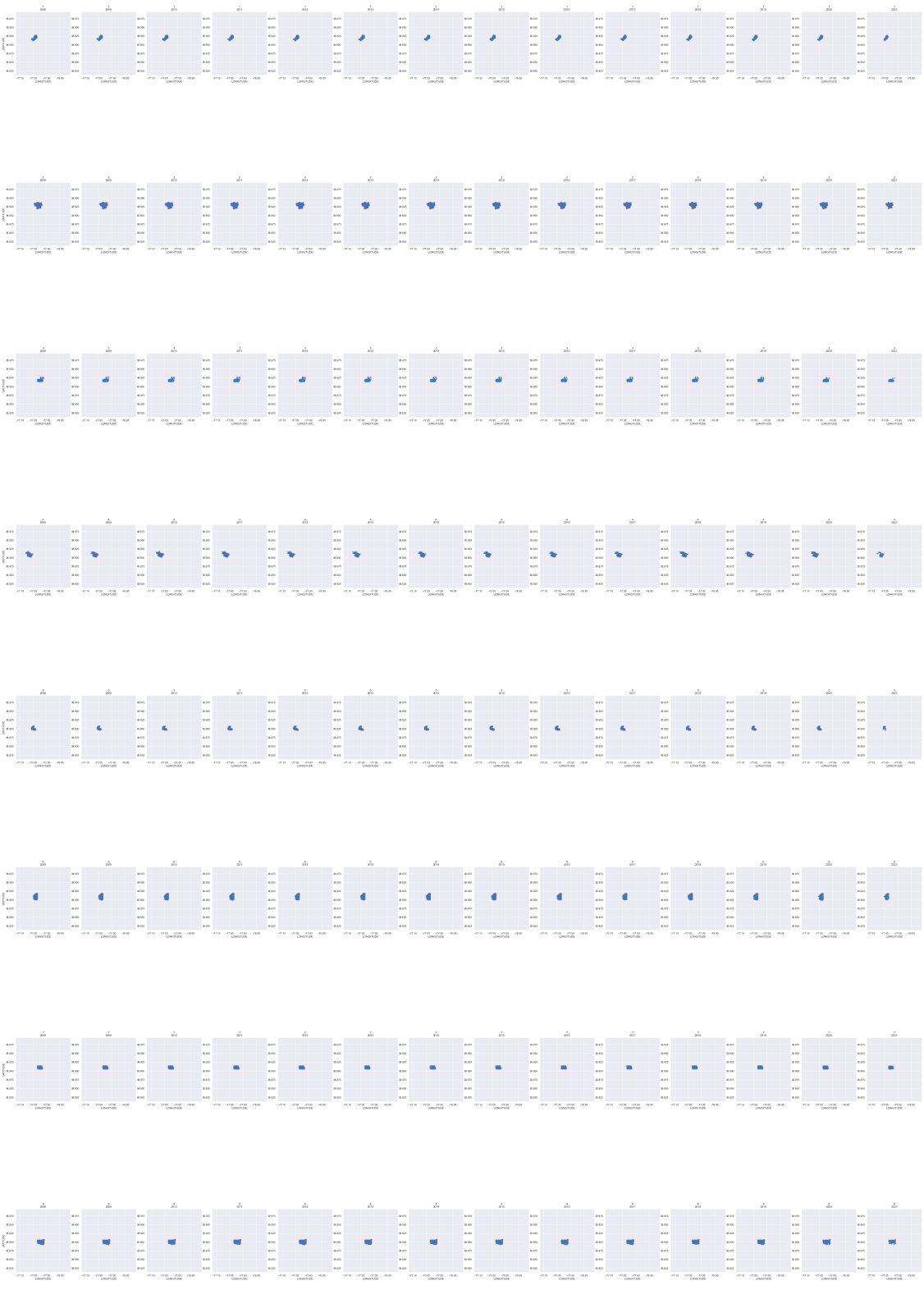
[169]:
```
changes(['LATITUDE','LONGITUDE','WARD'],'WARD')
```

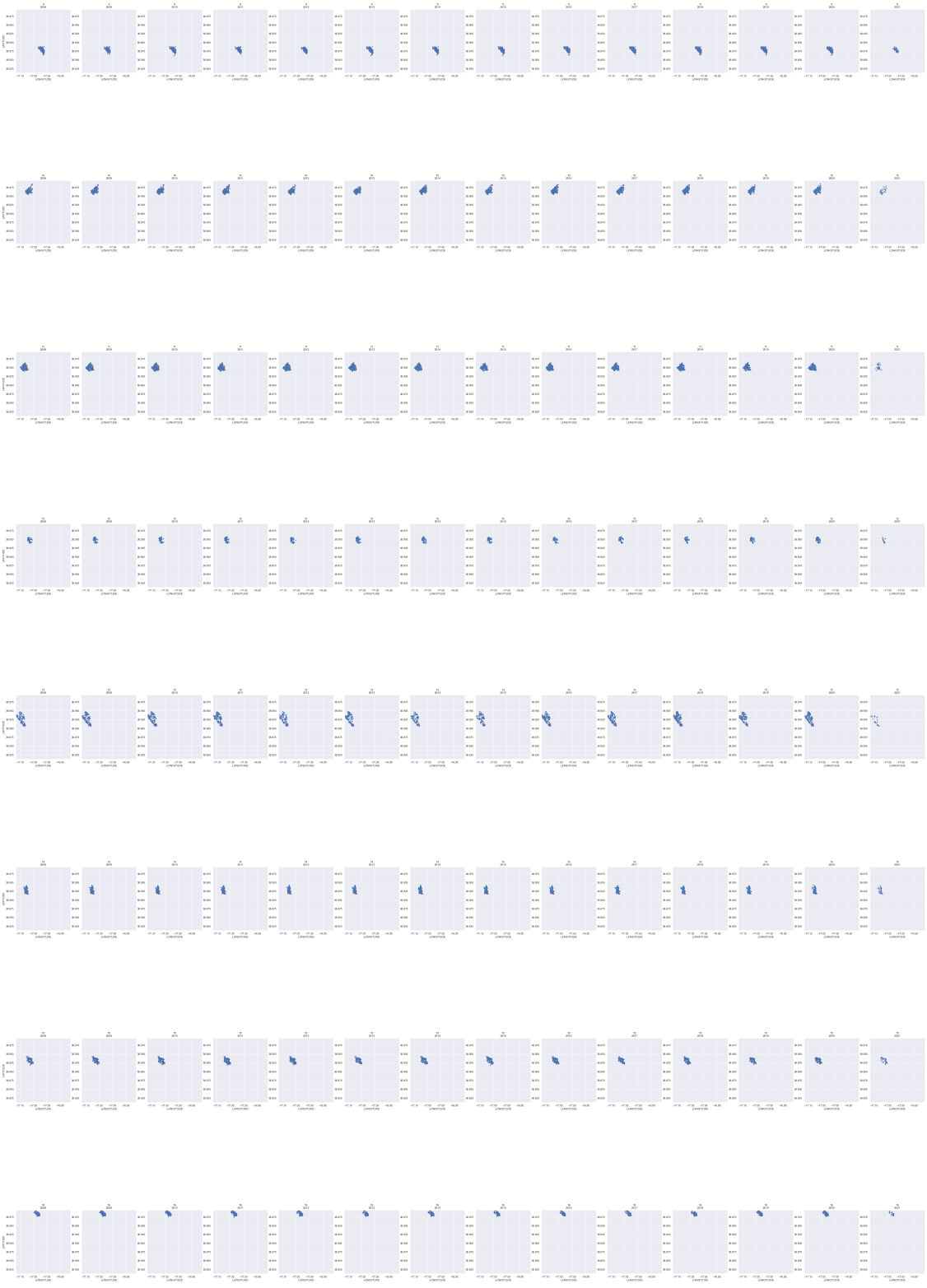


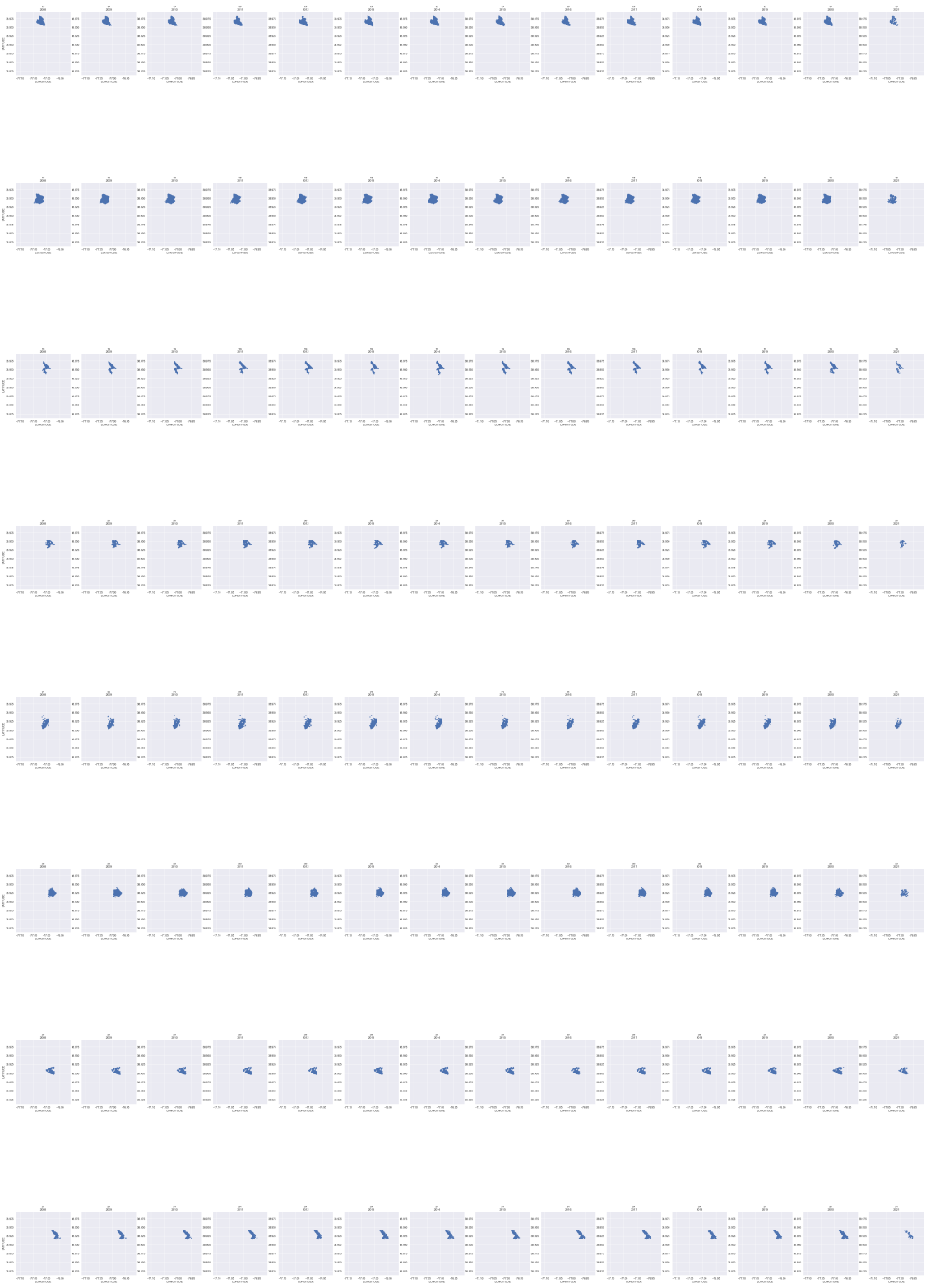


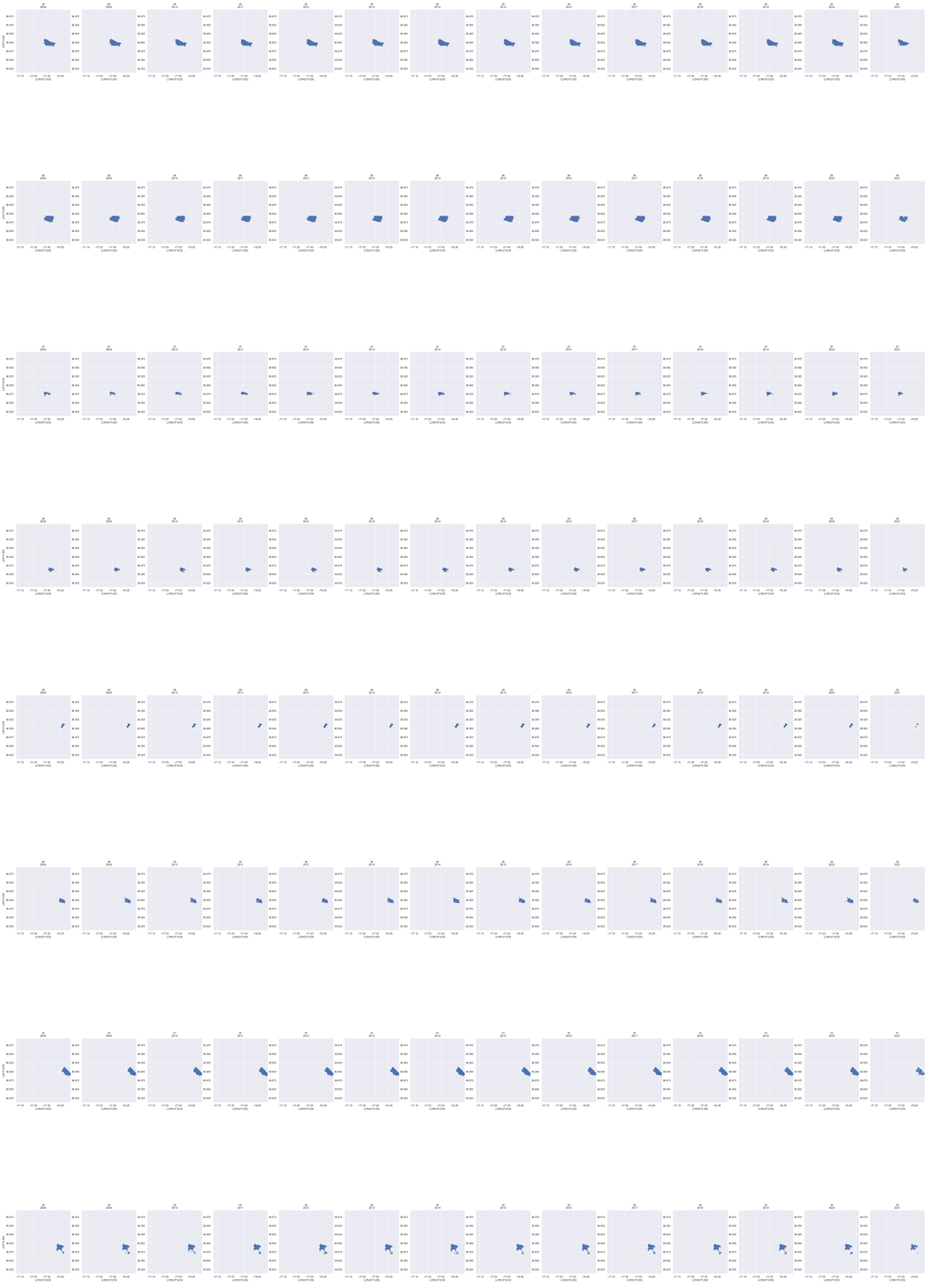
The distribution of crimes in each NEIGHBORHOOD_CLUSTER by year.

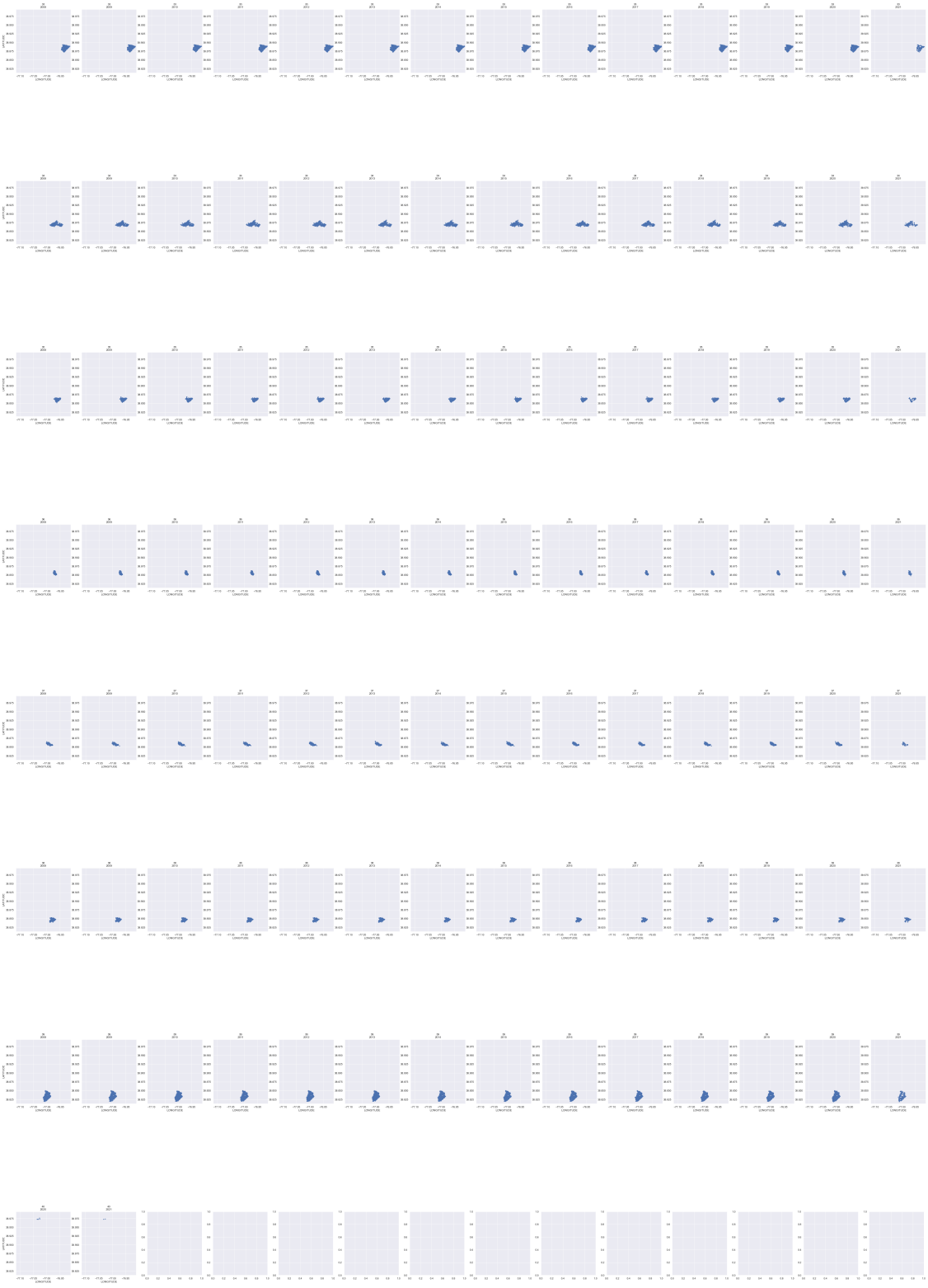
[170]:
```
changes(['LATITUDE','LONGITUDE','NEIGHBORHOOD_CLUSTER'],'NEIGHBORHOOD_CLUSTER')
```

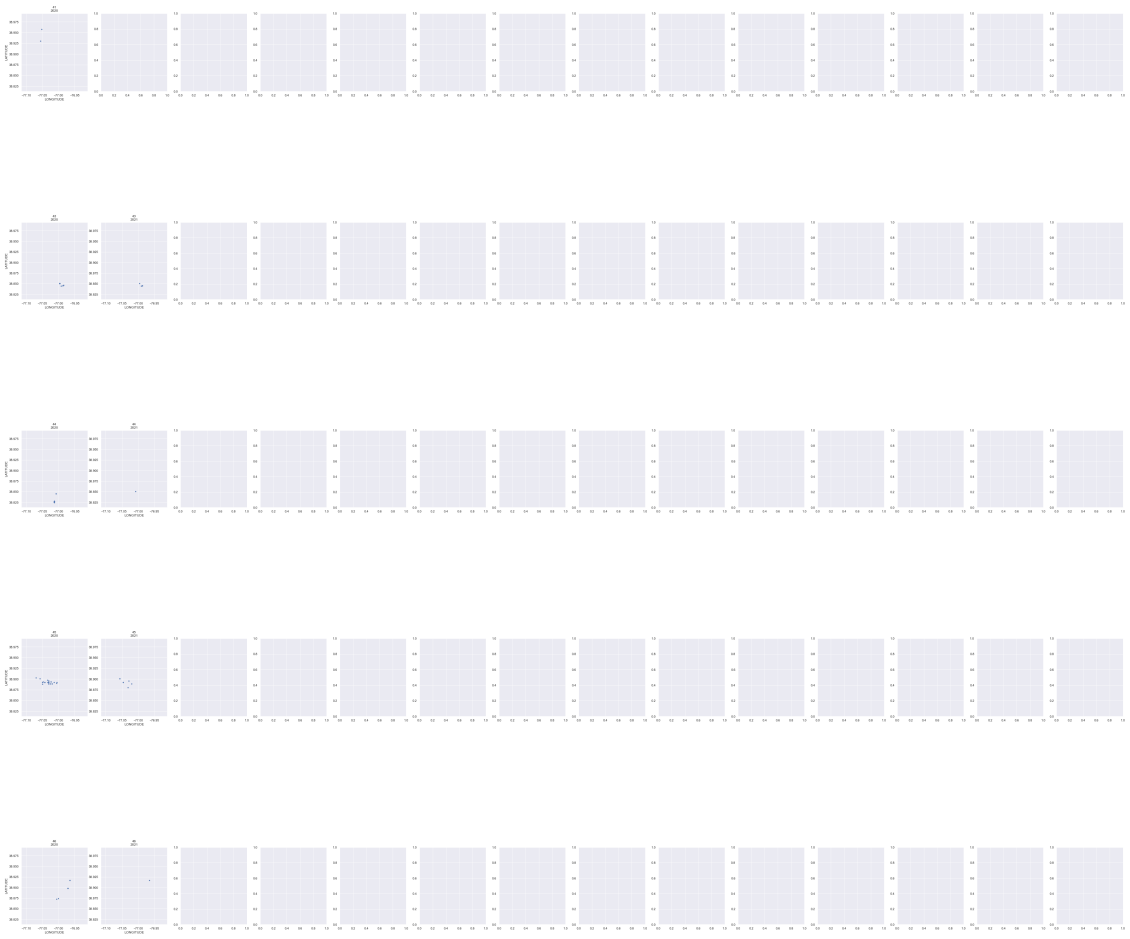












4 3 Classify(or cluster) the geography by the crime events(Task 3)

@task:According to the distance function defined by yourself, divide the block/location information into several categories. You are asked to use classification or clustering methods such as kNN/KMeans. Encourage multiple methods apply to this task.

4.0.1 3-1 Feature Selection——prestep (some features have been deleted in outlier detection step)

Correlation analysis.
Since YBLOCK and LATITUDE has corr = 1, XBLOCK and LONGITUDE has corr = 1, DISTRICT and PSA has corr = 1, YEAR and CCN has corr = 1, WARD and NEIGHBORHOOD_CLUSTER has corr = 0.93, we choose to delete YBLOCK, XBLOCK, PSA, CCN, NEIGHBORHOOD_CLUSTER.

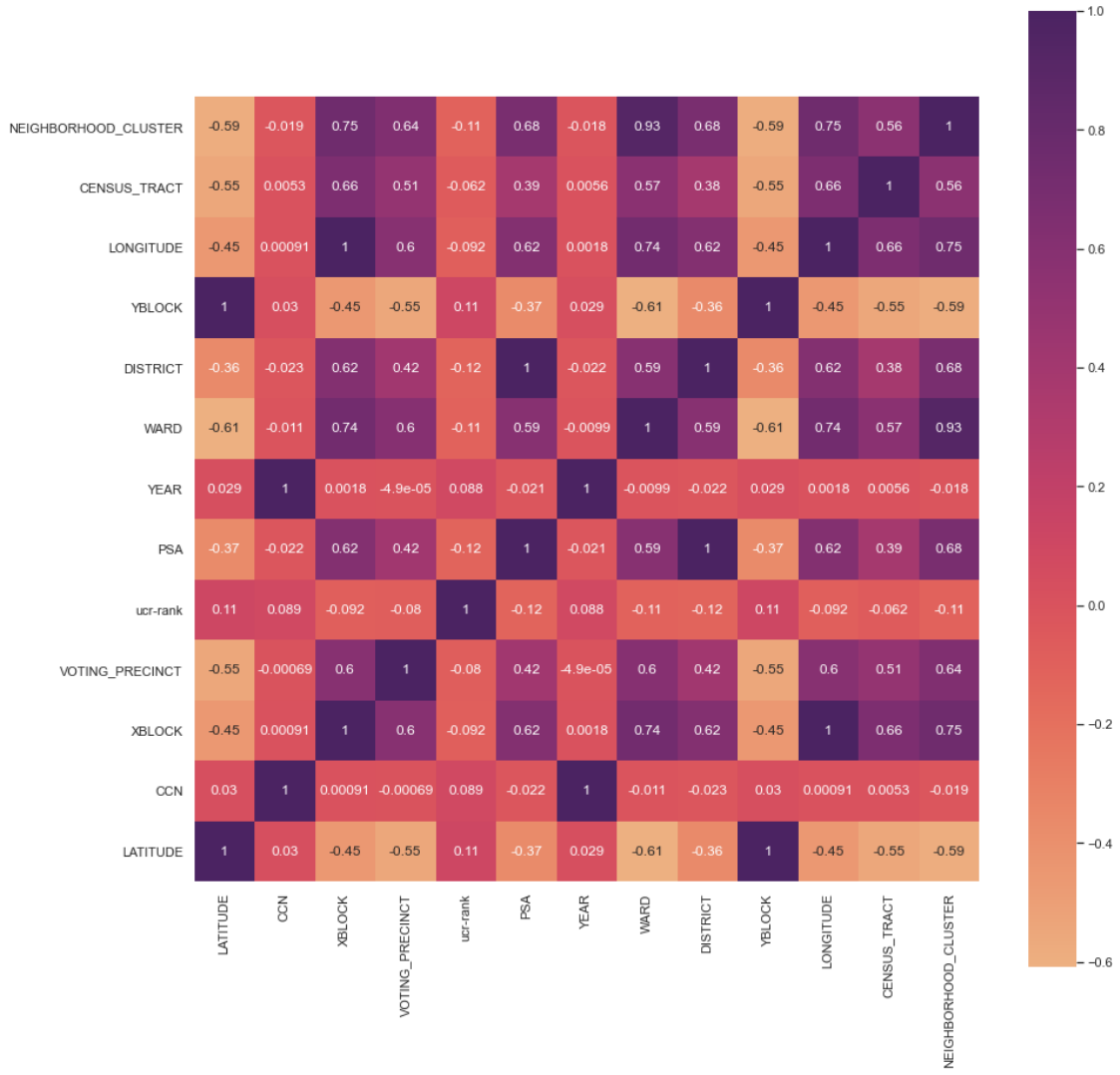
[171]:
```
f, ax=plt.subplots(figsize=(15, 15))
sns.heatmap(corr_matrix, cmap = 'flare',annot = True,square=True)
```

```
ax.set_ylim([13,0])
ax.set_xlim([13,0])
plt.show()
```

[171]: <AxesSubplot:>

[171]: (13.0, 0.0)

[171]: (13.0, 0.0)



[172]:
```
del crime_filtered['XBLOCK']
del crime_filtered['YBLOCK']
del crime_filtered['PSA']
del crime_filtered['CCN']
```

```
del crime_filtered['NEIGHBORHOOD_CLUSTER']
del crime_filtered_newfeature['XBLOCK']
del crime_filtered_newfeature['YBLOCK']
del crime_filtered_newfeature['PSA']
del crime_filtered_newfeature['CCN']
del crime_filtered_newfeature['NEIGHBORHOOD_CLUSTER']

crime_filtered_newfeature.columns
# show
```

[172]:
```
Index(['CENSUS_TRACT', 'offensegroup', 'LONGITUDE', 'END_DATE', 'SHIFT',
       'DISTRICT', 'WARD', 'YEAR', 'sector', 'ucr-rank', 'VOTING_PRECINCT',
       'BLOCK', 'START_DATE', 'OFFENSE', 'ANC', 'REPORT_DAT', 'METHOD',
       'LATITUDE', 'OFFENSE_code', 'SPAN', 'TIME_TO_REPORT', 'DATE', 'MONTH',
       'DAY', 'hour', 'dayofyear', 'week', 'weekofyear', 'dayofweek',
       'weekday', 'quarter'],
      dtype='object')
```

Besides, there are several features that are unhelpful for modeling. We delete them.
All information of START_DATE, END_DATE and REPORT_DAT are in DATE, SPAN and TIME_TO_REPORT, then we should delete START_DATE, END_DATE and REPORT_DAT. BLOCK has many values and theses values can not be interpreted well enough, which is unhelpful to the analysis. We delete it too.

[173]:
```
del crime_filtered['START_DATE']
del crime_filtered['END_DATE']
del crime_filtered['REPORT_DAT']
del crime_filtered['BLOCK']
del crime_filtered_newfeature['START_DATE']
del crime_filtered_newfeature['END_DATE']
del crime_filtered_newfeature['REPORT_DAT']
del crime_filtered_newfeature['BLOCK']

crime_filtered_newfeature.columns
# show
```

[173]:
```
Index(['CENSUS_TRACT', 'offensegroup', 'LONGITUDE', 'SHIFT', 'DISTRICT',
       'WARD', 'YEAR', 'sector', 'ucr-rank', 'VOTING_PRECINCT', 'OFFENSE',
       'ANC', 'METHOD', 'LATITUDE', 'OFFENSE_code', 'SPAN', 'TIME_TO_REPORT',
       'DATE', 'MONTH', 'DAY', 'hour', 'dayofyear', 'week', 'weekofyear',
       'dayofweek', 'weekday', 'quarter'],
      dtype='object')
```

4.0.2 3-2 Classify the geography by crime events

3-2-0 Preparation

[174]:
```
from functools import reduce;
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
np.random.seed(42)
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score
from matplotlib import rcParams
rcParams.update({'figure.autolayout':True})
plt.style.use('fivethirtyeight')
sns.set_style('darkgrid')
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
# for model evaluation
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.multiclass import OneVsRestClassifier
from sklearn.preprocessing import label_binarize, LabelBinarizer
```

Generate 'crime_model' specifically for modeling.

[175]:
```
crime_model = pd.get_dummies(crime_filtered_newfeature,␣
↪columns=['OFFENSE','offensegroup','SHIFT','METHOD'], drop_first=False)
```

[176]:
```
crime_model
# show
```

[176]:

	CENSUS_TRACT	LONGITUDE	DISTRICT	WARD	YEAR	sector	ucr-rank	\
2	4702	-77.020913	1	6	2017	11	6	
9	7200	-77.005025	1	6	2017	13	6	
10	5800	-77.019909	1	2	2017	11	3	
23	5800	-77.022433	1	2	2017	11	7	
27	6700	-76.984577	1	6	2017	12	6	
…	…	…	…	…	…	…	…	
449163	7304	-76.988516	7	8	2021	72	8	
449169	9807	-77.012634	7	8	2021	73	3	
449178	10900	-77.008913	7	8	2021	73	6	
449184	7409	-76.978424	7	8	2021	72	8	
449192	9700	-76.991523	7	8	2021	73	7	

	VOTING_PRECINCT	ANC	LATITUDE	…	OFFENSE_theft f/auto	\
2	18	65	38.902518	…	0	
9	131	64	38.876476	…	0	
10	129	23	38.899059	…	0	
23	129	23	38.898446	…	1	

```
27                  88  62  38.889799  …                    0
…                 …  ..       …  …                 …
449163             120  85  38.842541  …                    0
449169             124  84  38.831298  …                    0
449178             126  84  38.819381  …                    0
449184             116  82  38.844765  …                    0
449192             125  85  38.830632  …                    1

        OFFENSE_theft/other  offensegroup_property offensegroup_violent  \
2                         1                        1                     0
9                         1                        1                     0
10                        0                        0                     1
23                        0                        1                     0
27                        1                        1                     0
…                       …                        …                   …
449163                    0                        1                     0
449169                    0                        0                     1
449178                    1                        1                     0
449184                    0                        1                     0
449192                    0                        1                     0

        SHIFT_day  SHIFT_evening  SHIFT_midnight  METHOD_gun  METHOD_knife  \
2               1              0               0           0             0
9               0              1               0           0             0
10              0              0               1           0             0
23              1              0               0           0             0
27              0              1               0           0             0
…             …              …               …           …           …
449163          1              0               0           0             0
449169          0              0               1           1             0
449178          1              0               0           0             0
449184          0              1               0           0             0
449192          0              1               0           0             0

        METHOD_others
2                   1
9                   1
10                  1
23                  1
27                  1
…                 …
449163              1
449169              0
449178              1
449184              1
449192              1
```

```
[440236 rows x 40 columns]
```

[177]:
```
crime_model.columns
```

[177]:
```
Index(['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD', 'YEAR', 'sector',
       'ucr-rank', 'VOTING_PRECINCT', 'ANC', 'LATITUDE', 'OFFENSE_code',
       'SPAN', 'TIME_TO_REPORT', 'DATE', 'MONTH', 'DAY', 'hour', 'dayofyear',
       'week', 'weekofyear', 'dayofweek', 'weekday', 'quarter',
       'OFFENSE_arson', 'OFFENSE_assault w/dangerous weapon',
       'OFFENSE_burglary', 'OFFENSE_homicide', 'OFFENSE_motor vehicle theft',
       'OFFENSE_robbery', 'OFFENSE_sex abuse', 'OFFENSE_theft f/auto',
       'OFFENSE_theft/other', 'offensegroup_property', 'offensegroup_violent',
       'SHIFT_day', 'SHIFT_evening', 'SHIFT_midnight', 'METHOD_gun',
       'METHOD_knife', 'METHOD_others'],
      dtype='object')
```

3-2-1 Classification 1: View DISTRICT as label. Select features with numerical meaning and dummy variables.

[178]:
```
variables = ['LONGITUDE', 'WARD','ucr-rank', 'VOTING_PRECINCT', 'ANC',
             'LATITUDE','SPAN','TIME_TO_REPORT', 'YEAR', 'MONTH', 'DAY',
             'hour', 'dayofyear', 'week', 'weekofyear', 'dayofweek',
             'weekday','quarter',  'OFFENSE_arson',
             'OFFENSE_assault w/dangerous weapon',
             'OFFENSE_burglary', 'OFFENSE_homicide',
             'OFFENSE_motor vehicle theft', 'OFFENSE_robbery',
             'OFFENSE_sex abuse', 'OFFENSE_theft f/auto',
             'OFFENSE_theft/other','SHIFT_day','SHIFT_evening',
             'SHIFT_midnight','METHOD_gun', 'METHOD_knife','METHOD_others']
```

[179]:
```
# train set and test set
x = crime_model[variables]
y = crime_model['DISTRICT']
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
```

[180]:
```
# scaling
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[181]:
```
x_train.shape
```

[181]:
```
(330177, 33)
```

[182]:
```
y_train.shape
```

[182]: (330177,)

[183]:
```
x_test.shape
```

[183]: (110059, 33)

[184]:
```
y_test.shape
```

[184]: (110059,)

Decision tree

[185]:
```
decisiontree = DecisionTreeClassifier(criterion='gini', max_depth=5,␣
↪min_samples_leaf=10, min_samples_split=10)
```

[186]:
```
decisiontree.fit(x_train,y_train)
```

[186]: DecisionTreeClassifier(max_depth=5, min_samples_leaf=10, min_samples_split=10)

Model evaluation

[187]:
```
# define a function to print score of train_set, test_set and cross validation
def traintestcross(model,x,y,x_train,y_train,x_test,y_test):
    print('train_set:        ',model.score(x_train,y_train))
    print('test_set:         ',model.score(x_test,y_test))
    print('cross validation: ',cross_val_score(model, x,␣
↪y,scoring='accuracy',cv=5).mean())
```

[188]:
```
traintestcross(decisiontree,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:         0.9317093558909312
test_set:          0.9306553757530052
cross validation:  0.9314935613865358
```

[189]:
```
# define roc auc score for multiclass
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)
    return roc_auc_score(y_test, y_pred, average=average)
```

[190]:
```
# define a function to print precision, recall, roc_auc_score and F1_score
def accuracy(y_test,y_pred):
    print('Precision:',metrics.precision_score(y_test, y_pred,average='micro'))
    print('Recall:   ',metrics.recall_score(y_test, y_pred,average='macro'))
    print('roc auc:  ',multiclass_roc_auc_score(y_test,y_pred))
    print('F1 score: ',metrics.f1_score(y_test, y_pred, average='macro'))
```

[191]:
```
y_pred = decisiontree.predict(x_test)
accuracy(y_test,y_pred)
```

```
Precision: 0.9306553757530052
Recall:    0.9396517982904319
roc auc:   0.9638808188884488
F1 score:  0.9381452806920997
```

[192]:
```
feature_impDF = pd.DataFrame()
feature_impDF["importance"] = decisiontree.feature_importances_
feature_impDF["feature"] = x.columns
feature_impDF.sort_values("importance",ascending=False,inplace=True)
feature_impDF
```

[192]:

	importance	feature
4	0.524495	ANC
5	0.176954	LATITUDE
1	0.169045	WARD
0	0.116302	LONGITUDE
3	0.013182	VOTING_PRECINCT
8	0.000020	YEAR
12	0.000001	dayofyear
22	0.000000	OFFENSE_motor vehicle theft
23	0.000000	OFFENSE_robbery
24	0.000000	OFFENSE_sex abuse
25	0.000000	OFFENSE_theft f/auto
28	0.000000	SHIFT_evening
26	0.000000	OFFENSE_theft/other
27	0.000000	SHIFT_day
20	0.000000	OFFENSE_burglary
29	0.000000	SHIFT_midnight
30	0.000000	METHOD_gun
31	0.000000	METHOD_knife
21	0.000000	OFFENSE_homicide
16	0.000000	weekday
19	0.000000	OFFENSE_assault w/dangerous weapon
18	0.000000	OFFENSE_arson
17	0.000000	quarter
15	0.000000	dayofweek
14	0.000000	weekofyear
13	0.000000	week
11	0.000000	hour
10	0.000000	DAY
9	0.000000	MONTH
7	0.000000	TIME_TO_REPORT
6	0.000000	SPAN
2	0.000000	ucr-rank
32	0.000000	METHOD_others

*Conclusion: most of the features are not important at all.

Model improvement

[193]:
```
# select important features
selected_variables=['ANC',␣
 ↪'LATITUDE','WARD','LONGITUDE','VOTING_PRECINCT','YEAR','hour']

#
x = crime_model[selected_variables]
y = crime_model['DISTRICT']
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
decisiontree.fit(x_train,y_train)
```

[193]:
```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=10, min_samples_split=10)
```

[194]:
```
traintestcross(decisiontree,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:          0.9317093558909312
test_set:           0.9306553757530052
cross validation:   0.9314935613865358
```

[195]:
```
y_pred = decisiontree.predict(x_test)
accuracy(y_test,y_pred)
# no changes after selection
```

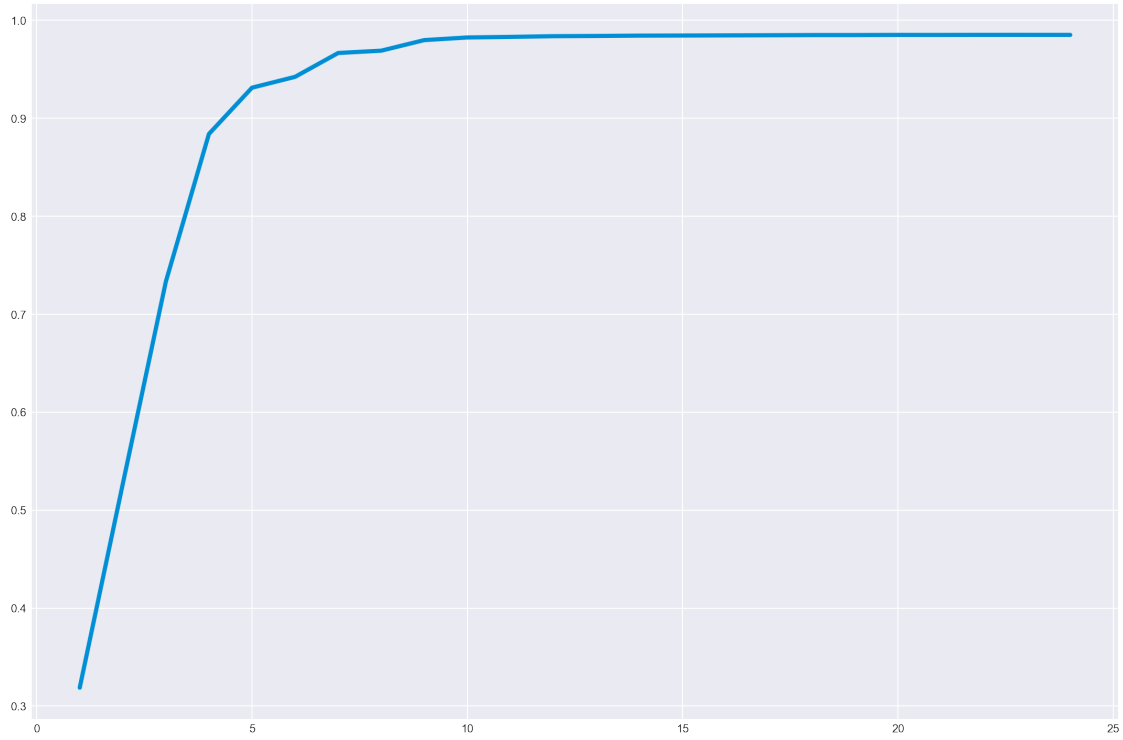
```
Precision: 0.9306553757530052
Recall:    0.9396517982904319
roc auc:   0.9638808188884488
F1 score:  0.9381452806920997
```

Find the best max_depth

[196]:
```
acc=list()
for n in range(1,25):
    decisiontree = DecisionTreeClassifier(criterion='gini', max_depth=n,␣
 ↪min_samples_leaf=10, min_samples_split=10)
    decisiontree = cross_val_score(decisiontree,x_train,y_train,cv=5)
    acc.append(decisiontree.mean())

plt.figure(figsize=(15,10))
plt.plot(range(1,25),acc)
plt.show()
# 9 is the best?
```

[196]:
```
<Figure size 1080x720 with 0 Axes>
```

[196]:
```
[<matplotlib.lines.Line2D at 0x2c55c0ca6a0>]
```



[197]:
```
# choose max_depth = 9   (avoid overfitting)
decisiontree = DecisionTreeClassifier(criterion='gini', max_depth=9,␣
 ↪min_samples_leaf=10, min_samples_split=10)
decisiontree.fit(x_train,y_train)
```

[197]:
```
DecisionTreeClassifier(max_depth=9, min_samples_leaf=10, min_samples_split=10)
```

Model evaluation for improved decision tree

[198]:
```
traintestcross(decisiontree,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:          0.9803257040920476
test_set:           0.9795473337028321
cross validation:   0.9795132613575843
```

[199]:
```
y_pred = decisiontree.predict(x_test)
```

[200]:
```
# confusion matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[14807   245    98     0   267     0     0]
 [   16 19598   370     0     0     0     0]
 [   94   347 19690   184     1     0     0]
 [    0     0   202 14168    38     0     0]
 [   61     0   173    45 14232     0     0]
 [    0     0     0     0     0 14729    33]
 [    0     0     0     0     0    77 10584]]
              precision    recall  f1-score   support

           1       0.99      0.96      0.97     15417
           2       0.97      0.98      0.98     19984
           3       0.96      0.97      0.96     20316
           4       0.98      0.98      0.98     14408
           5       0.98      0.98      0.98     14511
           6       0.99      1.00      1.00     14762
           7       1.00      0.99      0.99     10661

    accuracy                           0.98    110059
   macro avg       0.98      0.98      0.98    110059
weighted avg       0.98      0.98      0.98    110059
```

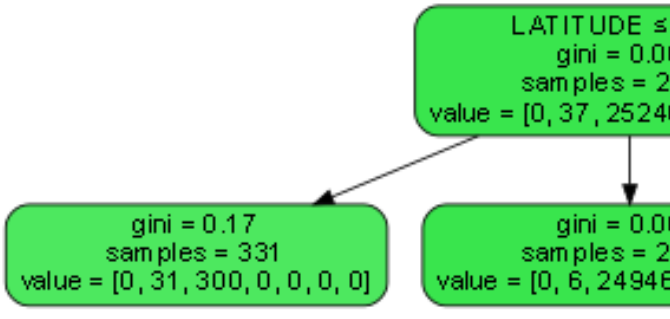
[201]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.9795473337028321
Recall:    0.9807089172725956
roc auc:   0.9886037944962623
F1 score:  0.9812412841866468
```

Decision tree visulization.

[202]:
```
from six import StringIO
from sklearn import tree
from IPython.display import Image
import pydot
dot_data = StringIO()
tree.export_graphviz(decisiontree, out_file = dot_data, feature_names =␣
 ↪crime_model[selected_variables].columns,
                     filled = True, rounded = True, special_characters = True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_png("decisiontree_district.png")
Image(graph[0].create_png())
```

[202]:


LATITUDE ≤
gini = 0.0
samples = 2
value = [0, 37, 2524
gini = 0.17
samples = 331
value = [0, 31, 300, 0, 0, 0, 0]
gini = 0.0
samples = 2
value = [0, 6, 24946

Next, we try some other methods.

kNN

[203]:
```
knn = KNeighborsClassifier(5)
```

[204]:
```
knn = knn.fit(x_train,y_train)
traintestcross(knn,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:          0.9756070228998386
test_set:           0.9633469321000554
cross validation:   0.9619953839525529
```

[205]:
```
y_pred = knn.predict(x_test)
accuracy(y_test,y_pred)
```

```
Precision: 0.9633469321000554
Recall:    0.9651336474980006
roc auc:   0.979455860104931
F1 score:  0.9653694810486233
```

Find the best n.

[206]:
```
train=[]
test=[]
for n in range(1,30):
    knn = KNeighborsClassifier(n)
    knn = knn.fit(x_train,y_train)
    train.append(1-knn.score(x_train,y_train))
    test.append(1-knn.score(x_test,y_test))
```

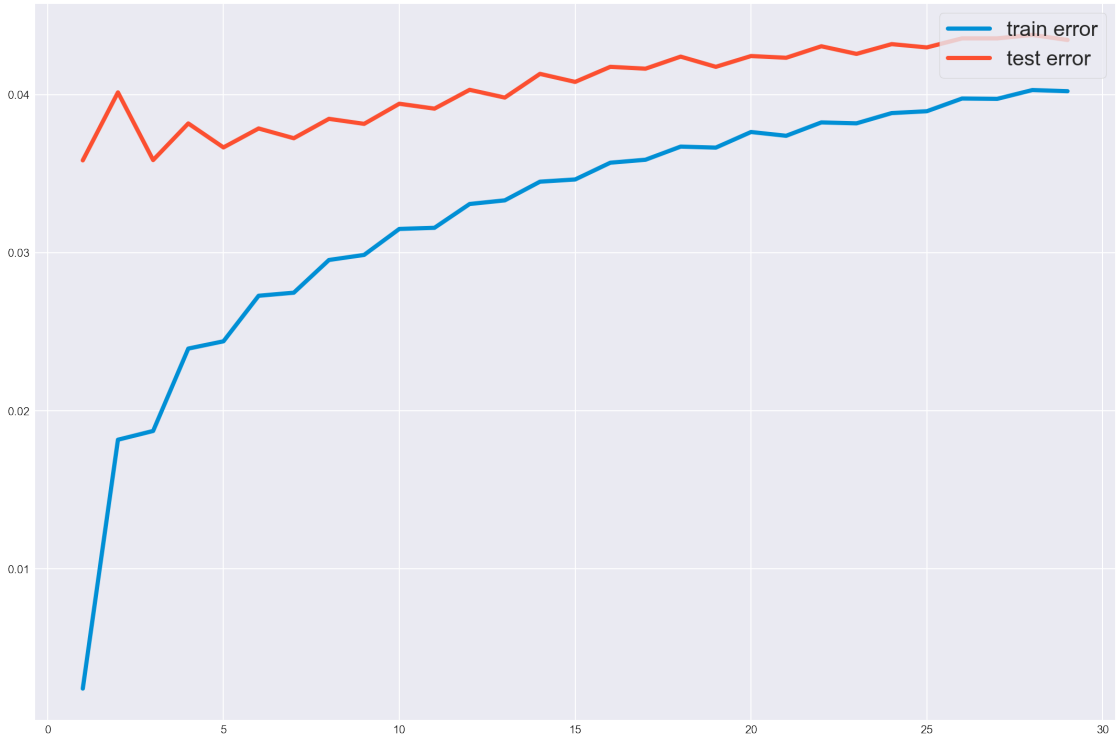
[207]:
```
plt.figure(figsize=(15,10))
plt.plot(range(1,30),train)
plt.plot(range(1,30),test)
plt.legend(['train error','test error'],loc='upper right',fontsize=20)
plt.show()
# error for different n
```

[207]:
```
<Figure size 1080x720 with 0 Axes>
```

[207]:
```
[<matplotlib.lines.Line2D at 0x2c55c4783d0>]
```

[207]:
```
[<matplotlib.lines.Line2D at 0x2c55c478880>]
```

[207]:
```
<matplotlib.legend.Legend at 0x2c55c4781f0>
```



[208]:
```
# choose n=3
knn = KNeighborsClassifier(3)
knn = knn.fit(x_train,y_train)
```

Model evaluation

[209]:
```
traintestcross(knn,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:          0.9812767091590269
test_set:           0.9641465032391717
cross validation:   0.9635036673136602
```

[210]:
```
y_pred = knn.predict(x_test)
```

[211]:
```
# confusion matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[14676   251   152     0   338     0     0]
 [  197 19411   376     0     0     0     0]
 [  224   533 19161   326    72     0     0]
 [    0     0   382 13891   135     0     0]
 [  352     0   199    90 13870     0     0]
 [    0     0     0     0     0 14588   174]
 [    0     0     0     0     0   145 10516]]
```

```
              precision    recall  f1-score   support

           1       0.95      0.95      0.95     15417
           2       0.96      0.97      0.97     19984
           3       0.95      0.94      0.94     20316
           4       0.97      0.96      0.97     14408
           5       0.96      0.96      0.96     14511
           6       0.99      0.99      0.99     14762
           7       0.98      0.99      0.99     10661

    accuracy                           0.96    110059
   macro avg       0.97      0.97      0.97    110059
weighted avg       0.96      0.96      0.96    110059
```

[212]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.9641465032391717
Recall:    0.9658524669591007
roc auc:   0.979886649934046
F1 score:  0.9660205101787851
```

Random forest ——best model

[213]:
```
forest = RandomForestClassifier(n_estimators=50,criterion='gini')
```

[214]:
```
forest.fit(x_train,y_train)
```

[214]:
```
RandomForestClassifier(n_estimators=50)
```

[215]:
```
traintestcross(forest,x,y,x_train,y_train,x_test,y_test)
```

```
train_set:         0.9977557491890713
test_set:          0.9856168055315785
cross validation:  0.9853192384409132
```

[216]:
```
y_pred = forest.predict(x_test)
```

[217]:
```
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[15076   117    59     0   165     0     0]
 [   64 19770   150     0     0     0     0]
 [   67   214 19828   151    56     0     0]
 [    0     0   177 14205    26     0     0]
 [   94     0   105    39 14273     0     0]
 [    0     0     0     0     0 14722    40]
 [    0     0     0     0     0    59 10602]]
              precision    recall  f1-score   support
```

```
           1       0.99      0.98      0.98     15417
           2       0.98      0.99      0.99     19984
           3       0.98      0.98      0.98     20316
           4       0.99      0.99      0.99     14408
           5       0.98      0.98      0.98     14511
           6       1.00      1.00      1.00     14762
           7       1.00      0.99      1.00     10661

    accuracy                           0.99    110059
   macro avg       0.99      0.99      0.99    110059
weighted avg       0.99      0.99      0.99    110059
```

The highest scores among all models I have tried.

[218]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.9856168055315785
Recall:    0.9863454172194465
roc auc:   0.9919501164565044
F1 score:  0.9865052442801879
```

[219]:
```
# testing oob score of random forests with sizes between 1 to 50 trees
OOB_Err = list(range(1,50))
for i in range(1,50):
    rfc = RandomForestClassifier(n_estimators = i, oob_score = True, n_jobs =␣
 ↪-1)
    rfc = rfc.fit(x_train,y_train)
    OOB_Err[i-1] = 1 - rfc.oob_score_
```

[220]:
```
# plotting OOB Scores
plt.figure(figsize = (15,10))
with sns.axes_style("darkgrid"):
    plt.plot(list(range(1,50)), OOB_Err)
plt.title('OOB Errors Over Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('OOB Error')
plt.show()
```

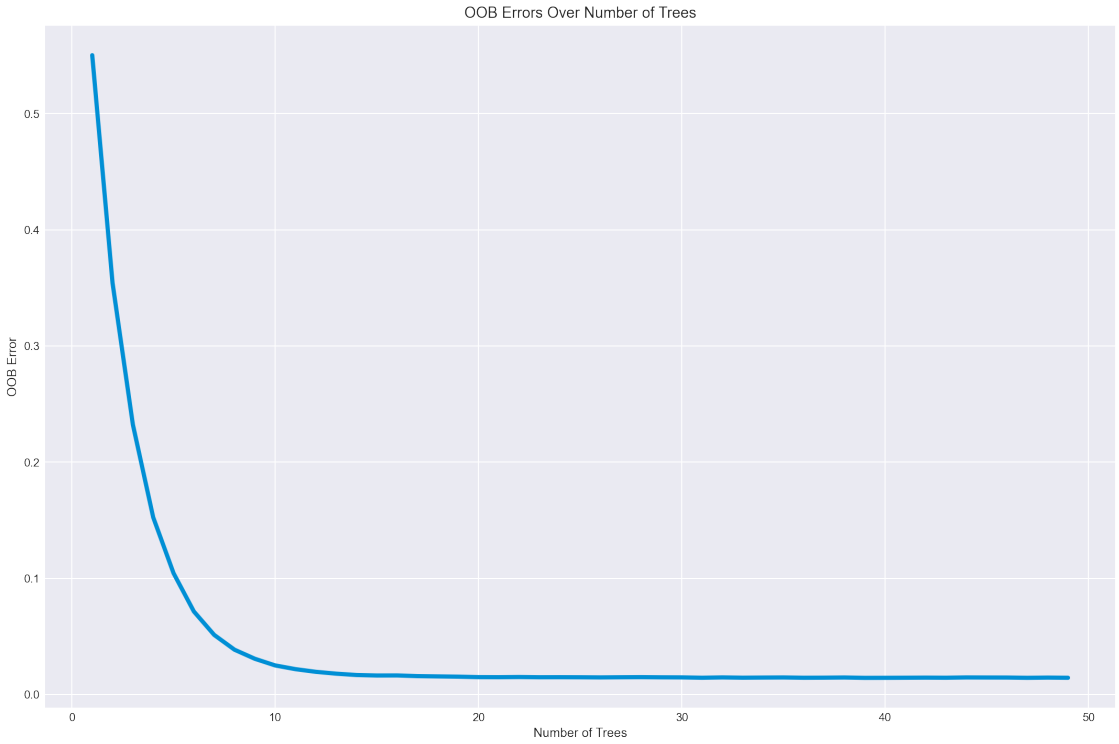
[220]: <Figure size 1080x720 with 0 Axes>

[220]: [<matplotlib.lines.Line2D at 0x2c55c46e430>]

[220]: Text(0.5, 1.0, 'OOB Errors Over Number of Trees')

[220]: Text(0.5, 0, 'Number of Trees')

[220]: Text(0, 0.5, 'OOB Error')



[221]:
```
plt.figure(figsize = (15,10))
with sns.axes_style("darkgrid"):
    plt.plot(list(range(15,50)), OOB_Err[14:])
plt.title('OOB Errors Over Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('OOB Error')
plt.show()
```

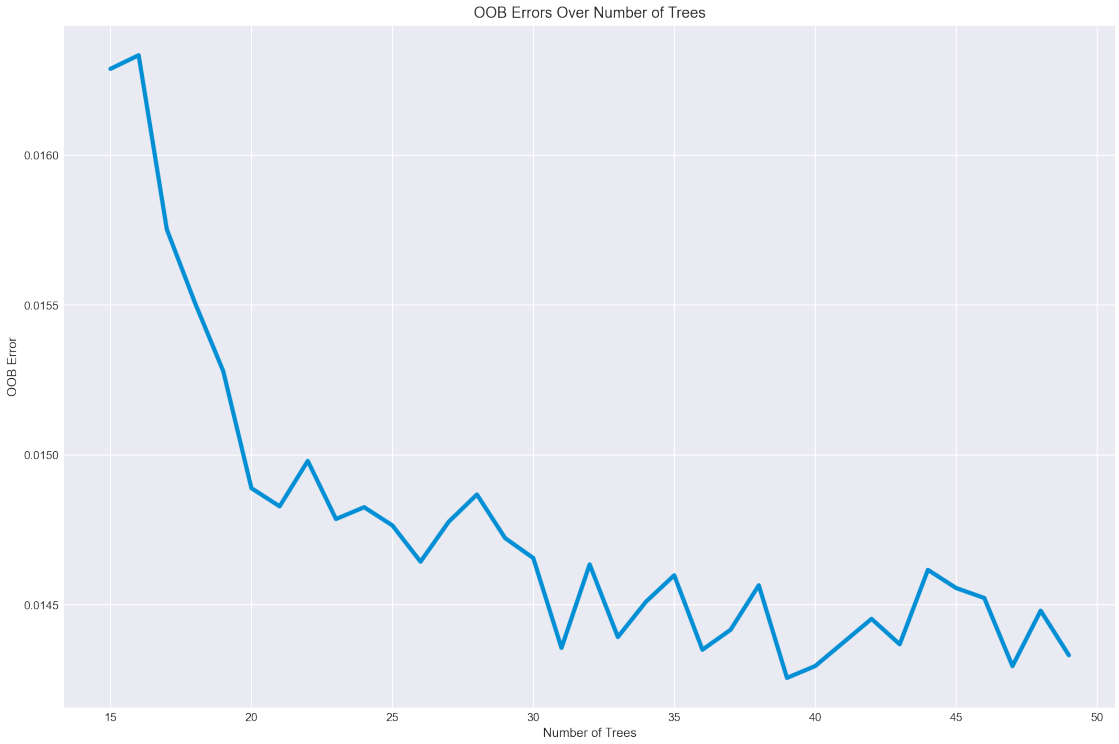
[221]:
```
<Figure size 1080x720 with 0 Axes>
```

[221]:
```
[<matplotlib.lines.Line2D at 0x2c55c453d60>]
```

[221]:
```
Text(0.5, 1.0, 'OOB Errors Over Number of Trees')
```

[221]:
```
Text(0.5, 0, 'Number of Trees')
```

[221]:
```
Text(0, 0.5, 'OOB Error')
```



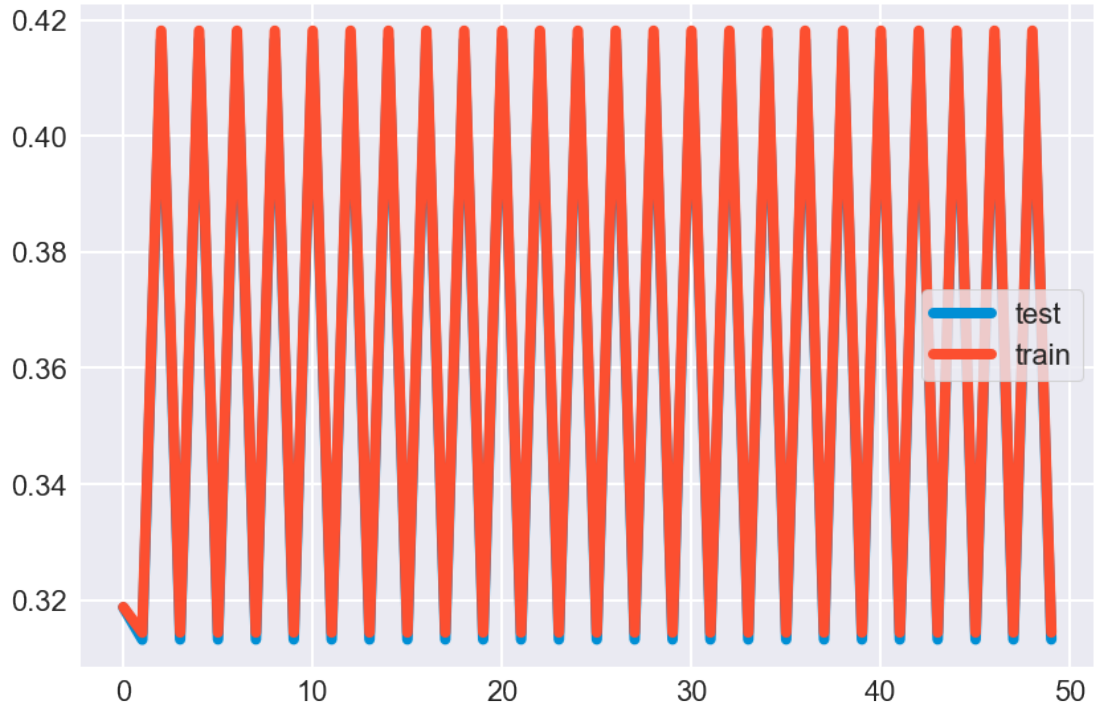
Adaboost

[222]:
```
from sklearn.ensemble import AdaBoostClassifier
adaboost = AdaBoostClassifier(n_estimators = 50)
```

[223]:
```
adaboost.fit(x_train,y_train)

y_preds_test = list(adaboost.staged_predict(x_test))
accuracy_tests = []
for y_pred in y_preds_test:
    accuracy_tests.append(accuracy_score(y_test,y_pred))

y_preds_train = list(adaboost.staged_predict(x_train))
accuracy_train = []
for y_pred in y_preds_train:
    accuracy_train.append(accuracy_score(y_train,y_pred))

df = pd.DataFrame()
df["test"] = accuracy_tests
df["train"] = accuracy_train
df.plot()
```

[223]:
```
AdaBoostClassifier()
```

[223]: <AxesSubplot:>



[224]:
```
y_pred = adaboost.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[    0  2429     0     0 12988     0     0]
 [    0 19984     0     0     0     0     0]
 [    0 16957     5     0  3354     0     0]
 [    0  2822 11066     0   520     0     0]
 [    0     4    23     0 14484     0     0]
 [    0     0     0     0 14762     0     0]
 [    0     0     0     0 10661     0     0]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00     15417
           2       0.47      1.00      0.64     19984
           3       0.00      0.00      0.00     20316
           4       0.00      0.00      0.00     14408
           5       0.26      1.00      0.41     14511
           6       0.00      0.00      0.00     14762
           7       0.00      0.00      0.00     10661

    accuracy                           0.31    110059
```

```
   macro avg       0.10      0.29      0.15    110059
weighted avg       0.12      0.31      0.17    110059
```

Bad performance.

[225]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.31322290771313566
Recall:    0.2854836362867096
roc auc:   0.5846910400142908
F1 score:  0.1499278149956788
```

Gradient Tree Boosting

[226]:
```
from sklearn.datasets import make_hastie_10_2
from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.
 ↪0,max_depth=1, random_state=0).fit(x_train, y_train)
clf.score(x_test, y_test)
```

[226]:
```
0.83276242742529
```

[227]:
```
y_pred = clf.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[14671   249   193     0   210    94     0]
 [   53 19592   339     0     0     0     0]
 [   34   925 18861   392   104     0     0]
 [    0    86   689 13480   153     0     0]
 [  456     0   447   687 12563   358     0]
 [    6  7991    26    13  4619  1870   237]
 [    0     0     0     0     1    44 10616]]
              precision    recall  f1-score   support

           1       0.96      0.95      0.96     15417
           2       0.68      0.98      0.80     19984
           3       0.92      0.93      0.92     20316
           4       0.93      0.94      0.93     14408
           5       0.71      0.87      0.78     14511
           6       0.79      0.13      0.22     14762
           7       0.98      1.00      0.99     10661

    accuracy                           0.83    110059
   macro avg       0.85      0.83      0.80    110059
weighted avg       0.84      0.83      0.80    110059
```

[228]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.83276242742529
Recall:    0.826311670773429
roc auc:   0.8988968261934144
F1 score:  0.7999988693841055
```

Neural network

[229]:
```
from sklearn.neural_network import MLPClassifier
```

[230]:
```
clf = MLPClassifier()
clf.fit(x_train, y_train)
```

[230]:
```
MLPClassifier()
```

[231]:
```
y_pred=clf.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[14832   259    42     0   284     0     0]
 [   57 19804   123     0     0     0     0]
 [  112   377 19593   167    67     0     0]
 [    0     0   381 13961    66     0     0]
 [  154     0   127    37 14193     0     0]
 [    0     0     0     0     0 14733    29]
 [    0     0     0     0     0   213 10448]]
              precision    recall  f1-score   support

           1       0.98      0.96      0.97     15417
           2       0.97      0.99      0.98     19984
           3       0.97      0.96      0.97     20316
           4       0.99      0.97      0.98     14408
           5       0.97      0.98      0.97     14511
           6       0.99      1.00      0.99     14762
           7       1.00      0.98      0.99     10661

    accuracy                           0.98    110059
   macro avg       0.98      0.98      0.98    110059
weighted avg       0.98      0.98      0.98    110059
```

[232]:
```
accuracy(y_test,y_pred)
```

```
Precision: 0.9773303409989188
Recall:    0.9775110352122605
roc auc:   0.9868271774711819
F1 score:  0.9782994728949415
```

Model comparison: Random forest > Neural network > Decision Tree > kNN > Gradient Tree

Boosting > Adaboost

3-2-2 Classification 2: View WARD as label. Use the best model—— random forest.

[233]:
```
variables = ['LONGITUDE','DISTRICT','ucr-rank', 'VOTING_PRECINCT',
             'ANC', 'LATITUDE', 'OFFENSE_code','SPAN','TIME_TO_REPORT',
             'YEAR', 'MONTH', 'DAY', 'hour', 'dayofyear', 'week',
             'weekofyear', 'dayofweek', 'weekday','quarter',
             'SHIFT_day','SHIFT_evening', 'SHIFT_midnight',
             'METHOD_gun', 'METHOD_knife','METHOD_others']
x = crime_model[variables]
y = crime_model['WARD']

#
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[234]:
```
forest = RandomForestClassifier(n_estimators=100,criterion='gini')
forest.fit(x_train,y_train)
traintestcross(forest,x,y,x_train,y_train,x_test,y_test)
y_pred = forest.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
accuracy(y_test,y_pred)
```

[234]:
```
RandomForestClassifier()

train_set:          1.0
test_set:           0.9994457518240216
cross validation:   0.924820156753005
[[15608     5     0     0     9     0     0     0]
 [    0 19966     0     0     0     0     0     0]
 [    0     0  6209     0     0     0     0     0]
 [    1     0     0 10173     2     0     0     0]
 [    3     0     0     0 15070    18     0     0]
 [    1    13     0     0     0 17106     0     0]
 [    0     0     0     0     0     2 13768     2]
 [    0     0     0     0     0     0     5 12098]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     15622
           2       1.00      1.00      1.00     19966
           3       1.00      1.00      1.00      6209
           4       1.00      1.00      1.00     10176
```

```
           5       1.00      1.00      1.00     15091
           6       1.00      1.00      1.00     17120
           7       1.00      1.00      1.00     13772
           8       1.00      1.00      1.00     12103

    accuracy                           1.00    110059
   macro avg       1.00      1.00      1.00    110059
weighted avg       1.00      1.00      1.00    110059

Precision: 0.9994457518240216
Recall:    0.9994870170786134
roc auc:   0.9997025018068644
F1 score:  0.9995155445451493
```

[235]:
```
feature_impDF = pd.DataFrame()
feature_impDF["importance"] = forest.feature_importances_
feature_impDF["feature"] = x.columns
feature_impDF.sort_values("importance",ascending=False,inplace=True)
feature_impDF
```

[235]:

	importance	feature
4	0.354949	ANC
1	0.164547	DISTRICT
0	0.161791	LONGITUDE
3	0.154479	VOTING_PRECINCT
5	0.153050	LATITUDE
6	0.004129	OFFENSE_code
2	0.001662	ucr-rank
7	0.001057	SPAN
24	0.000576	METHOD_others
8	0.000526	TIME_TO_REPORT
9	0.000463	YEAR
22	0.000441	METHOD_gun
12	0.000380	hour
13	0.000315	dayofyear
11	0.000241	DAY
21	0.000227	SHIFT_midnight
14	0.000218	week
15	0.000213	weekofyear
17	0.000155	weekday
16	0.000144	dayofweek
10	0.000123	MONTH
23	0.000100	METHOD_knife
19	0.000100	SHIFT_day
20	0.000065	SHIFT_evening
18	0.000048	quarter

*What if the variables do not contain any geograpy information ?

[236]:
```
variables = ['ucr-rank','OFFENSE_code','SPAN','TIME_TO_REPORT', 'YEAR',
             'MONTH', 'DAY', 'hour', 'dayofyear', 'week', 'weekofyear',
             'dayofweek', 'weekday','quarter',
             'SHIFT_day','SHIFT_evening',  'SHIFT_midnight','METHOD_gun',
             'METHOD_knife','METHOD_others']
```

[237]:
```
x = crime_model[variables]
y = crime_model['WARD']
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[238]:
```
forest = RandomForestClassifier(n_estimators=50,criterion='gini')
forest.fit(x_train,y_train)
traintestcross(forest,x,y,x_train,y_train,x_test,y_test)
y_pred = forest.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
accuracy(y_test,y_pred)
```

[238]:
```
RandomForestClassifier(n_estimators=50)
```

```
train_set:          0.9975740284756358
test_set:           0.22595153508572674
cross validation:   0.21755377967466333
[[3634 4304  341  808 1690 2351 1374 1120]
 [3136 8378  507  798 1853 3023 1348  923]
 [ 843 2007  408  384  645 1088  458  376]
 [1579 2263  283  929 1347 1631 1180  964]
 [2056 3310  318  877 2595 2294 2026 1615]
 [2583 4694  407  925 2035 3518 1686 1272]
 [1563 2526  245  748 1889 1895 3010 1896]
 [1366 2040  209  675 1661 1549 2207 2396]]
              precision    recall  f1-score   support

           1       0.22      0.23      0.22     15622
           2       0.28      0.42      0.34     19966
           3       0.15      0.07      0.09      6209
           4       0.15      0.09      0.11     10176
           5       0.19      0.17      0.18     15091
           6       0.20      0.21      0.20     17120
           7       0.23      0.22      0.22     13772
           8       0.23      0.20      0.21     12103

    accuracy                           0.23    110059
```

```
   macro avg       0.21      0.20      0.20    110059
weighted avg       0.22      0.23      0.22    110059

Precision: 0.22595153508572674
Recall:    0.20040157512966292
roc auc:   0.5436904437424308
F1 score:  0.19830910761565268
```

*Conclusion: random guess for this classification is $1/8 = 0.125$. It is clear that classification with non-geographic information is hard.

4.0.3 3-3 Cluster the geography by crime events

Feature selection for clustering Since we need to cluster geography by crimes, we should choose features that measure the nature of crime iteslf and assign proper values to these features.

[240]:
```
crime_model.columns
```

[240]:
```
Index(['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD', 'YEAR', 'sector',
       'ucr-rank', 'VOTING_PRECINCT', 'ANC', 'LATITUDE', 'OFFENSE_code',
       'SPAN', 'TIME_TO_REPORT', 'DATE', 'MONTH', 'DAY', 'hour', 'dayofyear',
       'week', 'weekofyear', 'dayofweek', 'weekday', 'quarter',
       'OFFENSE_arson', 'OFFENSE_assault w/dangerous weapon',
       'OFFENSE_burglary', 'OFFENSE_homicide', 'OFFENSE_motor vehicle theft',
       'OFFENSE_robbery', 'OFFENSE_sex abuse', 'OFFENSE_theft f/auto',
       'OFFENSE_theft/other', 'offensegroup_property', 'offensegroup_violent',
       'SHIFT_day', 'SHIFT_evening', 'SHIFT_midnight', 'METHOD_gun',
       'METHOD_knife', 'METHOD_others'],
      dtype='object')
```

[241]:
```
features = ['LONGITUDE','LATITUDE','ucr-rank','SHIFT_day','SHIFT_evening',
            'SHIFT_midnight', 'METHOD_gun', 'METHOD_knife','METHOD_others',
            'OFFENSE_arson','OFFENSE_assault w/dangerous weapon',
            'OFFENSE_burglary', 'OFFENSE_homicide','OFFENSE_motor vehicle␣
 ↪theft',
            'OFFENSE_robbery', 'OFFENSE_sex abuse','OFFENSE_theft f/auto',
            'OFFENSE_theft/other']

crime_model[features]
# show
```

[241]:

	LONGITUDE	LATITUDE	ucr-rank	SHIFT_day	SHIFT_evening	\
2	-77.020913	38.902518	6	1	0	
9	-77.005025	38.876476	6	0	1	
10	-77.019909	38.899059	3	0	0	
23	-77.022433	38.898446	7	1	0	
27	-76.984577	38.889799	6	0	1	
…	…	…	…	…	…	

```
449163 -76.988516  38.842541         8          1              0
449169 -77.012634  38.831298         3          0              0
449178 -77.008913  38.819381         6          1              0
449184 -76.978424  38.844765         8          0              1
449192 -76.991523  38.830632         7          0              1

        SHIFT_midnight  METHOD_gun  METHOD_knife  METHOD_others  \
2                    0           0             0              1
9                    0           0             0              1
10                   1           0             0              1
23                   0           0             0              1
27                   0           0             0              1
…                  …           …             …              …
449163               0           0             0              1
449169               1           1             0              0
449178               0           0             0              1
449184               0           0             0              1
449192               0           0             0              1

        OFFENSE_arson  OFFENSE_assault w/dangerous weapon  OFFENSE_burglary  \
2                   0                                   0                 0
9                   0                                   0                 0
10                  0                                   1                 0
23                  0                                   0                 0
27                  0                                   0                 0
…                 …                                   …                 …
449163              0                                   0                 0
449169              0                                   1                 0
449178              0                                   0                 0
449184              0                                   0                 0
449192              0                                   0                 0

        OFFENSE_homicide  OFFENSE_motor vehicle theft  OFFENSE_robbery  \
2                      0                            0                0
9                      0                            0                0
10                     0                            0                0
23                     0                            0                0
27                     0                            0                0
…                    …                            …                …
449163                 0                            1                0
449169                 0                            0                0
449178                 0                            0                0
449184                 0                            1                0
449192                 0                            0                0

        OFFENSE_sex abuse  OFFENSE_theft f/auto  OFFENSE_theft/other
2                       0                     0                    1
```

```
9                    0                    0                    1
10                   0                    0                    0
23                   0                    1                    0
27                   0                    0                    1
…                  …                    …                    …
449163               0                    0                    0
449169               0                    0                    0
449178               0                    0                    1
449184               0                    0                    0
449192               0                    1                    0

[440236 rows x 18 columns]
```

[242]:
```
# scaling
x = preprocessing.scale(crime_model[features])
```

[243]:
```
x.shape
```

[243]:
```
(440236, 18)
```

KMeans To cluster geography by crimes, there are two main factors of crimes. One is the amount of crimes in a area and another is the degree of dangerous of different crime. The number of crimes is the most important factor since a crime-ridden area is certainly a lot more dangerous than a crime-free area. Degree of dangerous can be measured by 'ucr-rank', 'METHOD', 'SHIFT' and 'OFFENSE' since an asssult with a gun is always more dangerous than a theft with neither a gun nor a knife.

[244]:
```
# define weight (according to the meaning of features)
x = x*[3,3,1,0.1,0.1,0.1,0.2,0.2,0.2,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.
↪05]
# importance: longitude=latitude > ucr-rank > method > shift> offense
```

[245]:
```
from sklearn.cluster import KMeans
crime_KMeans = crime_model
```

[246]:
```
num_clusters = 10
km = KMeans(n_clusters=num_clusters,  n_init=1, verbose=1)
km = km.fit(x)
```

```
Initialization complete
Iteration 0, inertia 1658304.611528151
Iteration 1, inertia 1262807.7192780536
Iteration 2, inertia 1205261.6657226293
Iteration 3, inertia 1174699.2743583617
Iteration 4, inertia 1160497.3003376275
Iteration 5, inertia 1153064.4401789529
Iteration 6, inertia 1146743.9405481515
Iteration 7, inertia 1142859.3409149693
```

```
Iteration 8, inertia 1140279.6621215153
Iteration 9, inertia 1138966.9612407757
Iteration 10, inertia 1138462.4633769032
Iteration 11, inertia 1138323.5433899977
Iteration 12, inertia 1138261.6634867226
Iteration 13, inertia 1138245.760850189
Converged at iteration 13: center shift 9.625191131692676e-05 within tolerance
0.00010651388888887139.
```

[247]:
```
crime_KMeans['cluster'] = km.labels_
crime_KMeans.groupby('cluster').mean()
```

[247]:

cluster	CENSUS_TRACT	LONGITUDE	DISTRICT	WARD	YEAR	ucr-rank	\
0	5834.956405	-77.022284	2.511719	3.154054	2014.357237	6.094151	
1	8112.401719	-76.992215	2.213372	5.854506	2014.232015	5.951361	
2	7596.724002	-76.975722	6.569457	7.736379	2013.909543	5.644308	
3	3450.849246	-77.020572	3.990343	4.088389	2014.114853	6.024049	
4	1018.854497	-77.076682	2.000000	3.066393	2014.050958	6.255160	
5	8647.509843	-76.937269	5.999748	6.999495	2013.983205	5.718016	
6	3180.671738	-77.031147	3.359866	1.562200	2013.941970	5.963101	
7	9724.396625	-76.999270	6.999120	7.999707	2013.385082	5.407435	
8	4866.590138	-77.050916	2.130058	2.023041	2013.990722	6.091855	
9	9457.598710	-76.983965	4.889542	5.000000	2014.406451	6.064919	

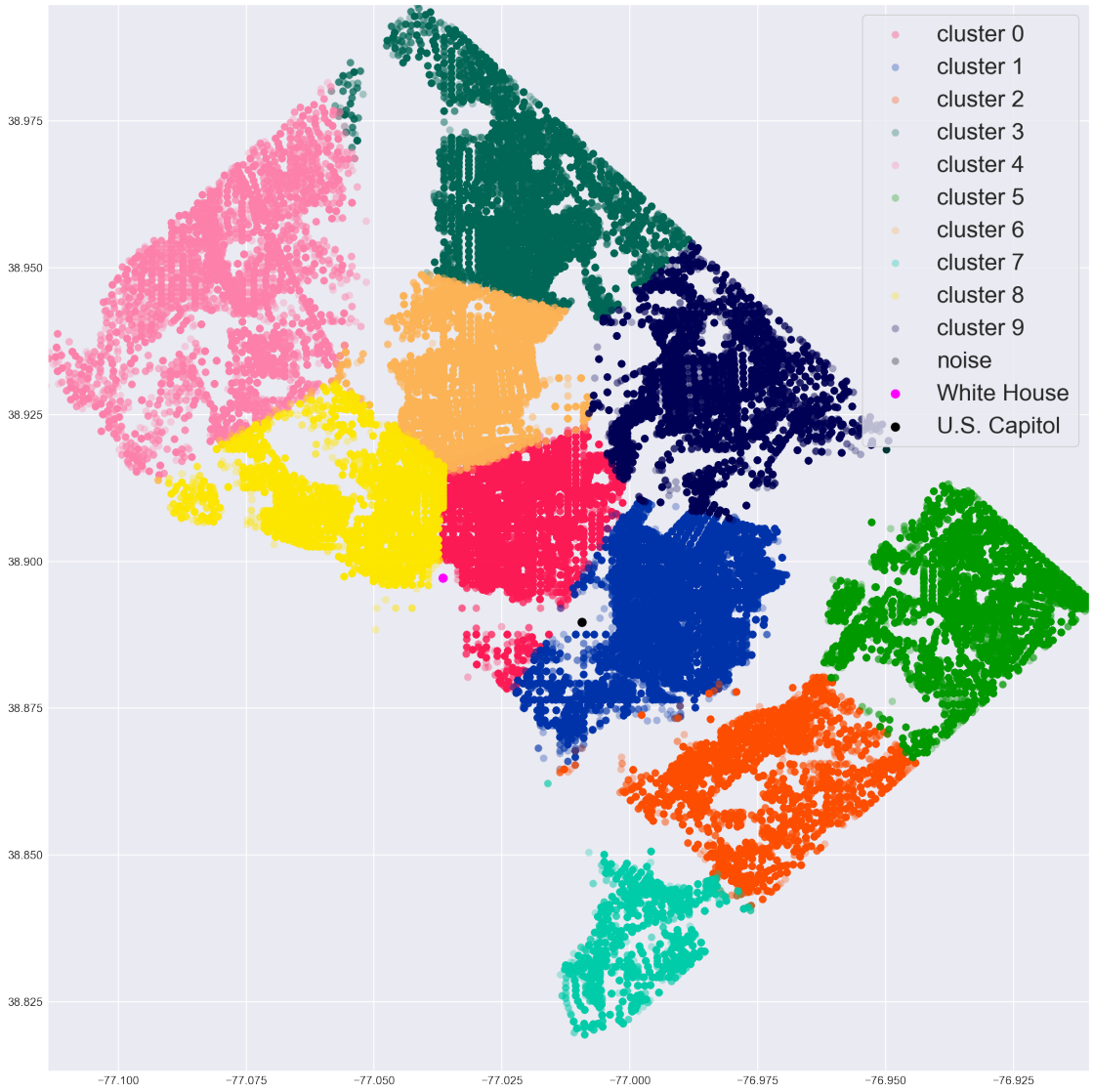
cluster	VOTING_PRECINCT	LATITUDE	SPAN	TIME_TO_REPORT	…	\
					…	
0	70.303596	38.907152	120183.807446	151444.064206	…	
1	91.165956	38.892114	162173.227970	134487.366690	…	
2	117.739392	38.860587	99252.056313	118029.302150	…	
3	58.406265	38.961866	135651.179350	125988.836861	…	
4	31.282436	38.946686	257146.077820	165521.192914	…	
5	101.510302	38.892682	96028.727584	87775.697251	…	
6	38.813670	38.929595	78369.910567	138527.266903	…	
7	123.389924	38.834451	101162.369773	97153.028467	…	
8	15.205620	38.908839	178932.113559	185211.428563	…	
9	74.713450	38.925664	127794.503376	134099.554406	…	

cluster	OFFENSE_theft f/auto	OFFENSE_theft/other	offensegroup_property	\
0	0.309961	0.460100	0.879988	
1	0.262450	0.410499	0.841398	
2	0.187128	0.268625	0.729680	
3	0.322046	0.302668	0.823943	
4	0.337910	0.469499	0.955722	
5	0.174559	0.276982	0.725656	
6	0.315536	0.386662	0.829220	

```
7                    0.135534             0.245879              0.673661
8                    0.238598             0.581950              0.928552
9                    0.310870             0.339473              0.843014

         offensegroup_violent  SHIFT_day  SHIFT_evening  SHIFT_midnight  \
cluster
0                    0.120012   0.345376       0.441018        0.213606
1                    0.158602   0.400903       0.428155        0.170942
2                    0.270320   0.377163       0.400409        0.222428
3                    0.176057   0.407801       0.413445        0.178754
4                    0.044278   0.448903       0.469453        0.081644
5                    0.274344   0.349807       0.420751        0.229442
6                    0.170780   0.371062       0.427428        0.201510
7                    0.326339   0.357251       0.404255        0.238494
8                    0.071448   0.375136       0.464379        0.160485
9                    0.156986   0.400137       0.385644        0.214219

         METHOD_gun  METHOD_knife  METHOD_others
cluster
0          0.030904      0.019717       0.949379
1          0.051399      0.028914       0.919688
2          0.119972      0.051094       0.828934
3          0.064810      0.031010       0.904180
4          0.009491      0.005759       0.984749
5          0.122201      0.050707       0.827093
6          0.043002      0.031470       0.925529
7          0.152605      0.062167       0.785229
8          0.011356      0.009525       0.979119
9          0.063052      0.027889       0.909058

[10 rows x 36 columns]
```

Visualization

[248]:
```
# this function is defined in Outlier Detection And Handling step
see_cluster(crime_KMeans,9,0,1)
```



Model evaluation and improvement

[249]:
```
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import pairwise_distances
labels = km.labels_
print('DBI score:',davies_bouldin_score(x, labels))
```

```
DBI score: 0.9534545947829038
```

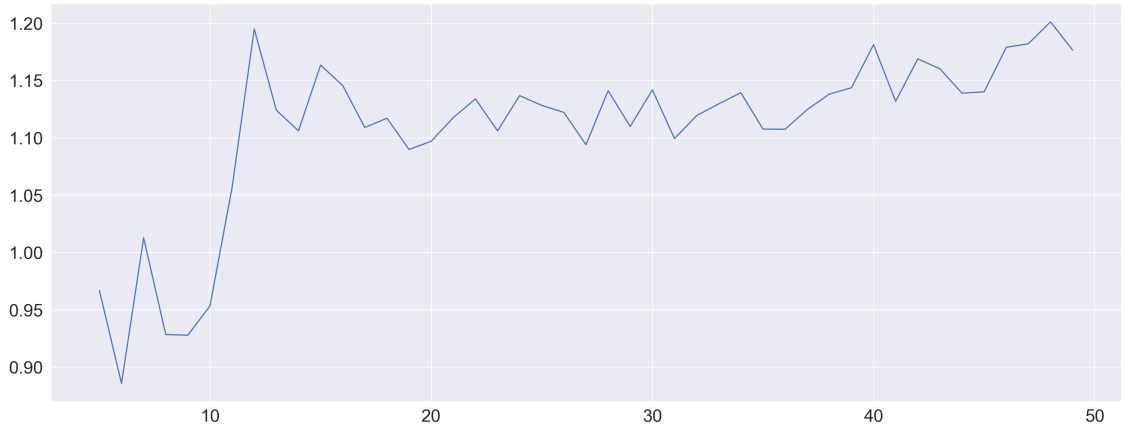
[252]:
```
DBI=[]
for i in range(5,50):
    km = KMeans(n_clusters=i, n_init=1, verbose=1)
    km = km.fit(x)
    labels=km.labels_
```

```
    DBI.append(davies_bouldin_score(x,labels))
```

DBI scores for different number of clusters

[253]:
```
plt.figure(figsize=(20,8))
plt.plot(range(5,50),DBI)
plt.show()
# n = 6 is the best?
```

[253]:



Let's see the case when the number of clusters is 6.

[254]:
```
num_clusters = 6
km = KMeans(n_clusters=num_clusters,  n_init=1, verbose=1)
km.fit(x)
```

```
Initialization complete
Iteration 0, inertia 2928981.224287721
Iteration 1, inertia 2118949.267120334
Iteration 2, inertia 1929778.3558270647
Iteration 3, inertia 1828109.8371348195
Iteration 4, inertia 1787878.9053991467
Iteration 5, inertia 1777078.0704753194
Iteration 6, inertia 1767971.4028330944
Iteration 7, inertia 1758241.2025561335
Iteration 8, inertia 1754760.463187591
Iteration 9, inertia 1753956.431396559
Iteration 10, inertia 1753851.5303511359
Iteration 11, inertia 1753834.0572488501
Converged at iteration 11: center shift 3.0031466002442076e-05 within tolerance
0.00010651388888887139.
```

[254]:
```
KMeans(n_clusters=6, n_init=1, verbose=1)
```

[255]:
```
crime_KMeans['cluster'] = km.labels_
crime_KMeans.groupby('cluster').mean()
```

[255]:

cluster	CENSUS_TRACT	LONGITUDE	DISTRICT	WARD	YEAR	ucr-rank	\
0	8450.219790	-76.991344	3.139926	5.568605	2014.307947	5.976669	
1	935.665595	-77.075781	2.002038	3.013875	2014.016932	6.243631	
2	8312.033493	-76.987250	6.511874	7.805900	2013.702660	5.508133	
3	4853.372935	-77.034340	2.497929	2.035645	2014.120672	6.079894	
4	8641.185795	-76.939221	5.978509	6.957018	2014.068027	5.766519	
5	3913.345824	-77.019023	4.039403	3.744231	2014.082526	6.010381	

cluster	VOTING_PRECINCT	LATITUDE	SPAN	TIME_TO_REPORT	…	\
0	80.912843	38.902687	142499.693994	137388.598426	…	
1	28.605040	38.942772	252031.536843	173685.861096	…	
2	120.669220	38.850999	108957.701733	116121.343179	…	
3	46.152002	38.912609	122992.597763	158114.373546	…	
4	103.098619	38.891375	97280.938753	90662.542324	…	
5	54.872872	38.951206	124562.496653	122418.711071	…	

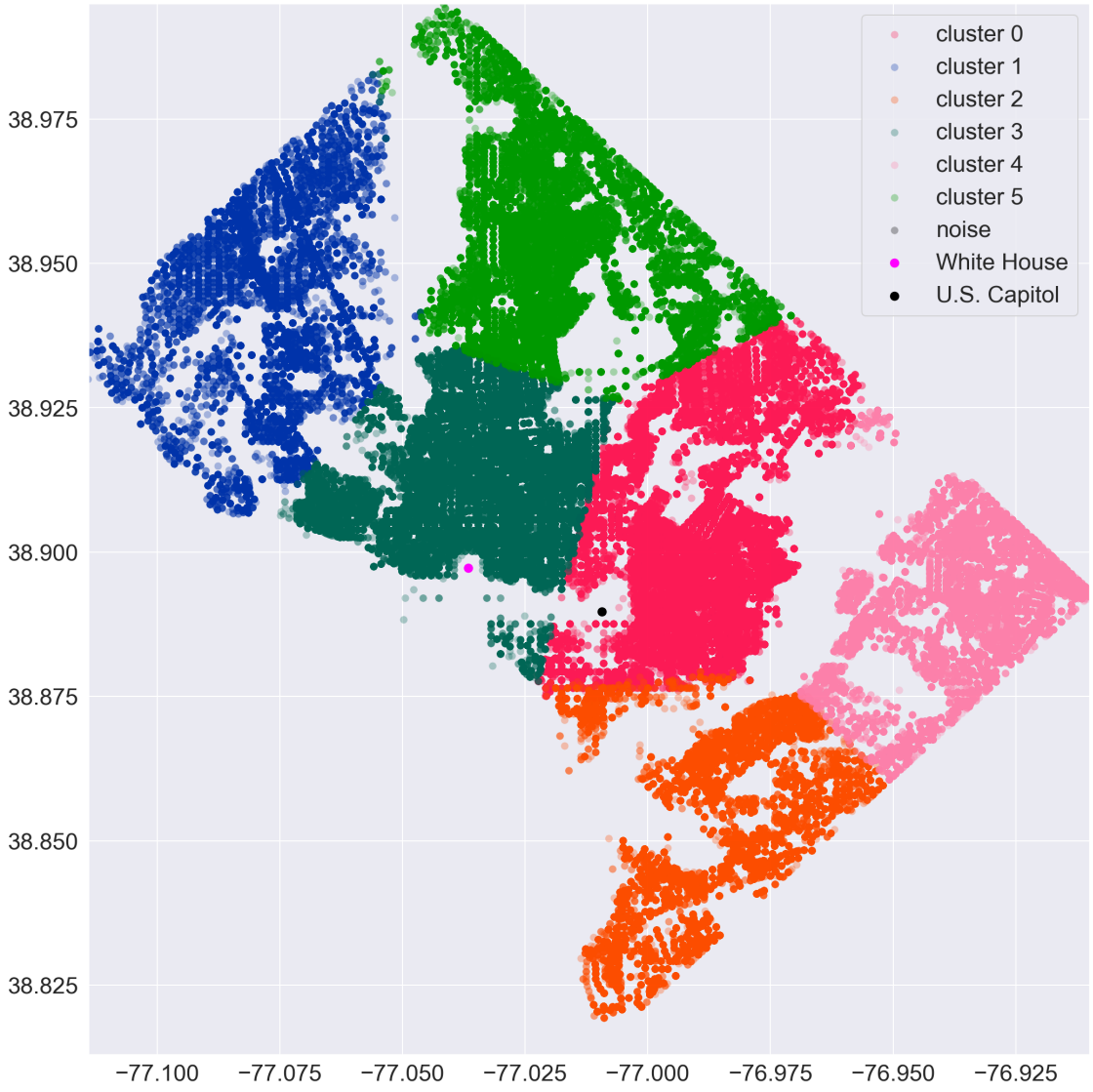
cluster	OFFENSE_theft f/auto	OFFENSE_theft/other	offensegroup_property	\
0	0.274588	0.391448	0.838713	
1	0.327702	0.475073	0.954339	
2	0.160981	0.262760	0.700344	
3	0.293652	0.487364	0.888005	
4	0.186185	0.283551	0.738577	
5	0.324228	0.319133	0.826636	

cluster	offensegroup_violent	SHIFT_day	SHIFT_evening	SHIFT_midnight	\
0	0.161287	0.394820	0.413171	0.192009	
1	0.045661	0.450929	0.462609	0.086462	
2	0.299656	0.366179	0.404001	0.229820	
3	0.111995	0.355927	0.447796	0.196278	
4	0.261423	0.358009	0.417591	0.224400	
5	0.173364	0.402059	0.416178	0.181763	

cluster	METHOD_gun	METHOD_knife	METHOD_others
0	0.054494	0.029291	0.916215
1	0.009446	0.006428	0.984126
2	0.133545	0.057615	0.808840
3	0.025638	0.017720	0.956642

```
4          0.117460      0.047871      0.834669
5          0.058739      0.031804      0.909458

[6 rows x 36 columns]
```

[256]:
```
see_cluster(crime_KMeans,5,0,1)
```



*Comment: Although n=6 has the lowest DBI score, the visulazition for n = 6 is not as good as the case of n = 10. I prefer to choose 10 clusters rather than 6. In general, KMeans has great performance for this clustering task.

DBSCAN At first, I tried to use DBSCAN on all the data, but it failed due to lack of memory. Therefore, I decided to cluster(with DBSCAN) according to the data of different years.

[257]:
```
from sklearn.cluster import DBSCAN
```

[258]:
```
# year 2021
crime_DBSCAN = crime_model[crime_model.YEAR==2021]
x_21= crime_model[crime_model.YEAR==2021][['LONGITUDE','LATITUDE']]
x_21 = preprocessing.scale(x_21)
```

I have tried many times, and eps =0.2 has a good effect.

[259]:
```
# eps = 0.2
db = DBSCAN(eps=0.2, min_samples=50).fit(x_21)
crime_DBSCAN['cluster'] = db.labels_
crime_DBSCAN.groupby('cluster').mean()
```

[259]:

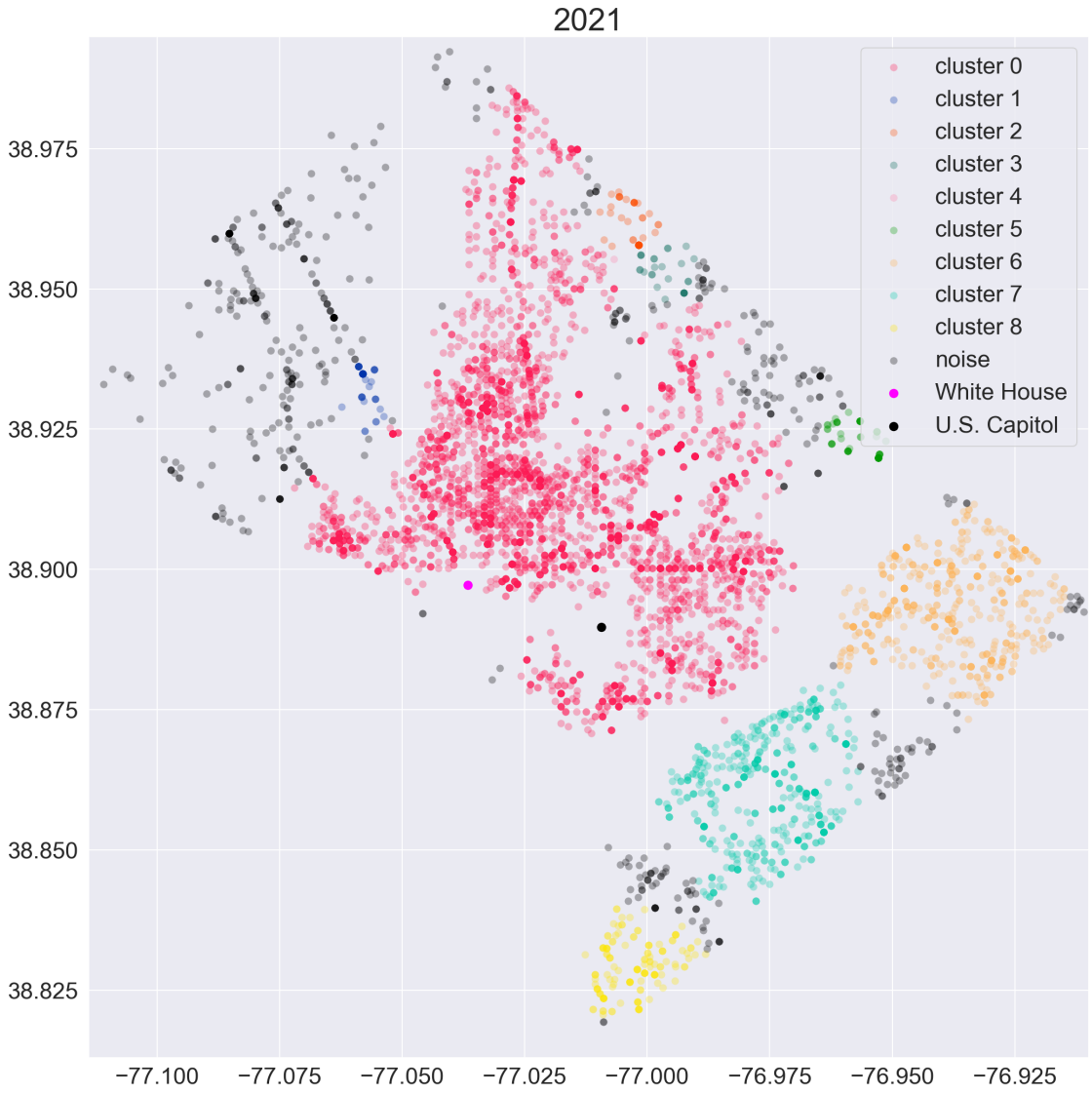
cluster	CENSUS_TRACT	LONGITUDE	DISTRICT	WARD	YEAR	ucr-rank	\
-1	4688.668582	-77.029642	3.662835	4.505747	2021.0	6.212644	
0	5708.631362	-77.018082	3.025450	3.611568	2021.0	6.230077	
1	1039.117647	-77.057585	2.000000	3.000000	2021.0	6.000000	
2	9352.078431	-77.002855	4.000000	4.000000	2021.0	6.392157	
3	9508.185185	-76.996363	4.000000	4.814815	2021.0	6.333333	
4	9508.000000	-77.006772	4.000000	5.000000	2021.0	7.000000	
5	9344.262295	-76.956677	5.000000	5.000000	2021.0	6.180328	
6	8644.320388	-76.937965	6.000000	7.000000	2021.0	5.955340	
7	7612.288100	-76.975501	6.553236	7.697286	2021.0	5.818372	
8	9943.594406	-77.002692	7.000000	8.000000	2021.0	5.377622	

cluster	VOTING_PRECINCT	LATITUDE	SPAN	TIME_TO_REPORT	…	\
-1	59.319923	38.922771	72686.524904	171557.873563	…	
0	61.852185	38.915618	143430.898972	184706.808226	…	
1	33.529412	38.933193	214723.509804	102932.490196	…	
2	64.764706	38.961000	77940.862745	-38269.607843	…	
3	65.814815	38.953098	81944.851852	-20603.296296	…	
4	44.000000	38.955281	57613.000000	16052.000000	…	
5	127.852459	38.922890	91952.147541	27194.524590	…	
6	101.279612	38.893217	53088.714563	128368.947573	…	
7	117.311065	38.860575	45375.162839	57397.453027	…	
8	124.363636	38.829232	8014.951049	149746.384615	…	

cluster	OFFENSE_theft f/auto	OFFENSE_theft/other	offensegroup_property	\
-1	0.386973	0.325670	0.862069	
0	0.311311	0.430077	0.893573	
1	0.117647	0.764706	0.960784	
2	0.411765	0.372549	0.921569	
3	0.296296	0.444444	0.888889	
4	1.000000	0.000000	1.000000	

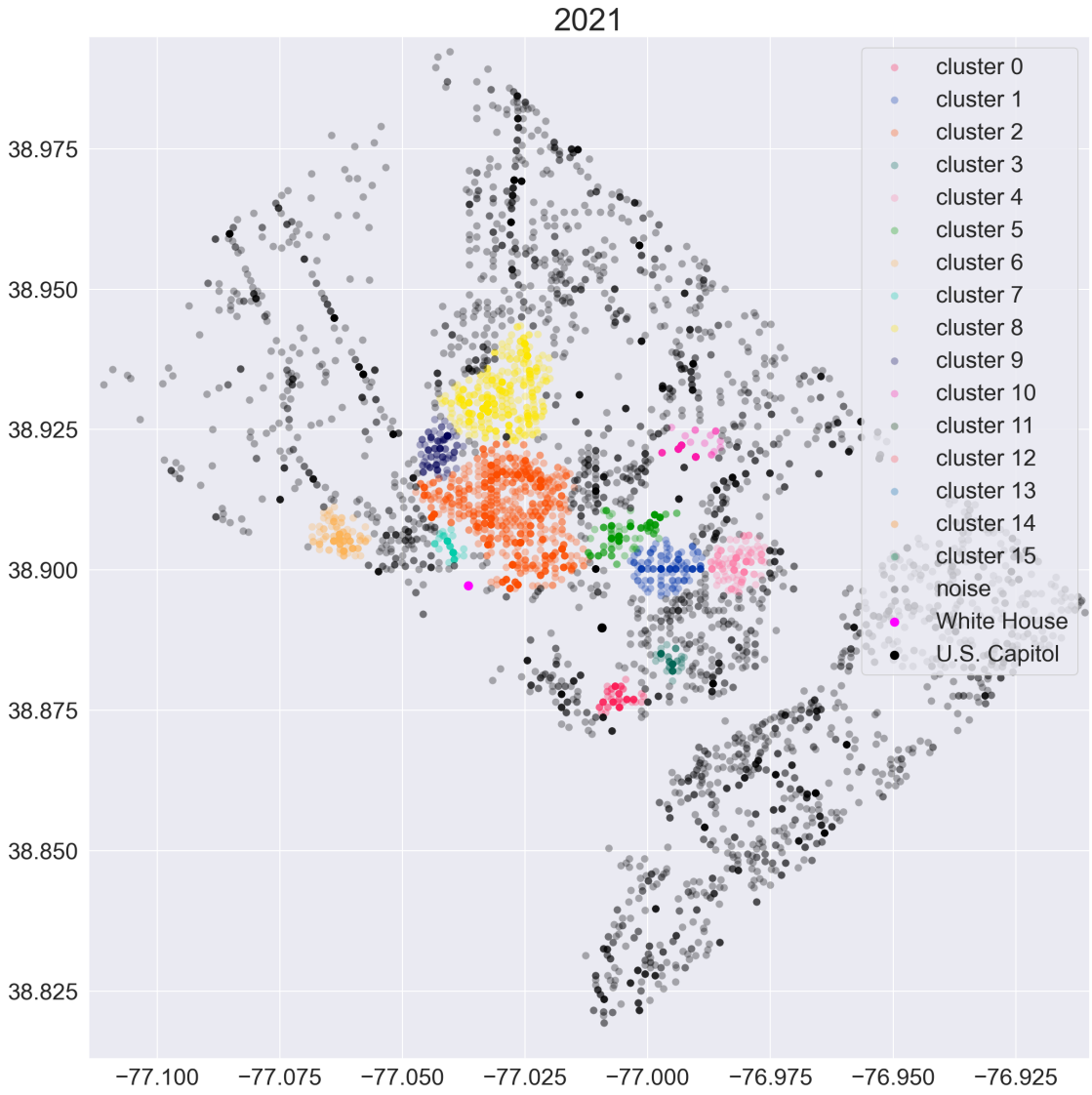
```
 5                  0.442623             0.377049              0.868852
 6                  0.225243             0.244660              0.737864
 7                  0.254697             0.300626              0.745303
 8                  0.139860             0.279720              0.608392

         offensegroup_violent  SHIFT_day  SHIFT_evening  SHIFT_midnight  \
cluster
-1                   0.137931   0.398467       0.463602        0.137931
 0                   0.106427   0.385347       0.479949        0.134704
 1                   0.039216   0.470588       0.509804        0.019608
 2                   0.078431   0.333333       0.549020        0.117647
 3                   0.111111   0.407407       0.444444        0.148148
 4                   0.000000   0.000000       1.000000        0.000000
 5                   0.131148   0.508197       0.442623        0.049180
 6                   0.262136   0.345631       0.438835        0.215534
 7                   0.254697   0.367432       0.382046        0.250522
 8                   0.391608   0.300699       0.349650        0.349650

         METHOD_gun  METHOD_knife  METHOD_others
cluster
-1         0.086207      0.028736       0.885057
 0         0.047301      0.020308       0.932391
 1         0.000000      0.039216       0.960784
 2         0.058824      0.000000       0.941176
 3         0.074074      0.000000       0.925926
 4         0.000000      0.000000       1.000000
 5         0.081967      0.000000       0.918033
 6         0.165049      0.031068       0.803883
 7         0.167015      0.027140       0.805846
 8         0.286713      0.055944       0.657343

[10 rows x 36 columns]
```

[260]:
```
see_cluster(crime_DBSCAN,8,2021,1)
```



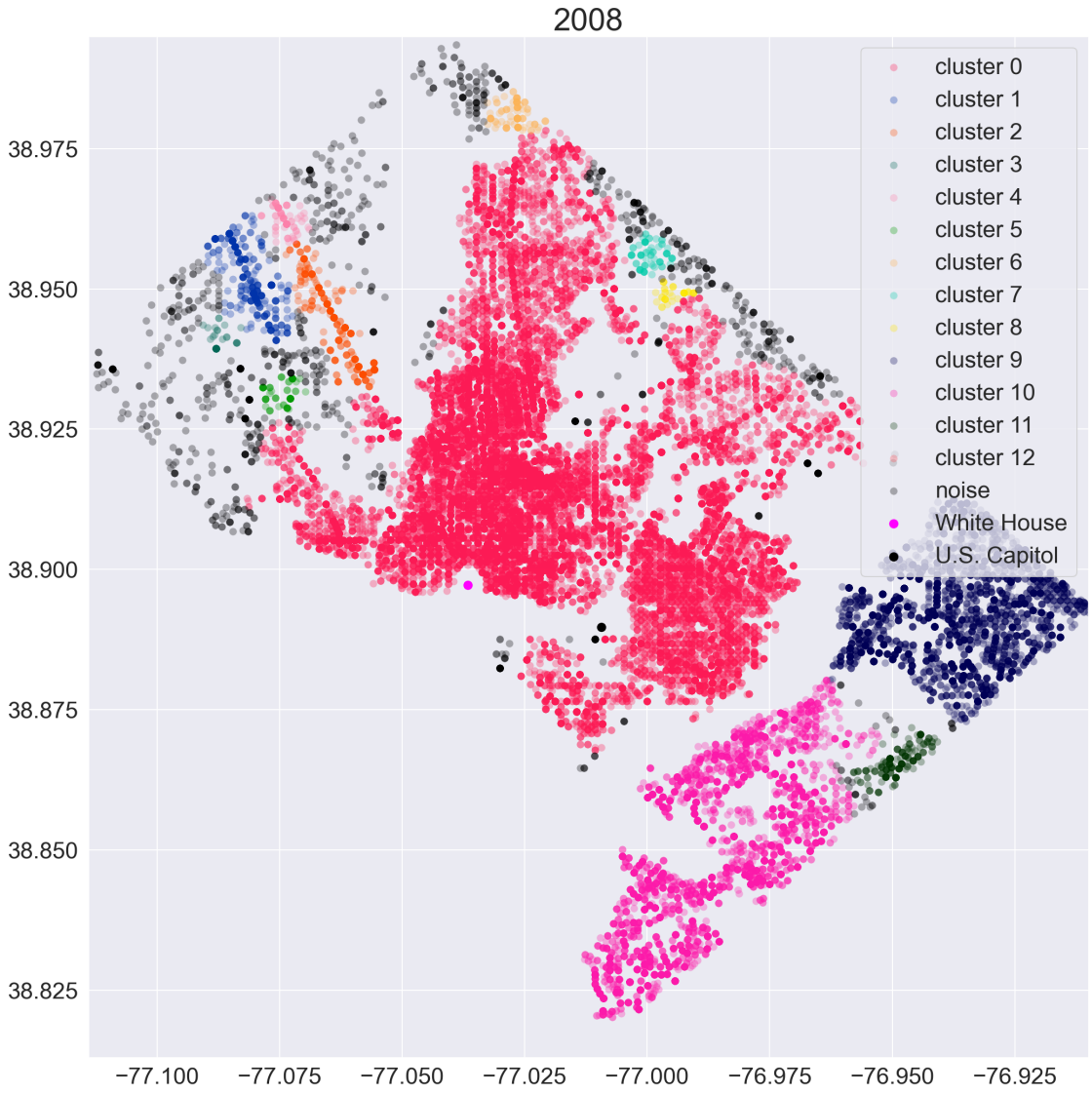
Considering only the intensity of crimes, the clustering effect is good.
Now, let's find crime-ridden areas, which can be done by reducing the value of eps.

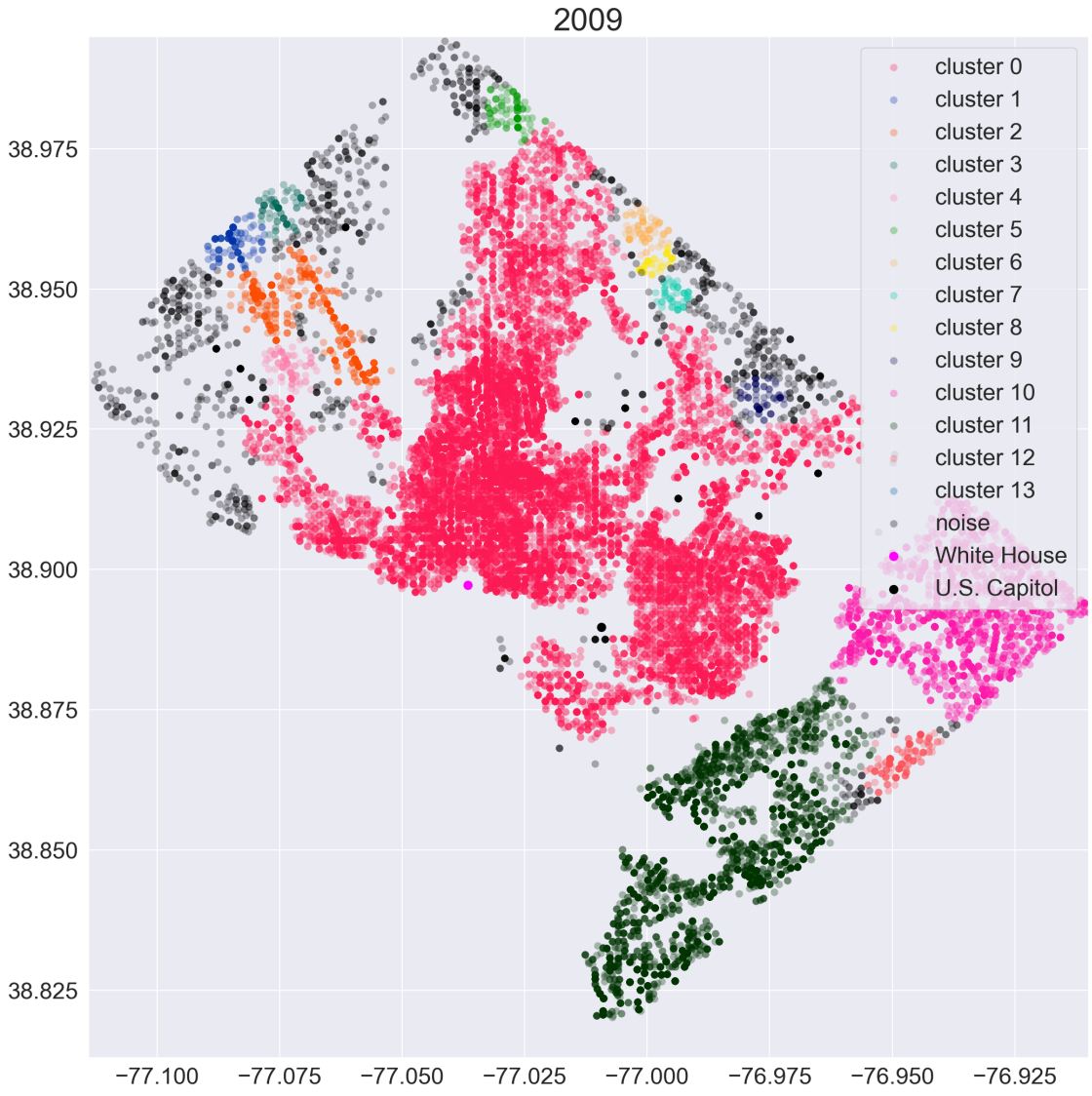
[261]:
```
# eps = 0.1
db = DBSCAN(eps=0.1, min_samples=50).fit(x_21)
crime_DBSCAN['cluster'] = db.labels_
see_cluster(crime_DBSCAN,15,2021,1)
# crime-ridden areas
```

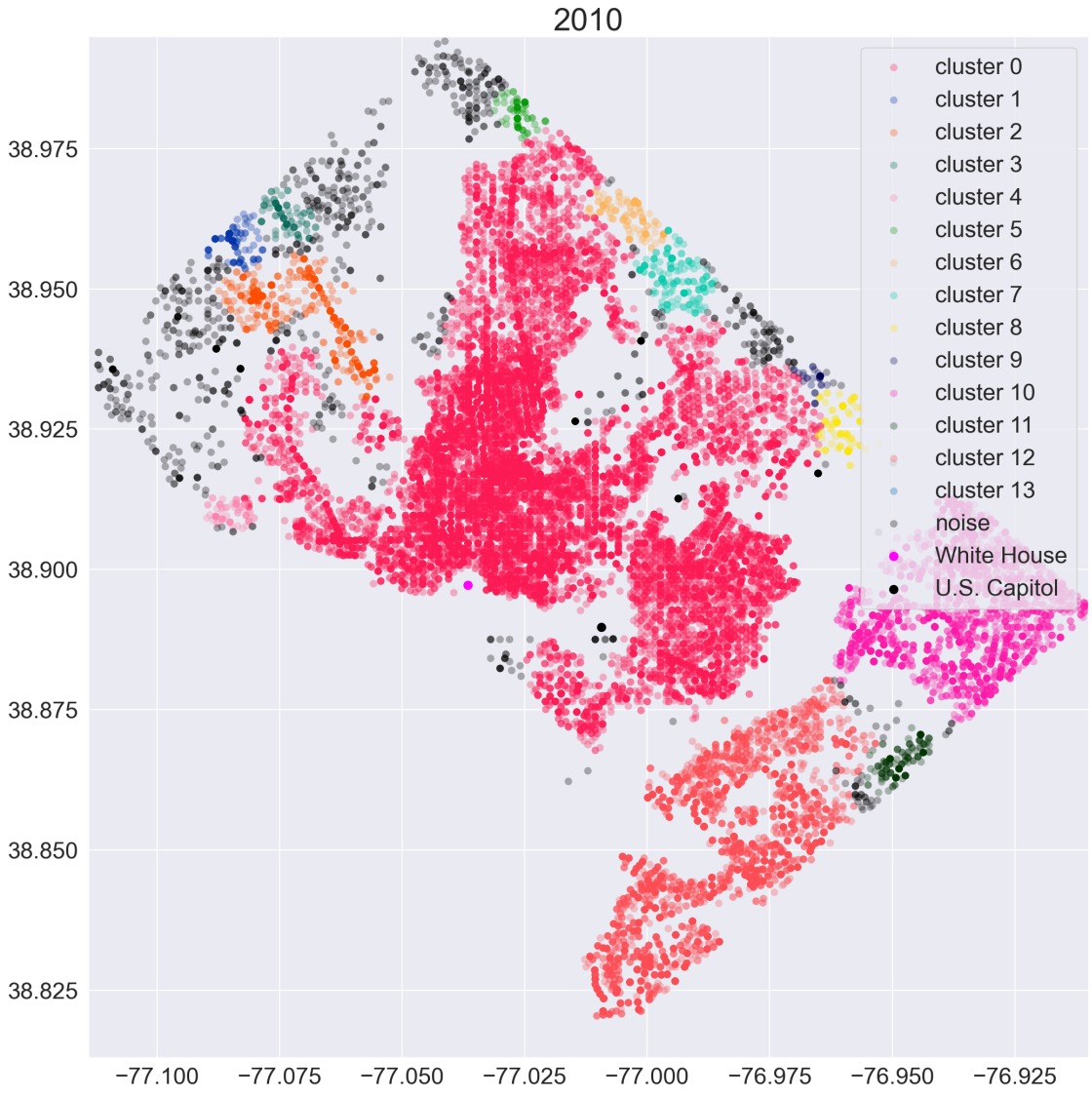


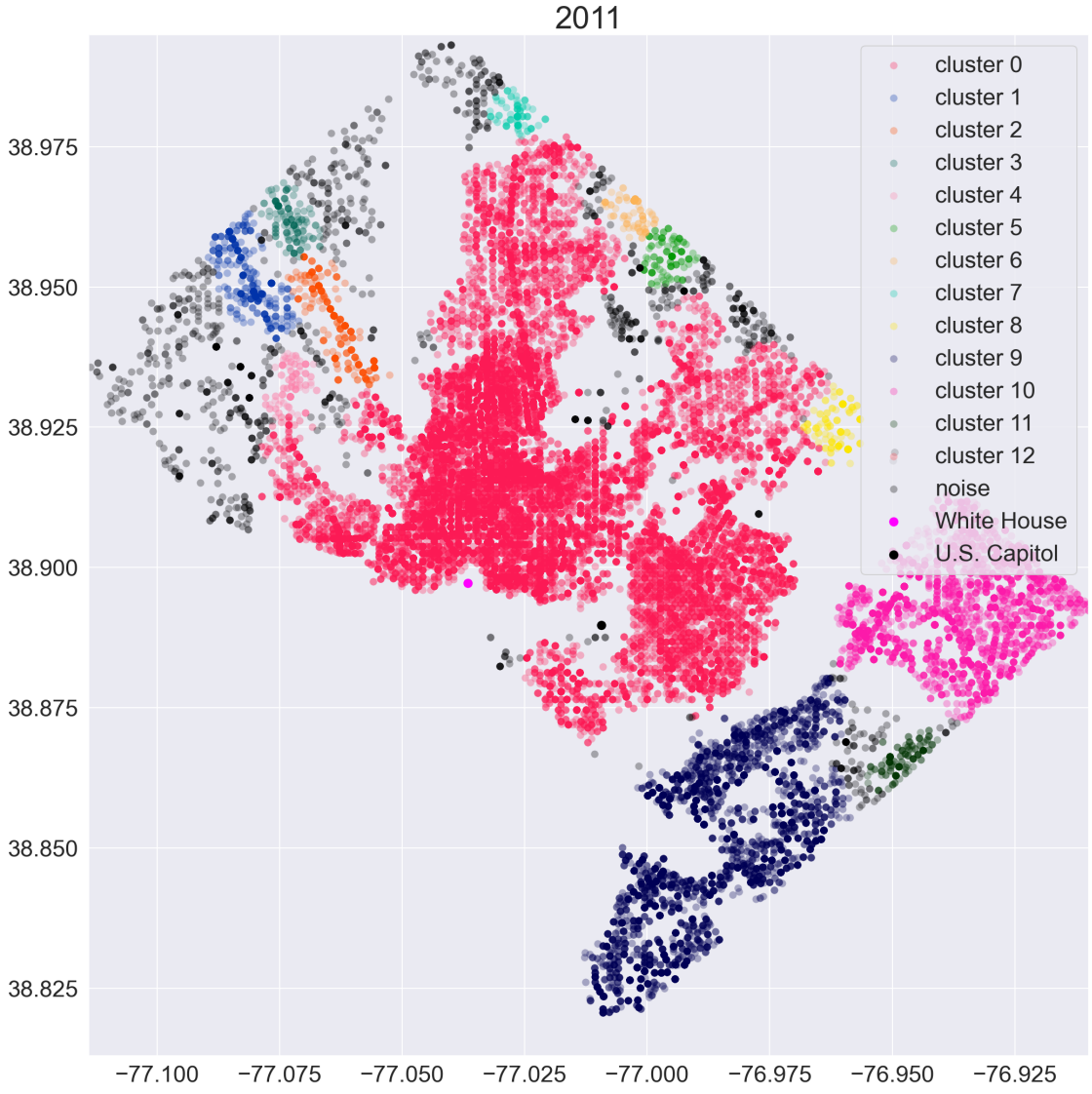
DBSCAN for crimes from 2008 to 2020.

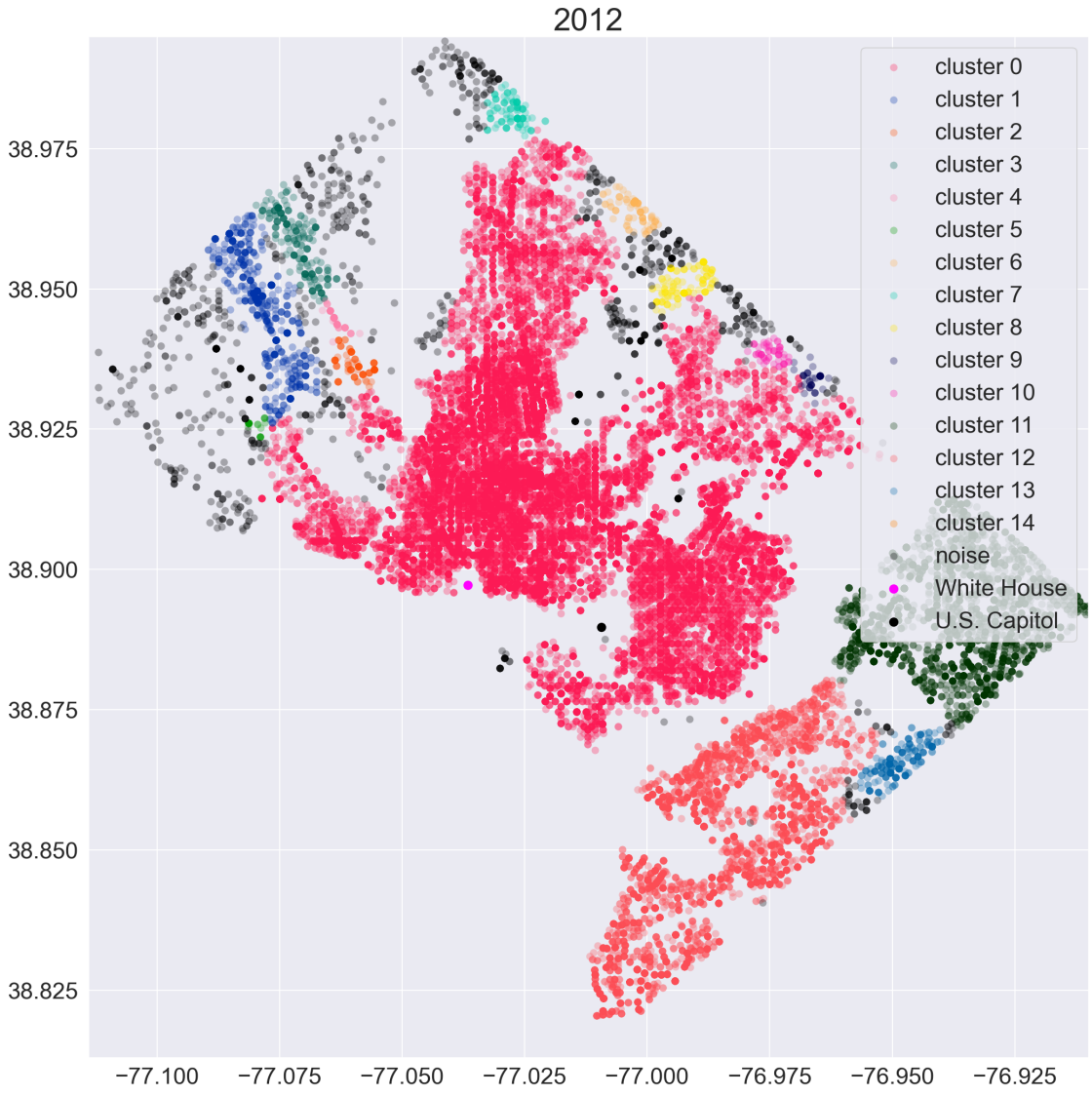
[262]:
```
for i in range(2008,2021):
    crime_DBSCAN = crime_model[crime_model.YEAR==i]
    x = crime_model[crime_model.YEAR==i][['LONGITUDE','LATITUDE']]
    x = preprocessing.scale(x)
    db = DBSCAN(eps=0.1, min_samples=50).fit(x)
    crime_DBSCAN['cluster'] = db.labels_
    temp = crime_DBSCAN['cluster'].max()+1
    see_cluster(crime_DBSCAN,temp,i,1)
# DBSCAN  clusters
```

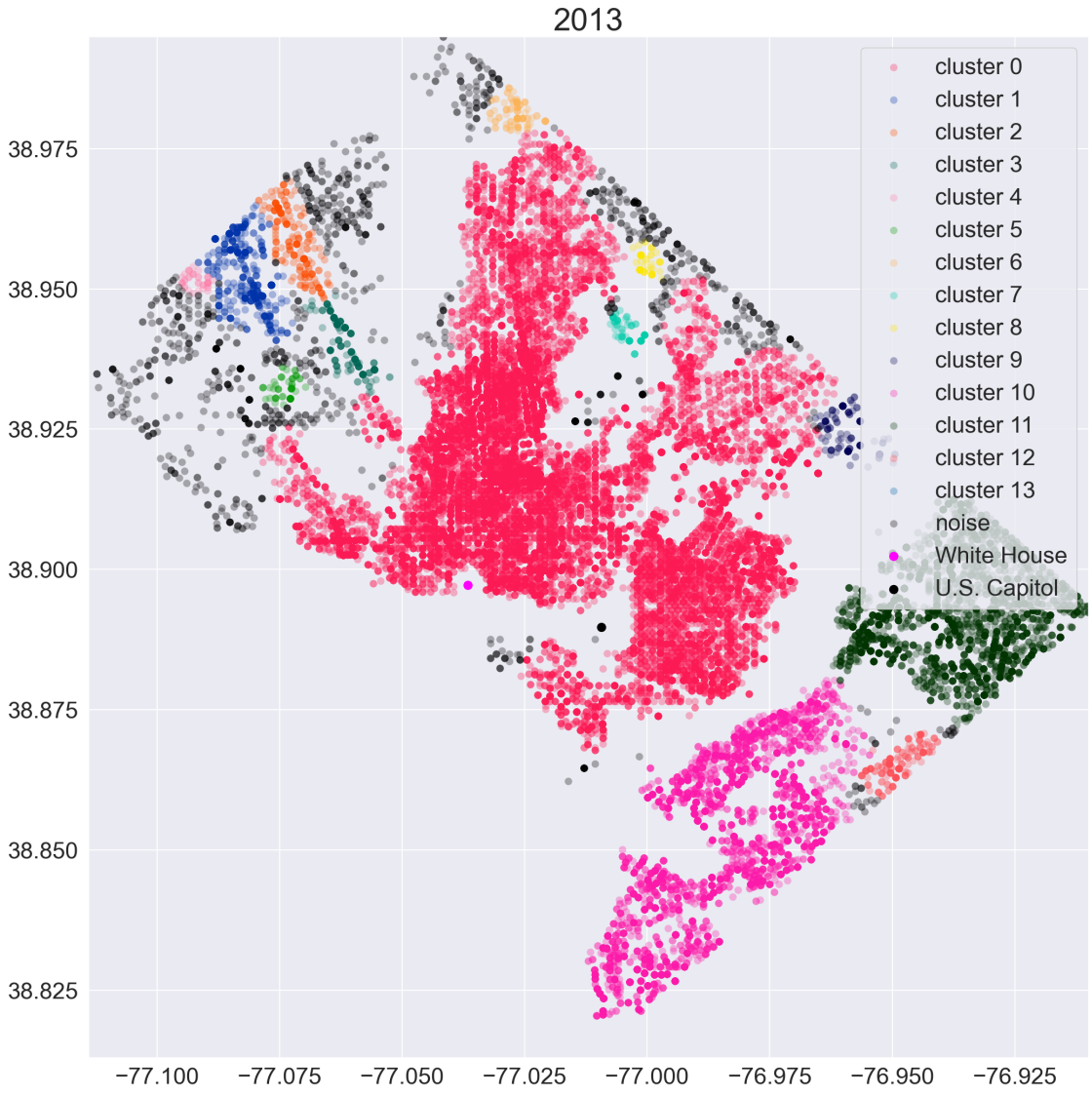

2008
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

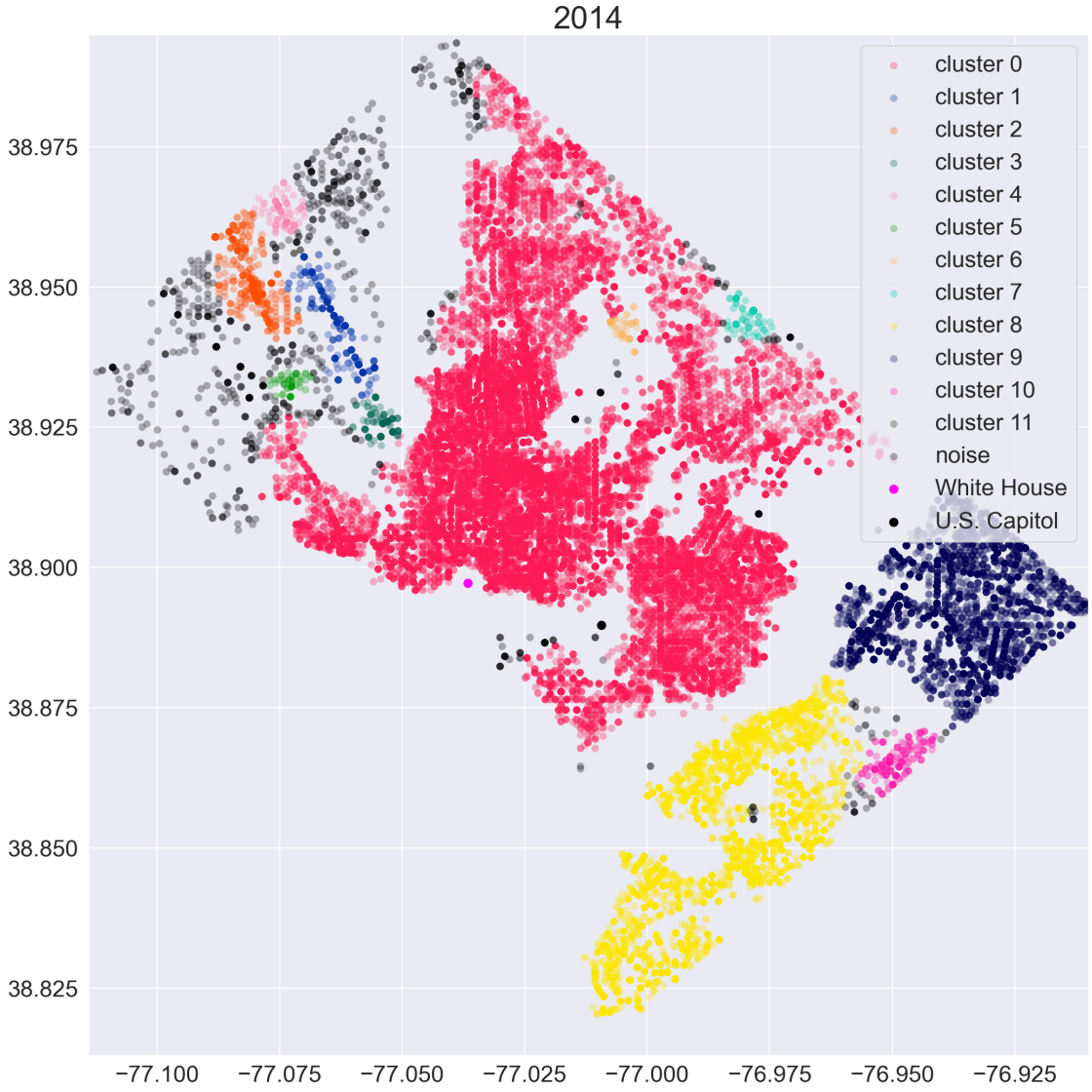

2009
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

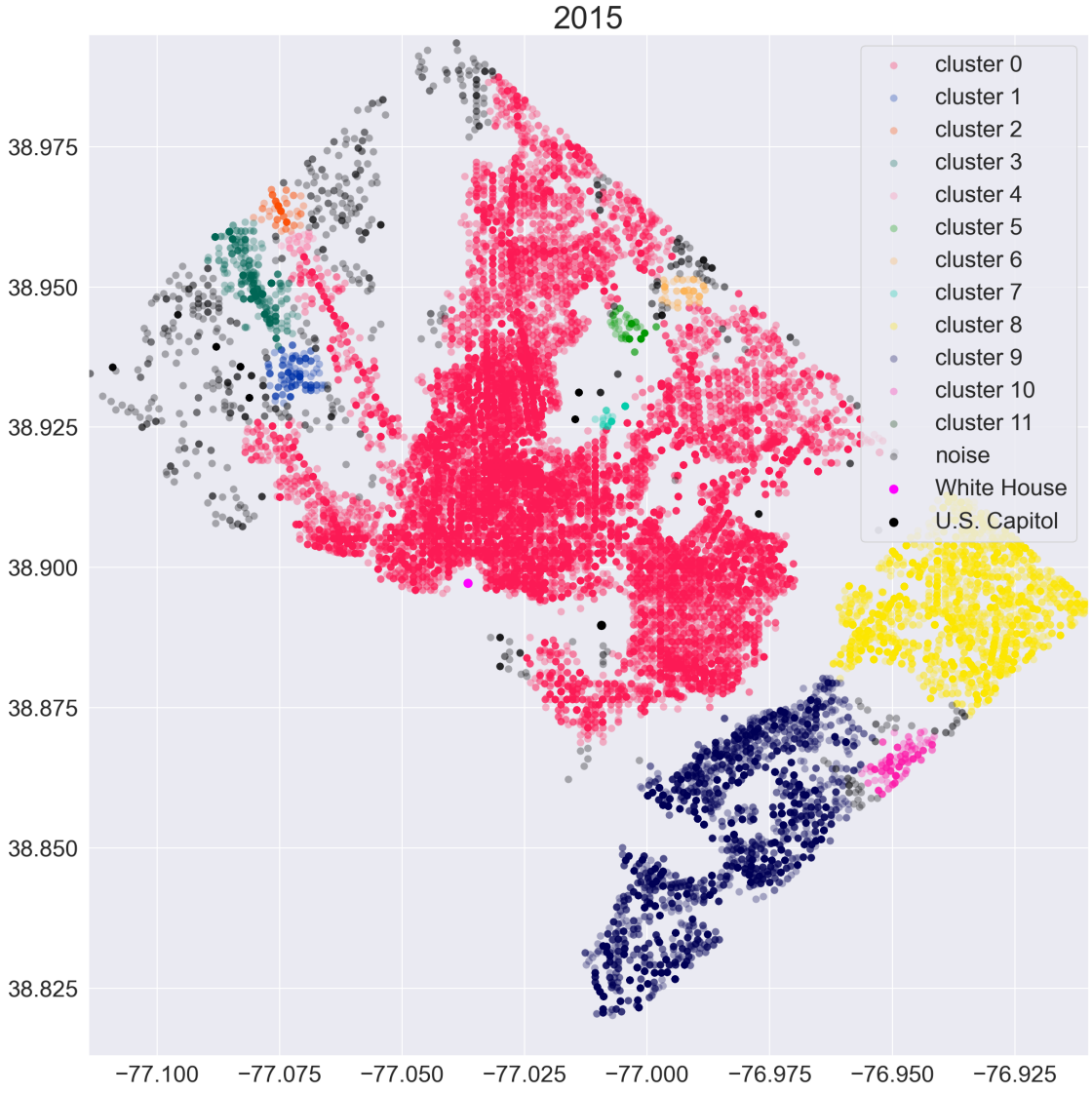

2010
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

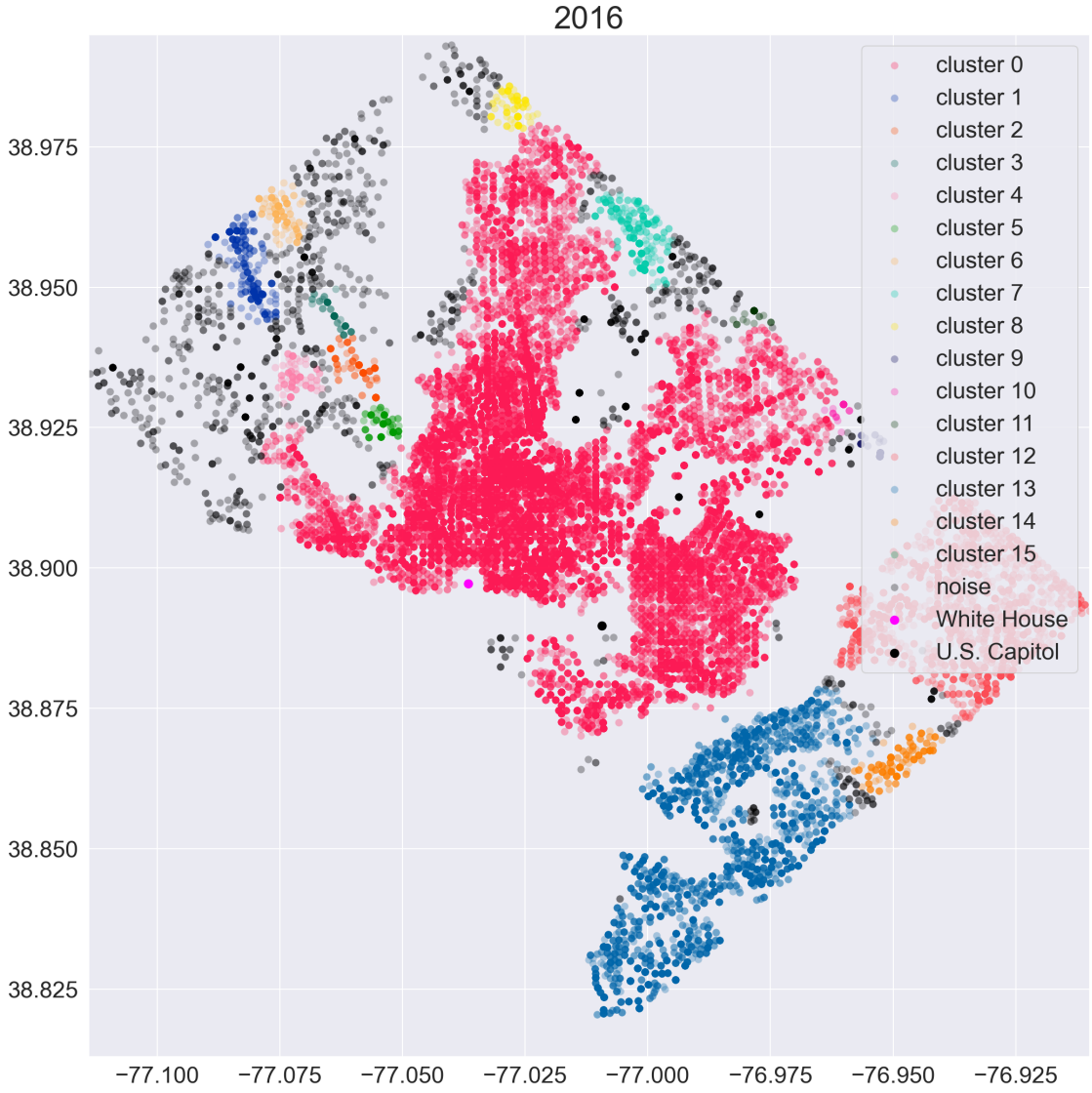

2011
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

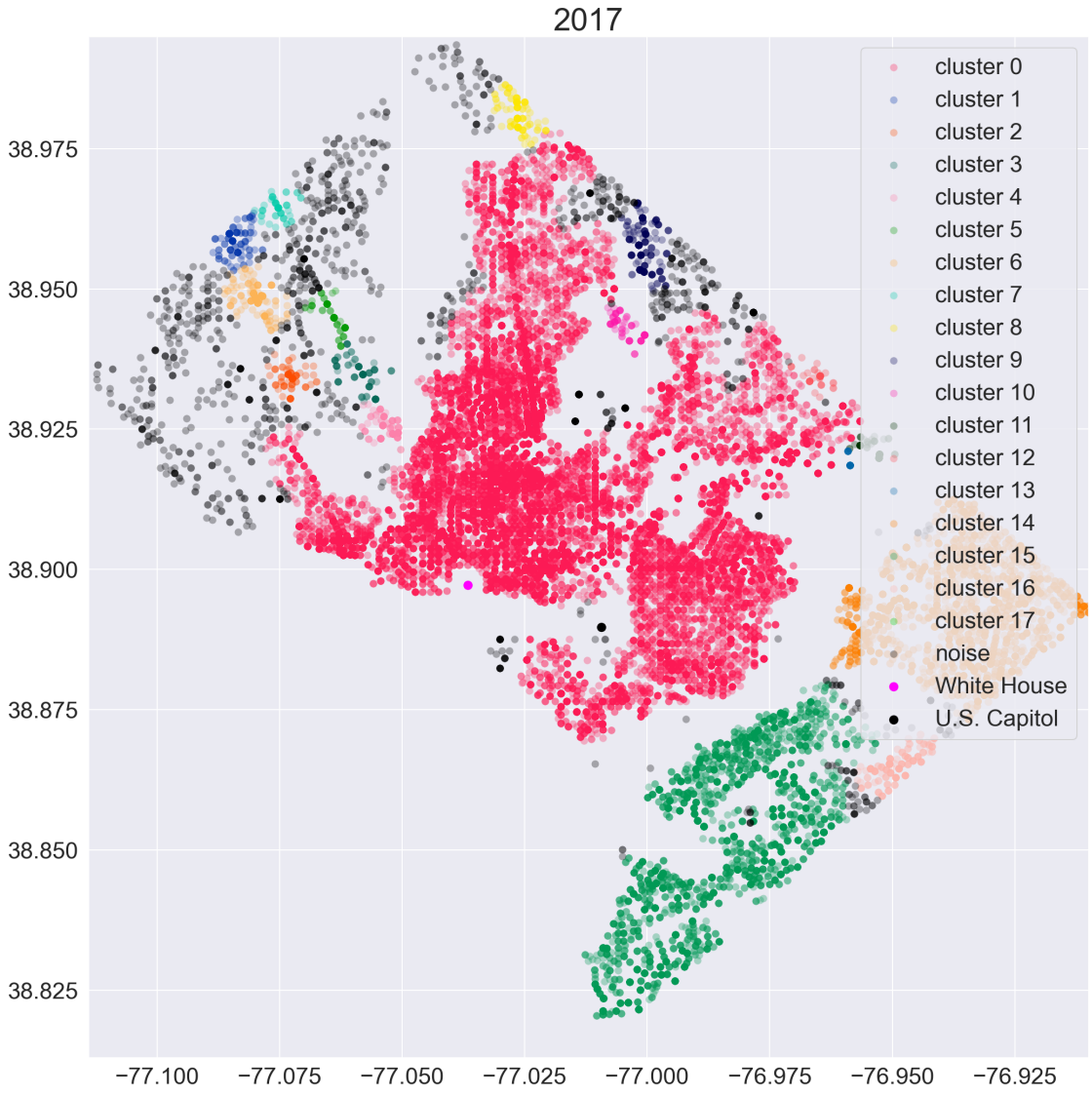

2012
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
cluster 14
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

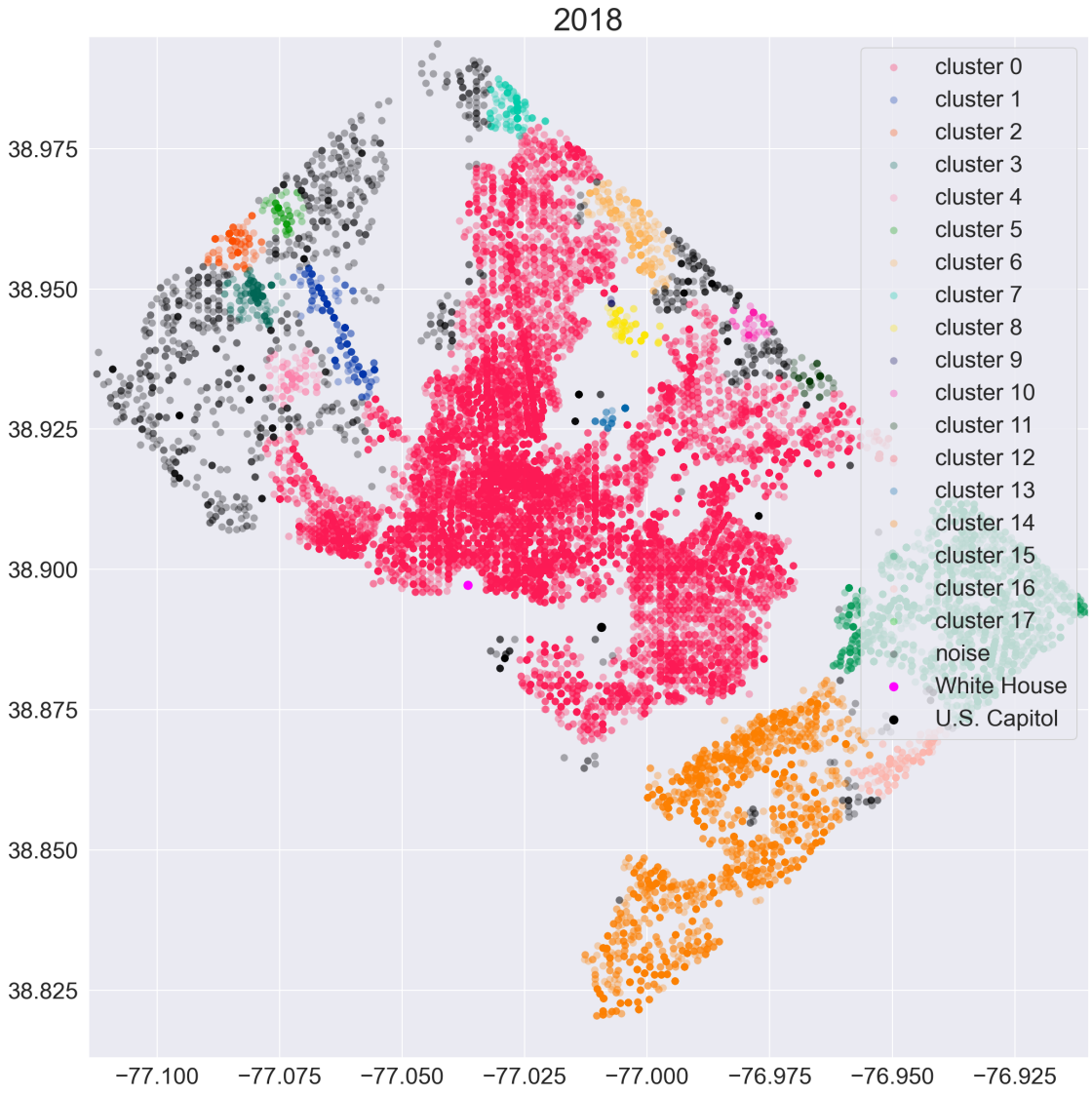

2013
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

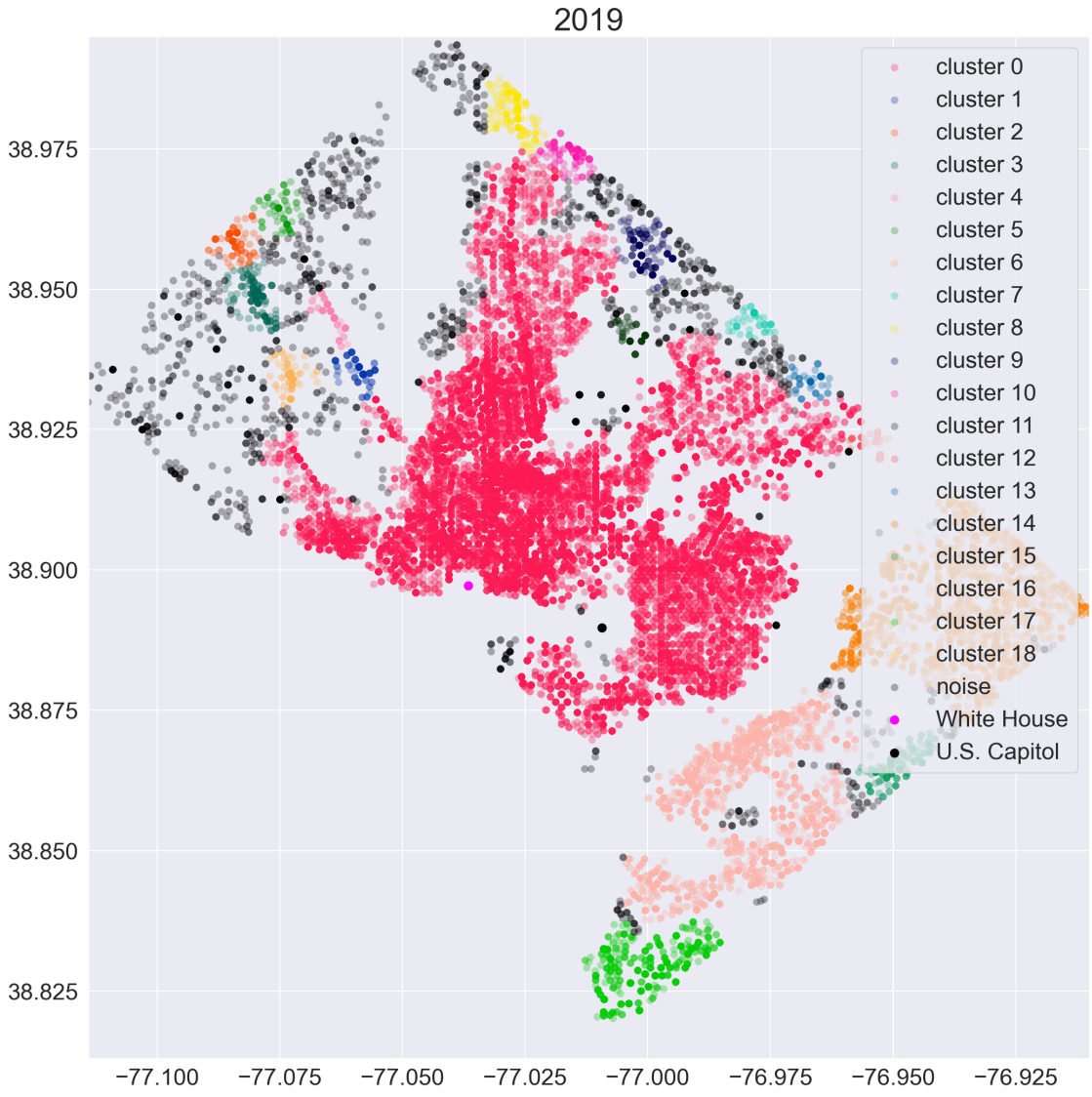

2014
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

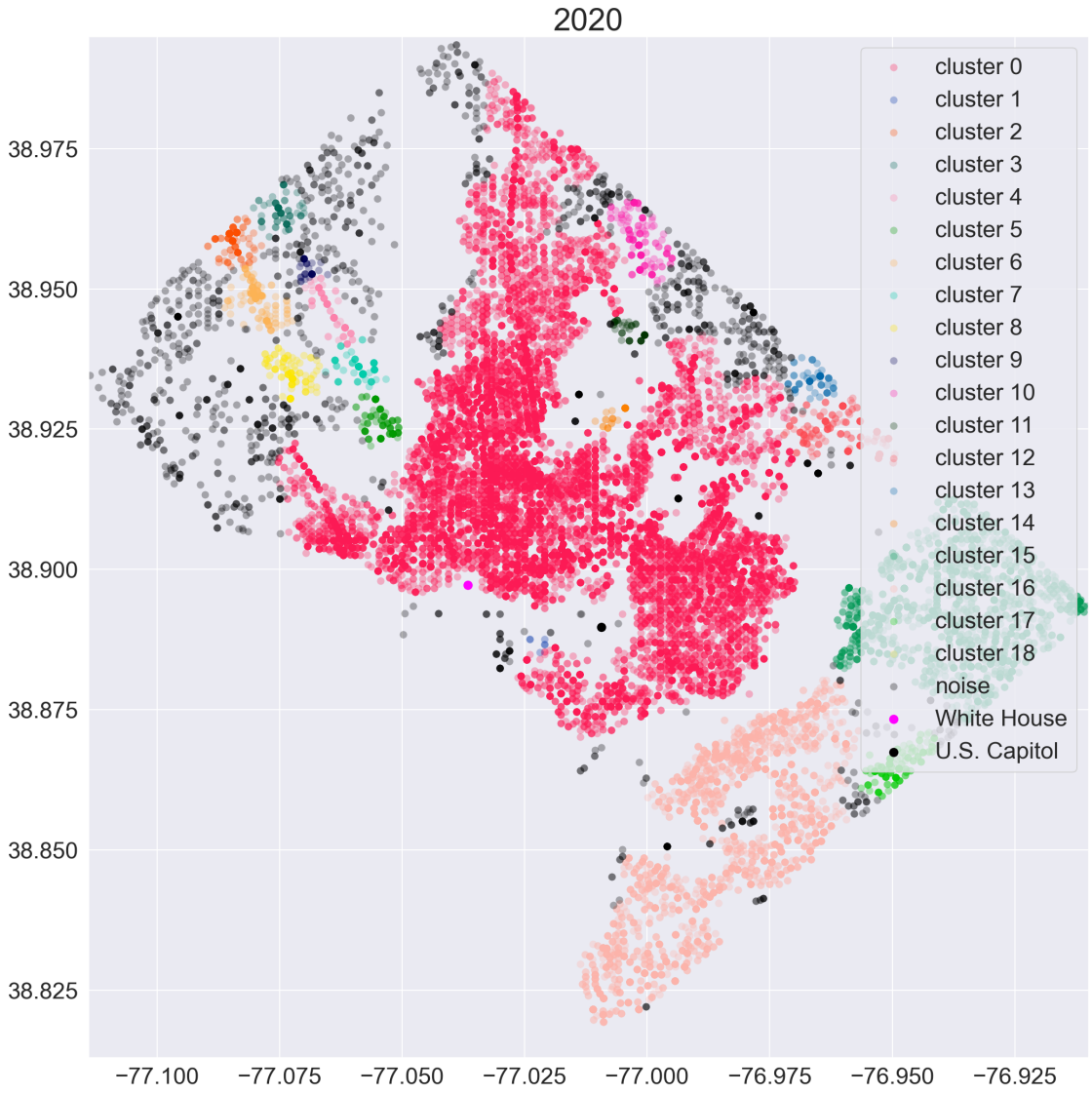

2015
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


2016
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
cluster 14
cluster 15
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

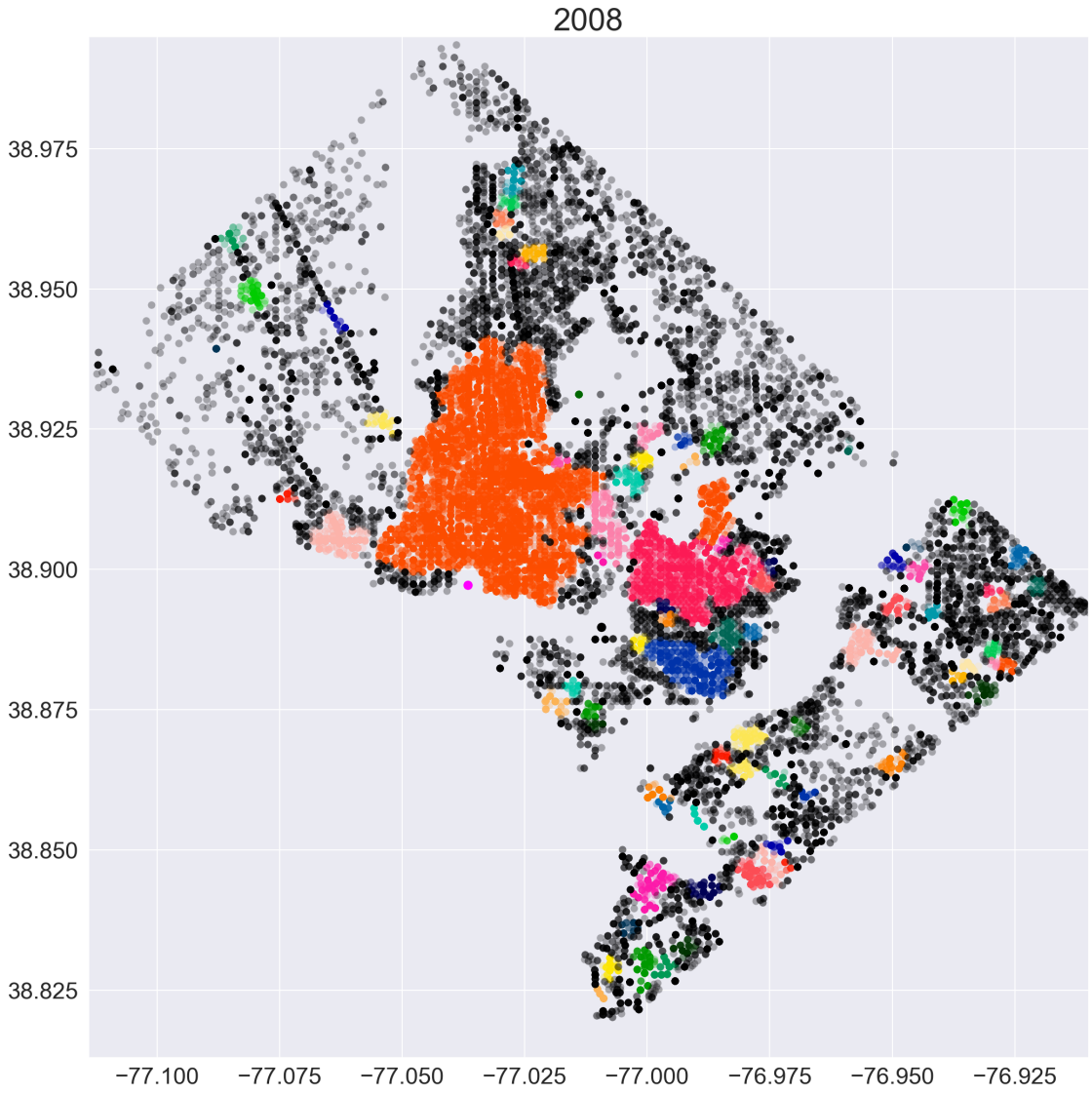

2017
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
cluster 14
cluster 15
cluster 16
cluster 17
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

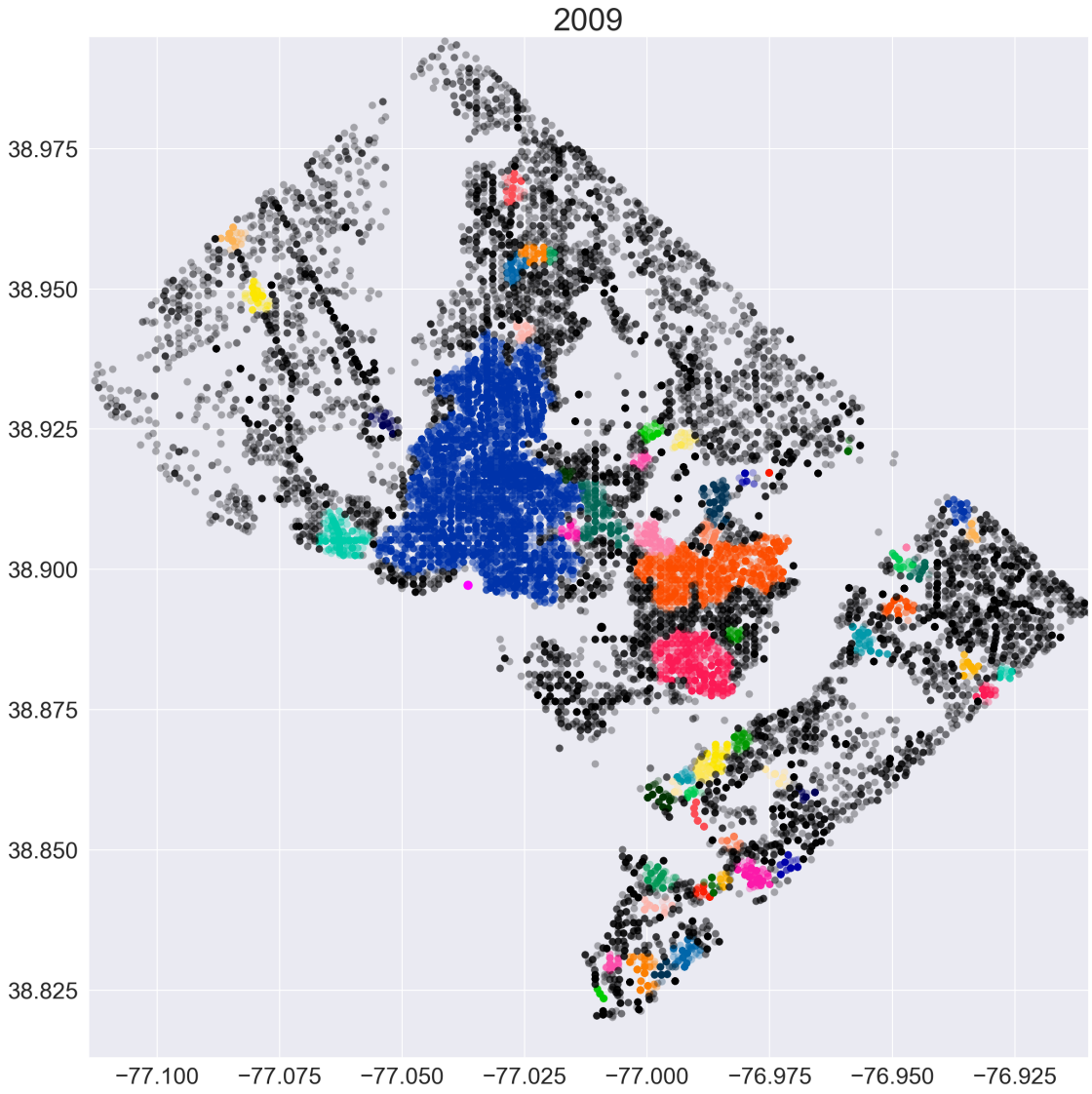

2018
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
cluster 14
cluster 15
cluster 16
cluster 17
noise
White House
U.S. Capitol
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

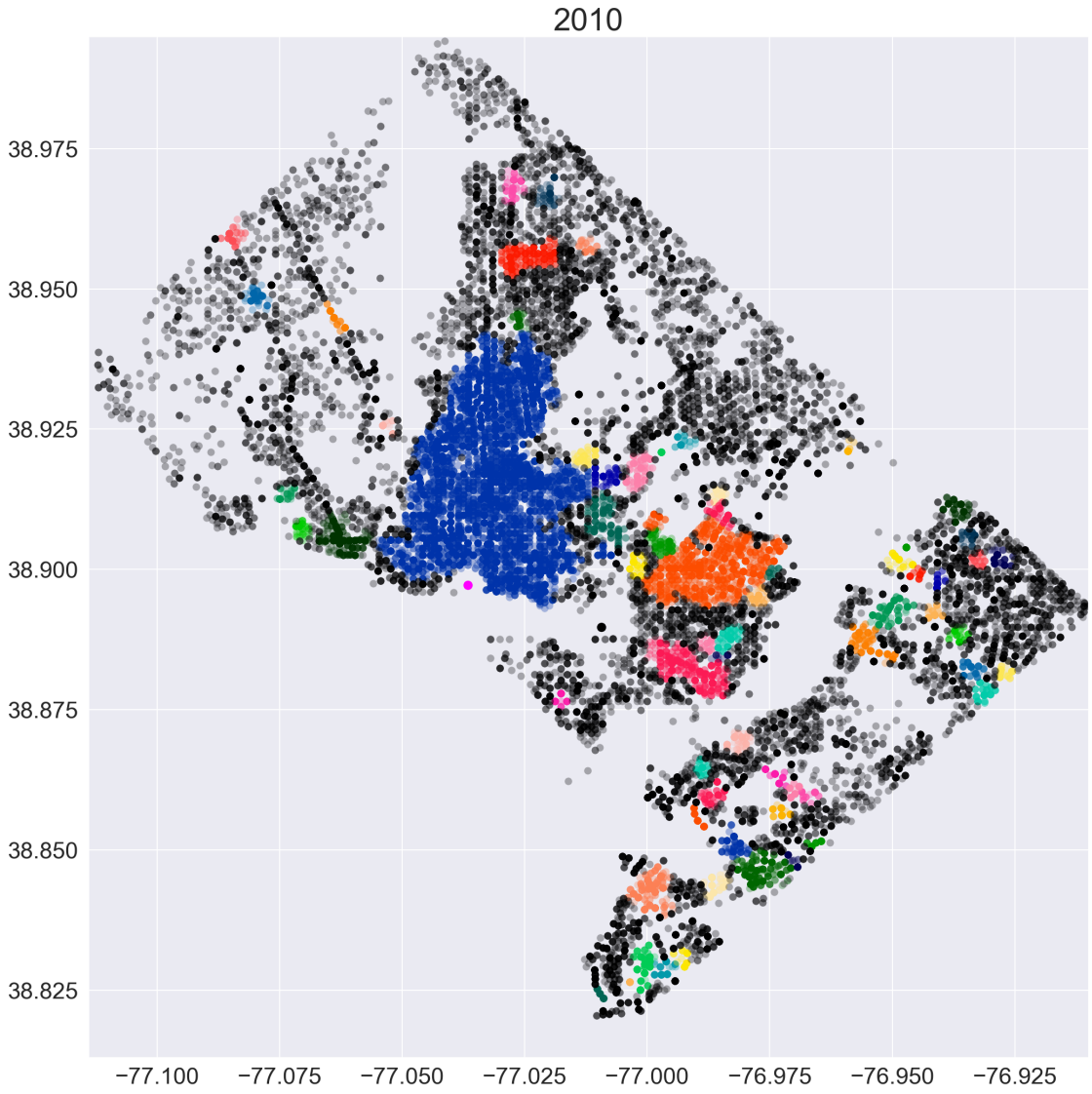

2019
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
cluster 10
cluster 11
cluster 12
cluster 13
cluster 14
cluster 15
cluster 16
cluster 17
cluster 18
noise
White House
U.S. Capitol

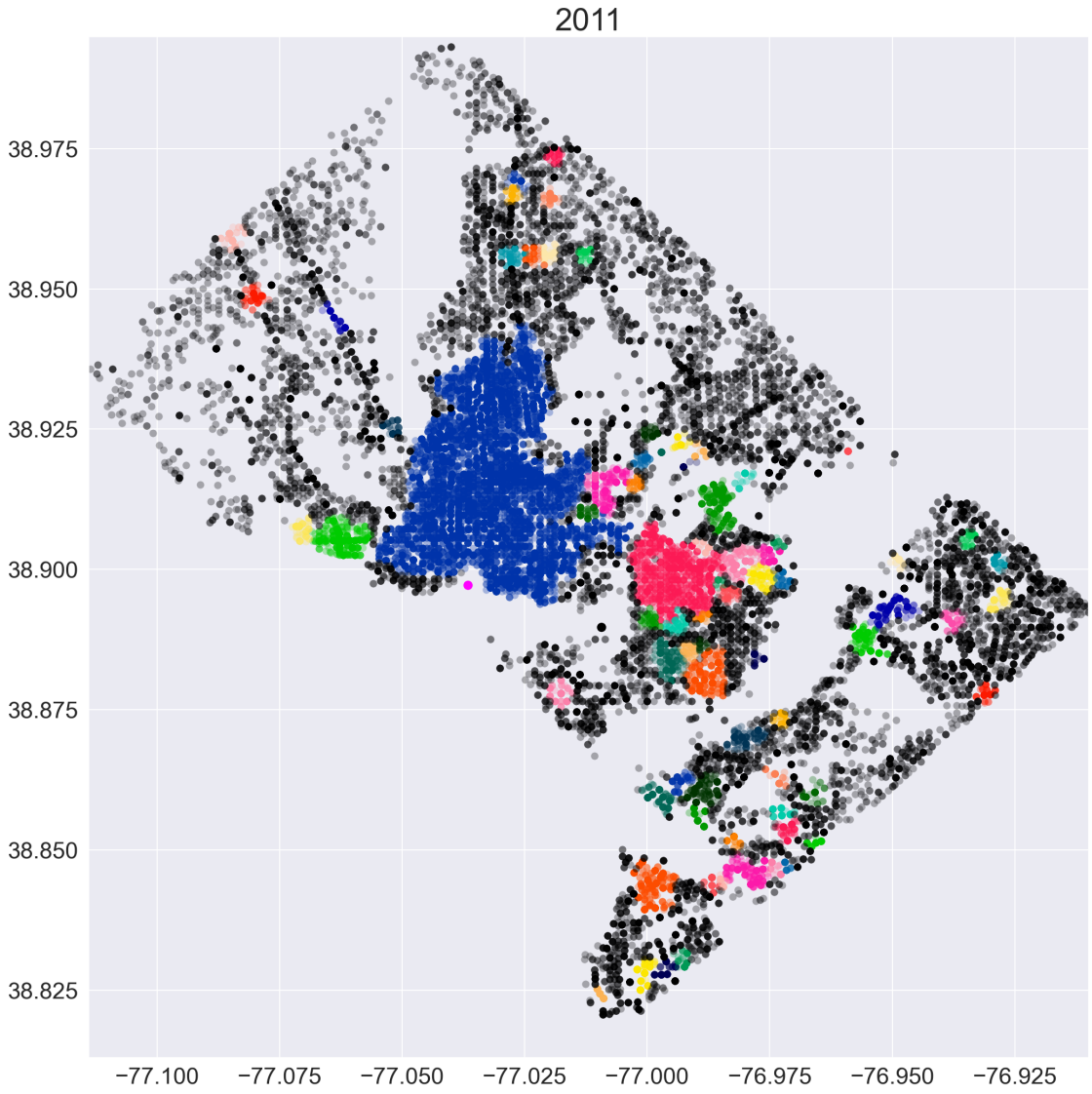


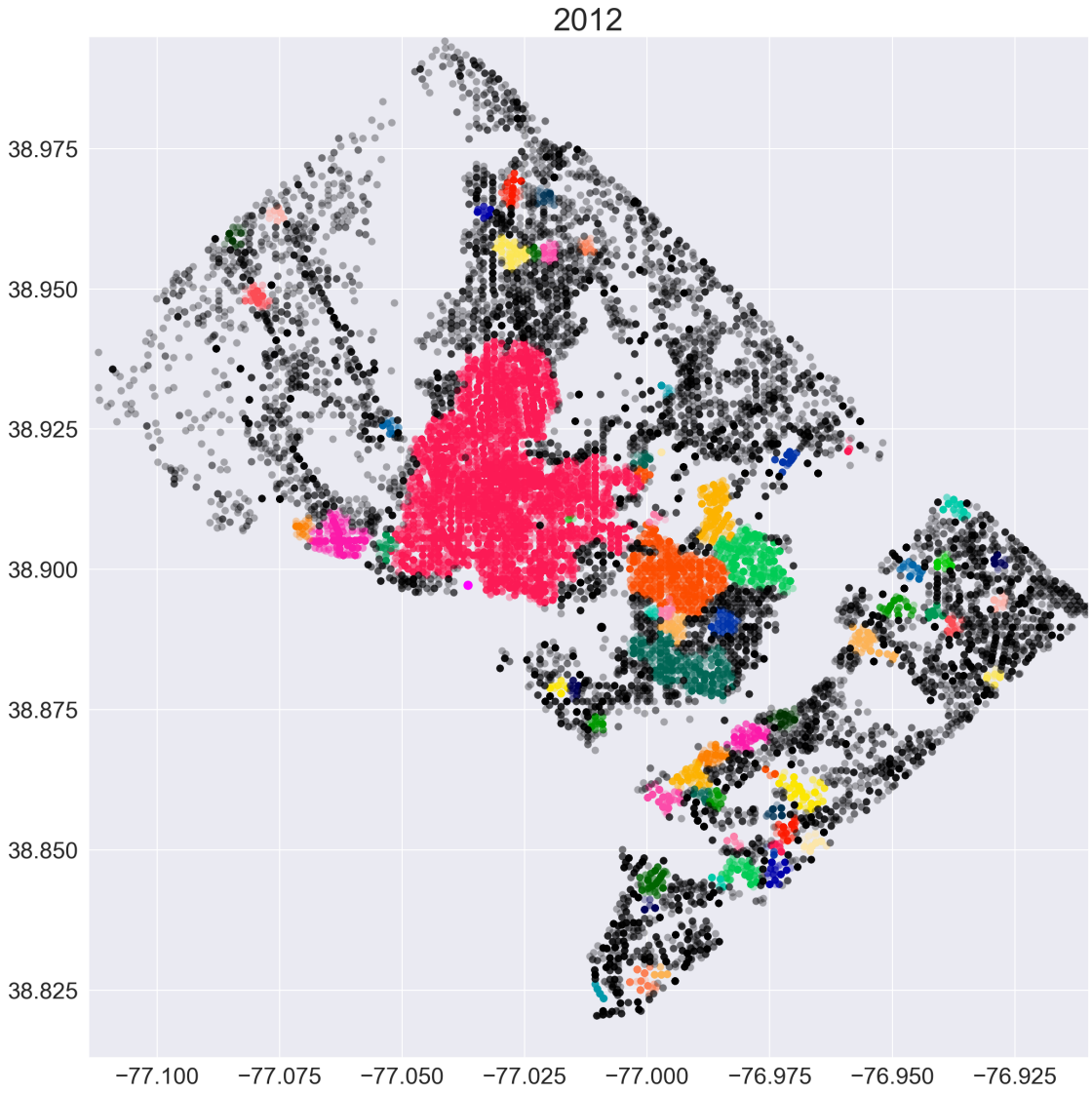
[263]:
```
for i in range(2008,2021):
    crime_DBSCAN = crime_model[crime_model.YEAR==i]
    x = crime_model[crime_model.YEAR==i][['LONGITUDE','LATITUDE']]
    x = preprocessing.scale(x)
    db = DBSCAN(eps=0.05, min_samples=50).fit(x)
    crime_DBSCAN['cluster'] = db.labels_
    temp = crime_DBSCAN['cluster'].max()+1
    see_cluster(crime_DBSCAN,temp,i,0)
# crime_ridden areas
```

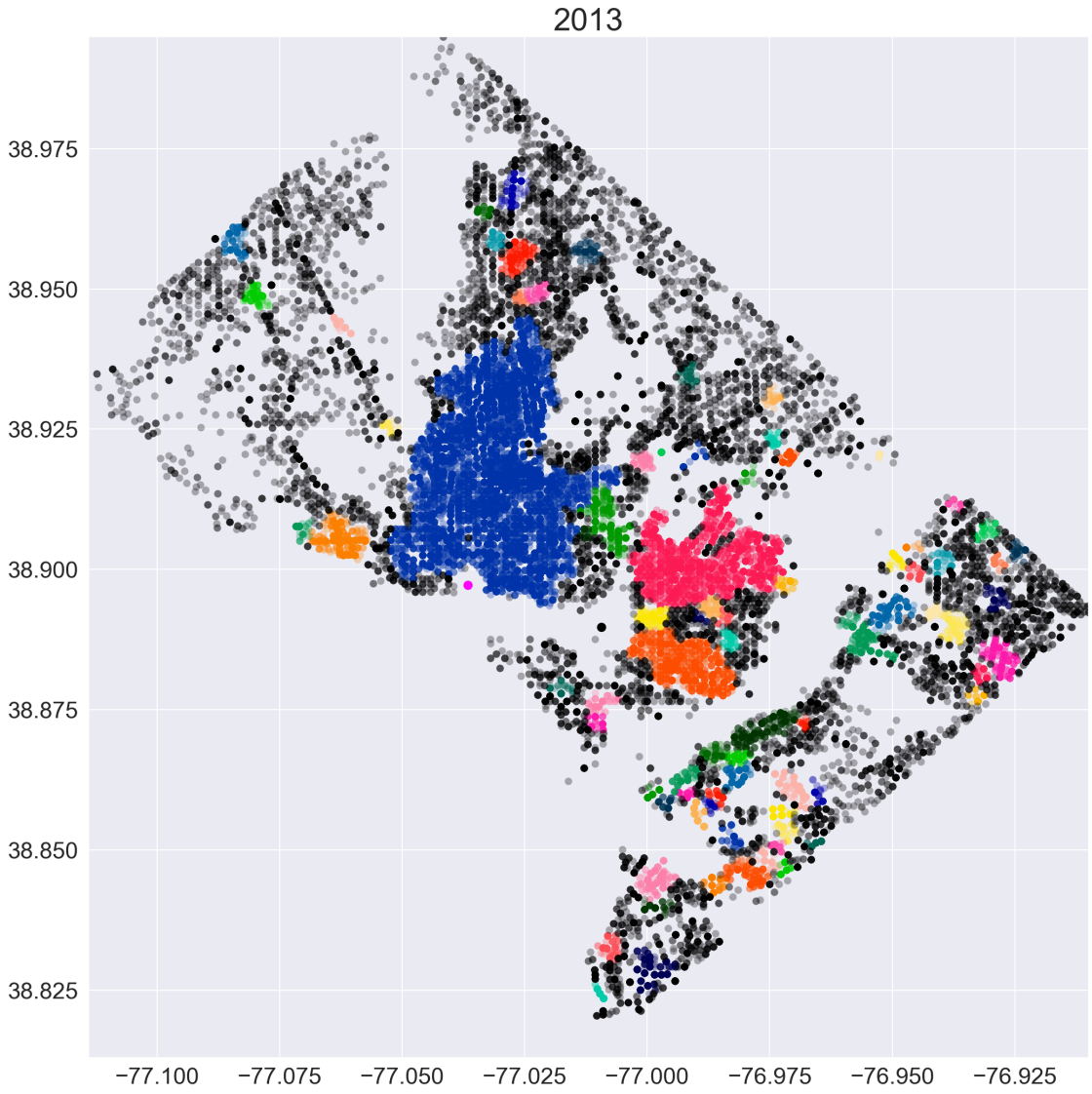

2008
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

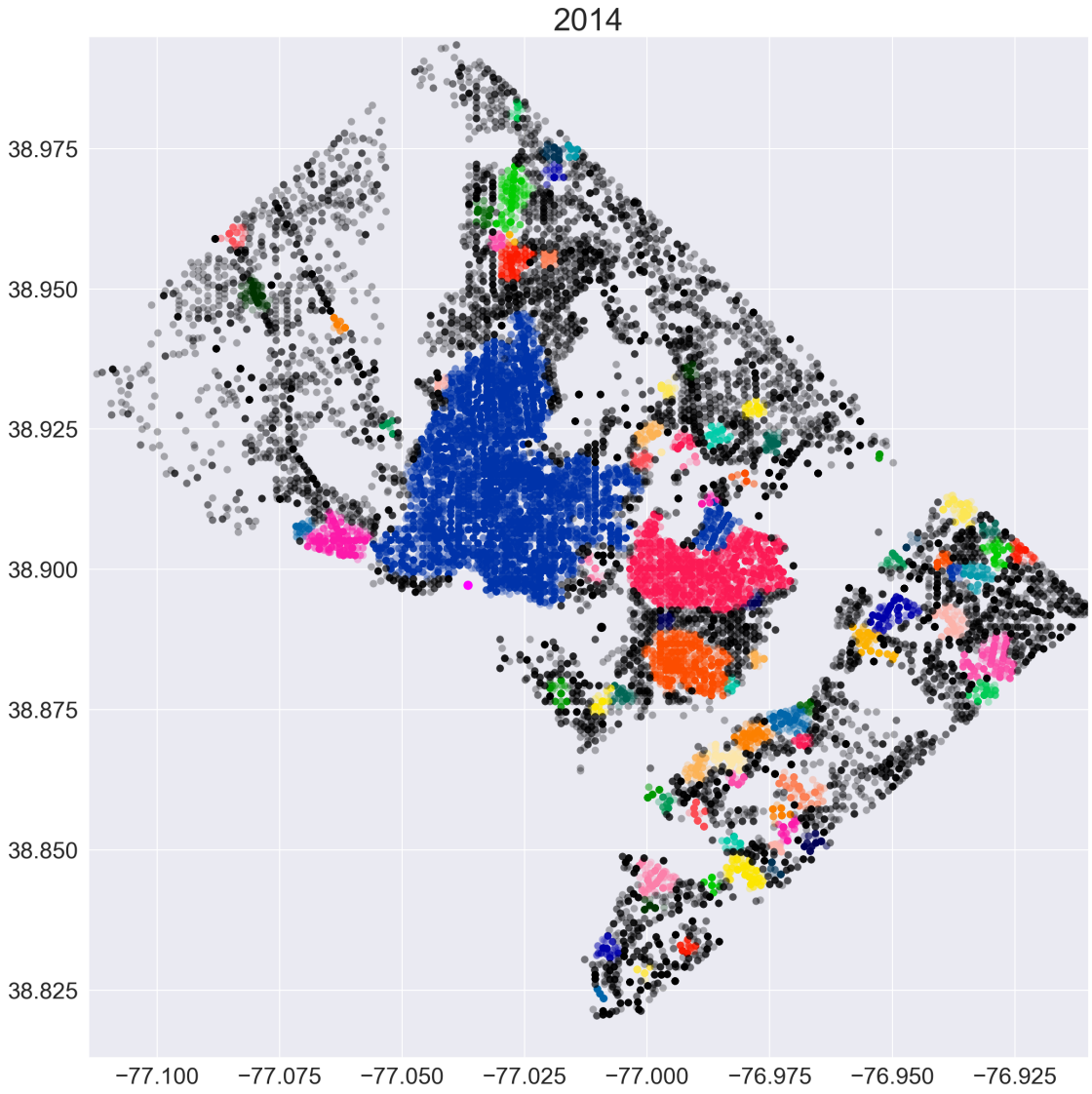

2009
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

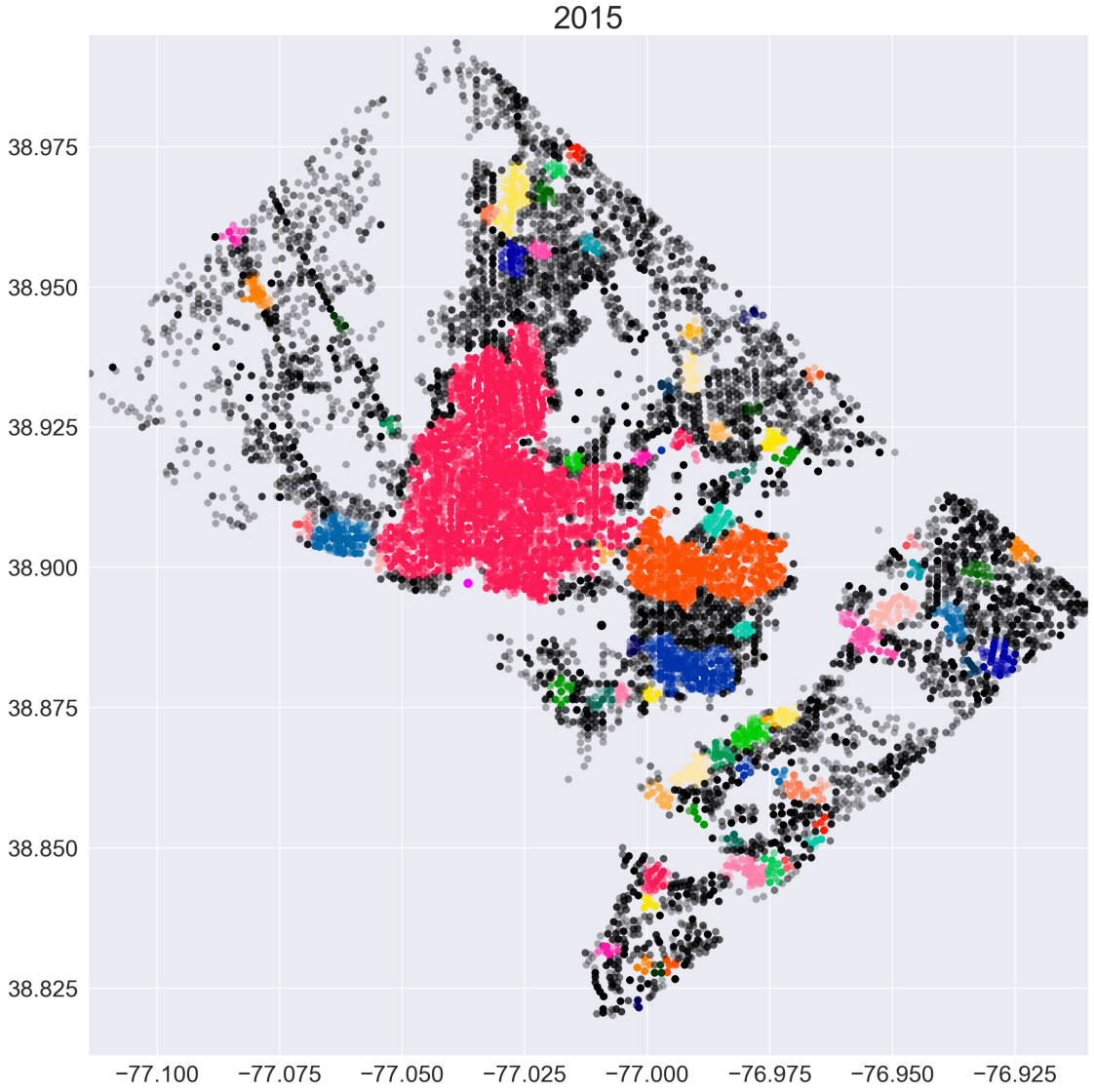

2010
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

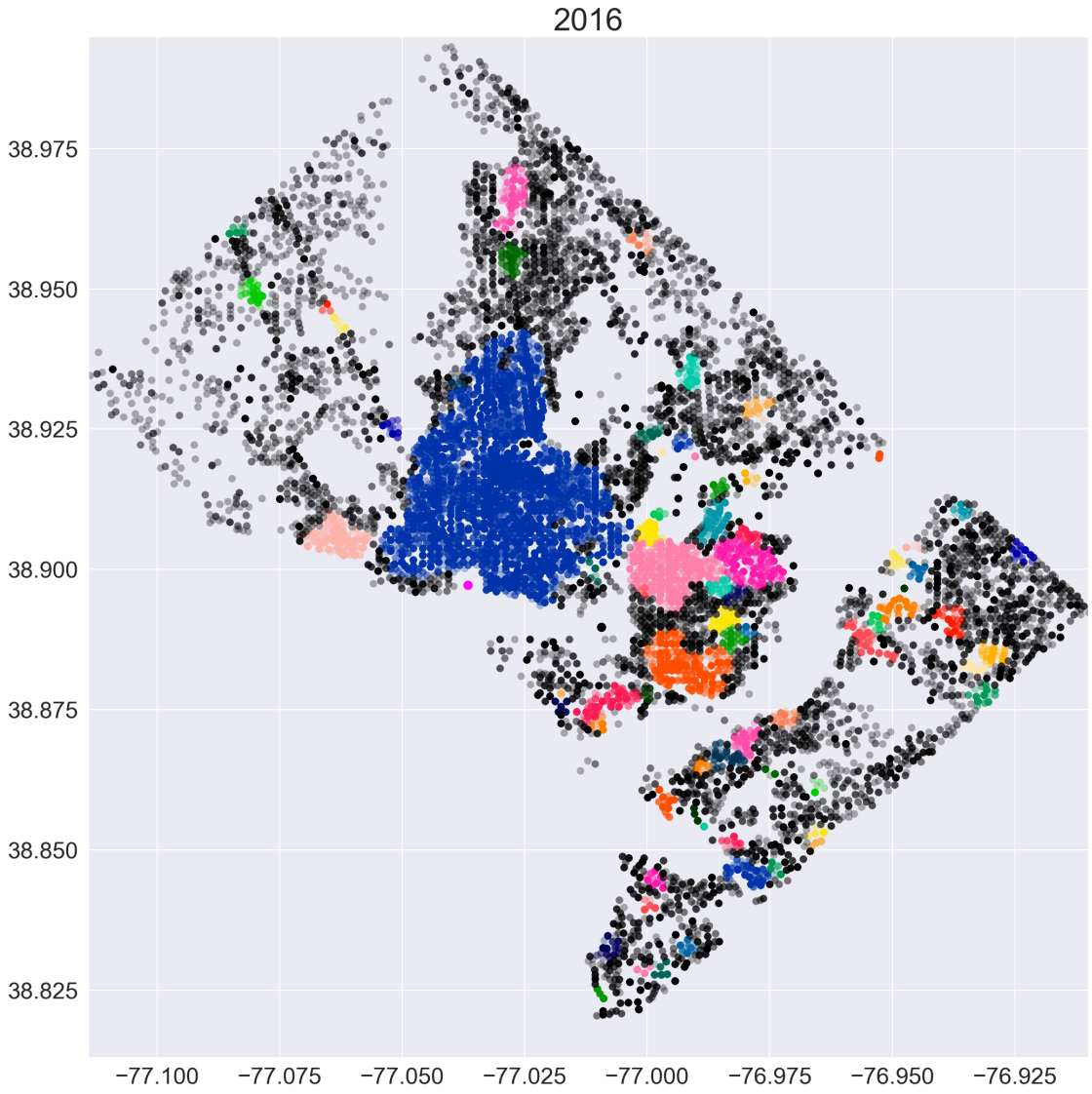

2011
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

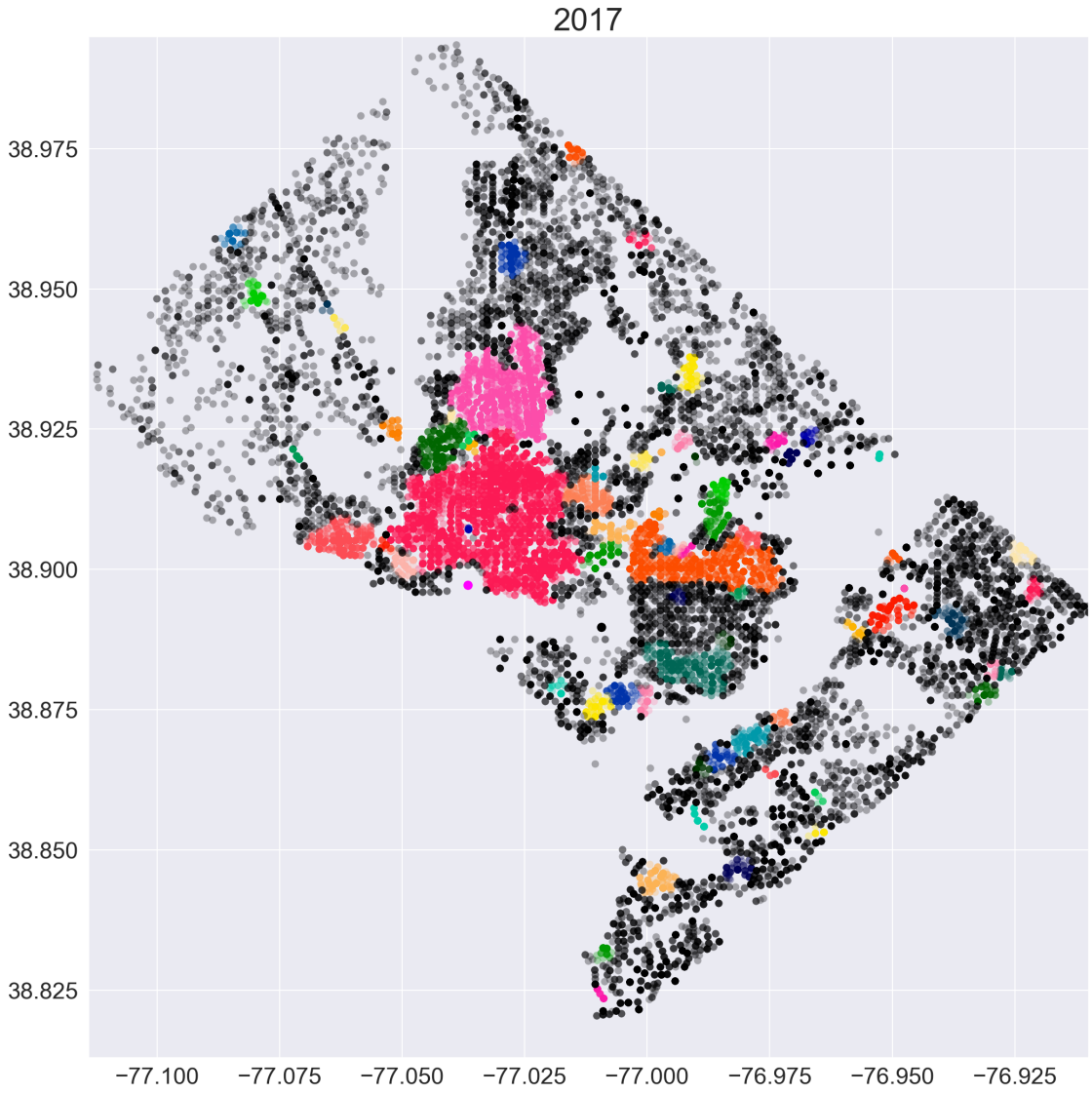

2012
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

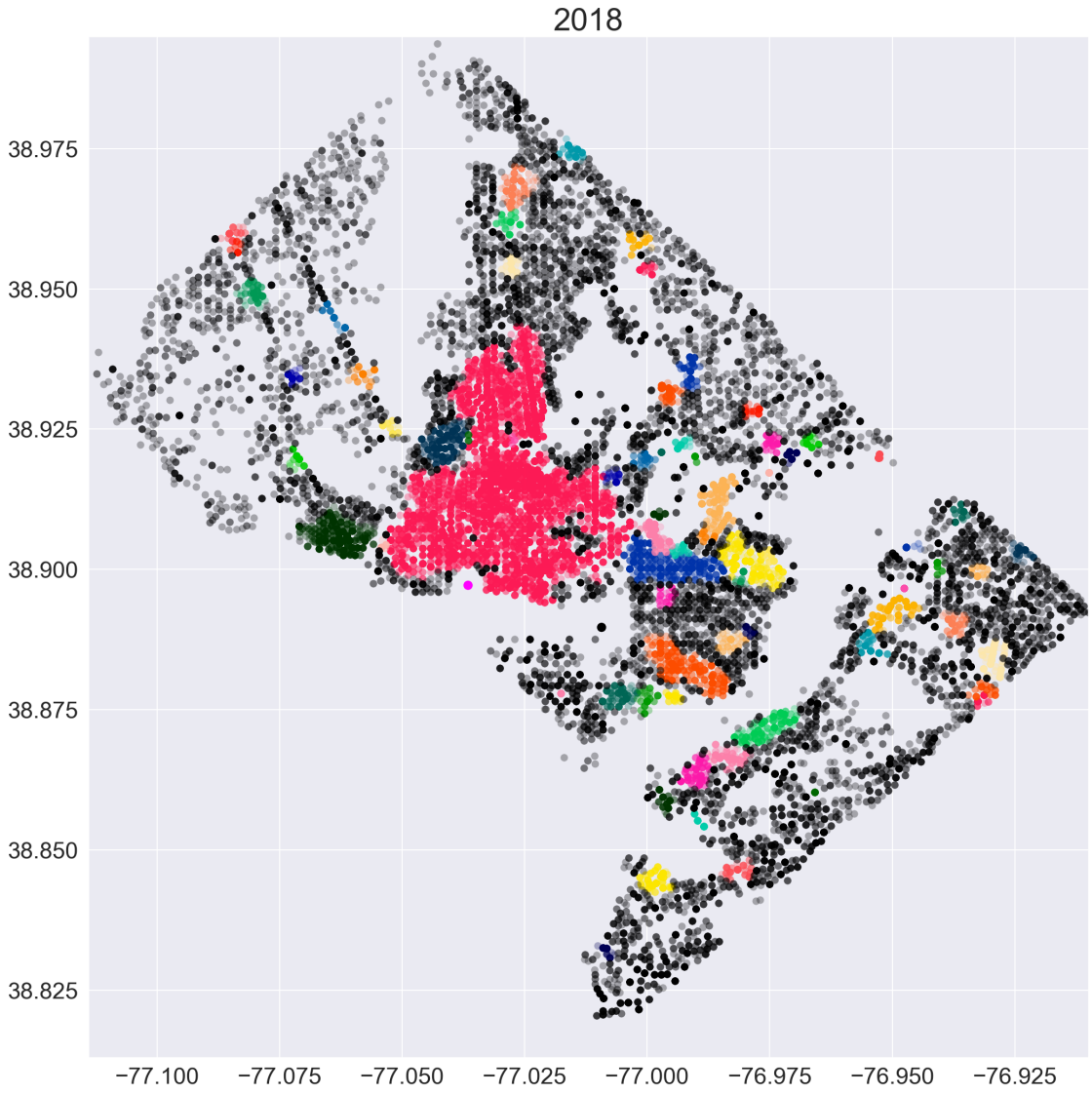

2013
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

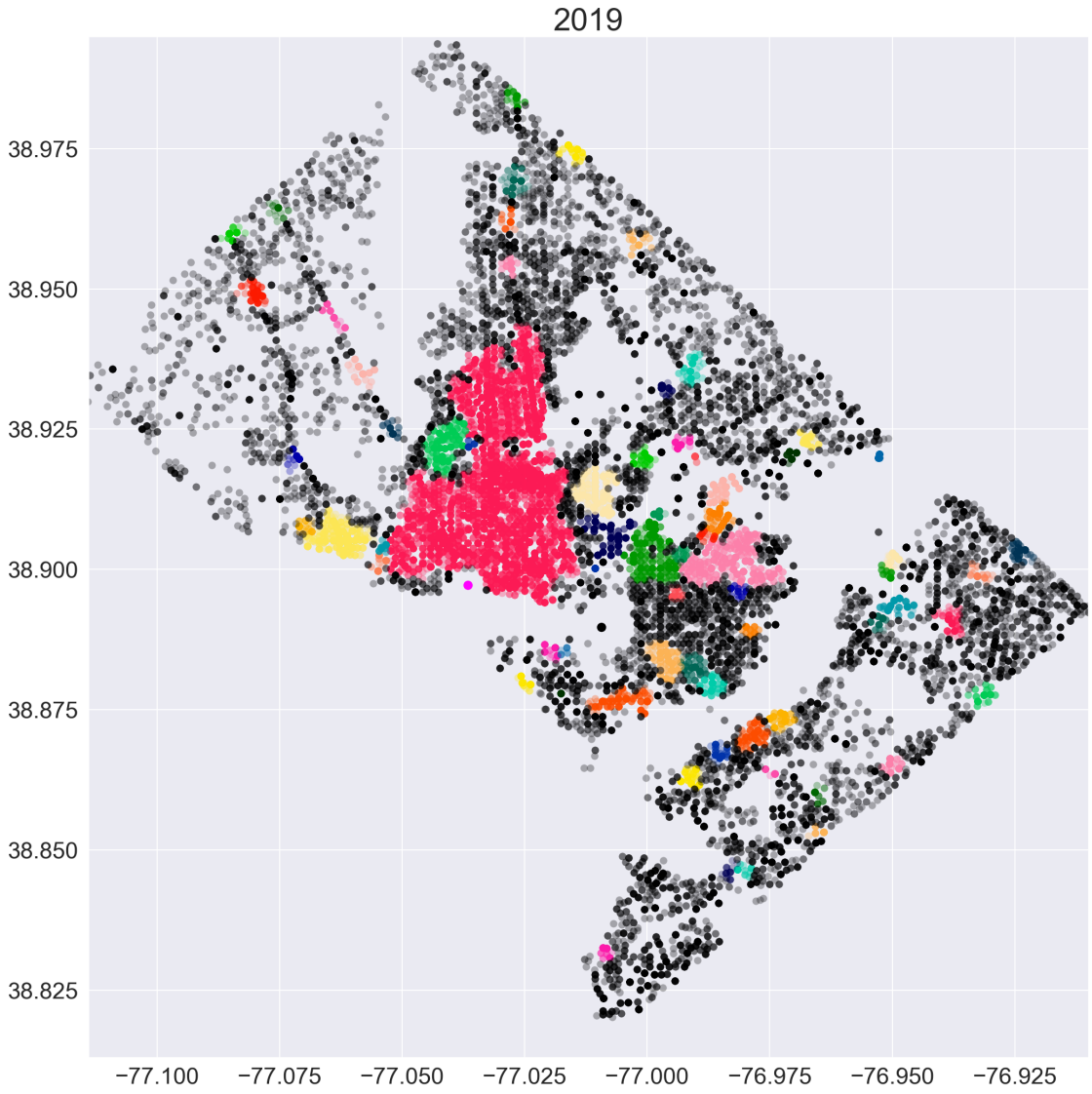

2014
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925

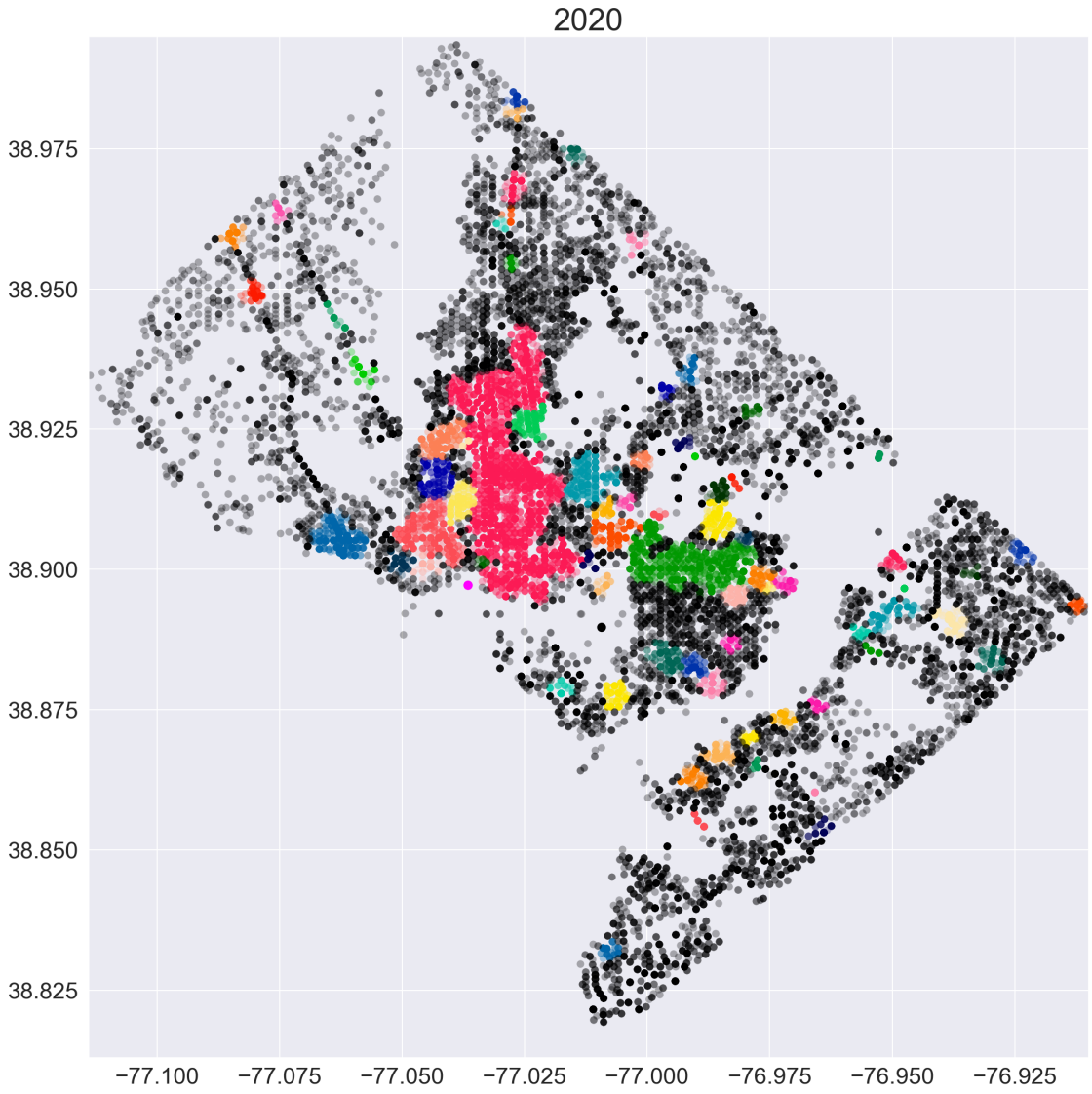

2015
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


2016
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


2017
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


2018
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925


2019
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925



*Conclusion: we can find that the geography of crime-ridden areas has not changed significantly. Besides, DBSCAN also has a good effect for clustering.

5 4 Predicting the housing price(Task 4)

@task:Is there any correlation between the crime and the housing price? We provide you an additional dataset named "DC_Properties.csv" which records the housing price and related information in the DC area. In particular, sufficient geographical information (such as latitude, longitude, zipcode, census_tract, census_block, ward, etc) and temporal information (such as ayb, eyb, saledata, etc.) are provided so that you can make a correspondence between the two datasets. Please combine the two datasets to make a new dataset with new features that you can define by yourselves. Remember to keep the most important information in the Crime data and the Properties data so that you can get everything to predict the housing price. The prediction can be made by using

regression methods. Try what you learned in the course to make the regression as good as possible.

[264]:
```
import sys
sys.path.append('D:\\python\python39\lib\site-packages')
import os
import json
import pandas as pd
import numpy as np
import warnings
import matplotlib as mpl
import missingno as msno
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
warnings.filterwarnings("ignore")
```

[265]:
```
properties = pd.read_csv("DC_Properties.csv",index_col=[0])
properties
```

[265]:

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	\
0	4	0	Warm Cool	Y	2.0	8	4	1910.0	
1	3	1	Warm Cool	Y	2.0	11	5	1898.0	
2	3	1	Hot Water Rad	Y	2.0	9	5	1910.0	
3	3	1	Hot Water Rad	Y	2.0	8	5	1900.0	
4	2	1	Warm Cool	Y	1.0	11	3	1913.0	
…	…	…	… ..	…	…	…	…		
158952	1	0	Forced Air	Y	NaN	3	1	1938.0	
158953	1	0	Forced Air	Y	NaN	4	2	1938.0	
158954	2	0	Forced Air	Y	NaN	4	2	1920.0	
158955	1	0	Warm Cool	Y	NaN	2	0	1965.0	
158956	1	0	Warm Cool	Y	NaN	2	0	1965.0	

	YR_RMDL	EYB	…	LONGITUDE	ASSESSMENT_NBHD	\
0	1988.0	1972	…	-77.040832	Old City 2	
1	2007.0	1972	…	-77.040764	Old City 2	
2	2009.0	1984	…	-77.040678	Old City 2	
3	2003.0	1984	…	-77.040629	Old City 2	
4	2012.0	1985	…	-77.039361	Old City 2	
…	…	…	…	…	…	
158952	2006.0	1938	…	-77.019420	Old City 2	
158953	2006.0	1938	…	-77.019420	Old City 2	
158954	2007.0	1920	…	-77.019420	Old City 2	
158955	NaN	1965	…	-77.018230	Southwest Waterfront	
158956	NaN	1965	…	-77.018230	Southwest Waterfront	

```
        ASSESSMENT_SUBNBHD CENSUS_TRACT  CENSUS_BLOCK    WARD  SQUARE  \
0         040 D Old City 2       4201.0   004201 2006  Ward 2     152
1         040 D Old City 2       4201.0   004201 2006  Ward 2     152
2         040 D Old City 2       4201.0   004201 2006  Ward 2     152
3         040 D Old City 2       4201.0   004201 2006  Ward 2     152
4         040 D Old City 2       4201.0   004201 2006  Ward 2     152
…                      …            …             …     …     …
158952    040 B Old City 2       4801.0           NaN  Ward 6     477
158953    040 B Old City 2       4801.0           NaN  Ward 6     477
158954    040 B Old City 2       4801.0           NaN  Ward 6     477
158955                 NaN      11000.0           NaN  Ward 6     504
158956                 NaN      11000.0           NaN  Ward 6     504

                X          Y QUADRANT
0      -77.040429  38.914881       NW
1      -77.040429  38.914881       NW
2      -77.040429  38.914881       NW
3      -77.040429  38.914881       NW
4      -77.040429  38.914881       NW
…             …          …      …
158952 -77.019422  38.911848       NW
158953 -77.019422  38.911848       NW
158954 -77.019422  38.911848       NW
158955 -77.018232  38.872961       SW
158956 -77.018232  38.872961       SW

[158957 rows x 48 columns]
```

show general information of properties

[266]:
```
properties.info()
```

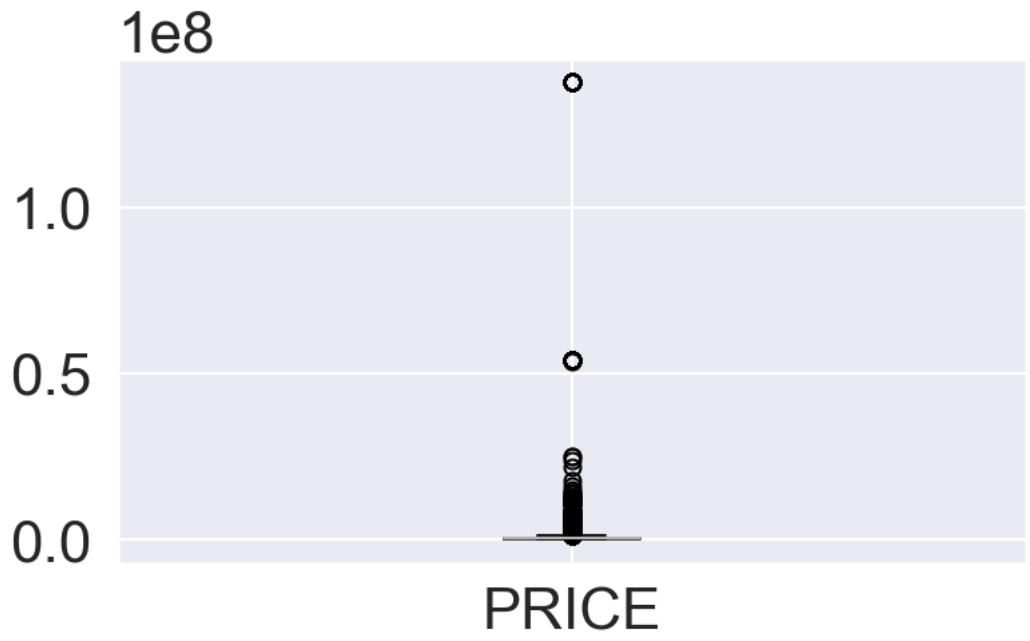
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158957 entries, 0 to 158956
Data columns (total 48 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   BATHRM              158957 non-null  int64
 1   HF_BATHRM           158957 non-null  int64
 2   HEAT                158957 non-null  object
 3   AC                  158957 non-null  object
 4   NUM_UNITS           106696 non-null  float64
 5   ROOMS               158957 non-null  int64
 6   BEDRM               158957 non-null  int64
 7   AYB                 158686 non-null  float64
 8   YR_RMDL             80928 non-null   float64
 9   EYB                 158957 non-null  int64
 10  STORIES             106652 non-null  float64
```

```
 11  SALEDATE            132187 non-null  object
 12  PRICE               98216 non-null   float64
 13  QUALIFIED           158957 non-null  object
 14  SALE_NUM            158957 non-null  int64
 15  GBA                 106696 non-null  float64
 16  BLDG_NUM            158957 non-null  int64
 17  STYLE               106696 non-null  object
 18  STRUCT              106696 non-null  object
 19  GRADE               106696 non-null  object
 20  CNDTN               106696 non-null  object
 21  EXTWALL             106696 non-null  object
 22  ROOF                106696 non-null  object
 23  INTWALL             106696 non-null  object
 24  KITCHENS            106695 non-null  float64
 25  FIREPLACES          158957 non-null  int64
 26  USECODE             158957 non-null  int64
 27  LANDAREA            158957 non-null  int64
 28  GIS_LAST_MOD_DTTM   158957 non-null  object
 29  SOURCE              158957 non-null  object
 30  CMPLX_NUM           52261 non-null   float64
 31  LIVING_GBA          52261 non-null   float64
 32  FULLADDRESS         106040 non-null  object
 33  CITY                106051 non-null  object
 34  STATE               106051 non-null  object
 35  ZIPCODE             158956 non-null  float64
 36  NATIONALGRID        106051 non-null  object
 37  LATITUDE            158956 non-null  float64
 38  LONGITUDE           158956 non-null  float64
 39  ASSESSMENT_NBHD     158956 non-null  object
 40  ASSESSMENT_SUBNBHD  126406 non-null  object
 41  CENSUS_TRACT        158956 non-null  float64
 42  CENSUS_BLOCK        106051 non-null  object
 43  WARD                158956 non-null  object
 44  SQUARE              158957 non-null  object
 45  X                   158720 non-null  float64
 46  Y                   158720 non-null  float64
 47  QUADRANT            158720 non-null  object
dtypes: float64(15), int64(10), object(23)
memory usage: 59.4+ MB
```

show outliers of some features, and deal with them
consider values over or under the values of 1.5 times difference between quatiles plus or minus quatiles as outliers

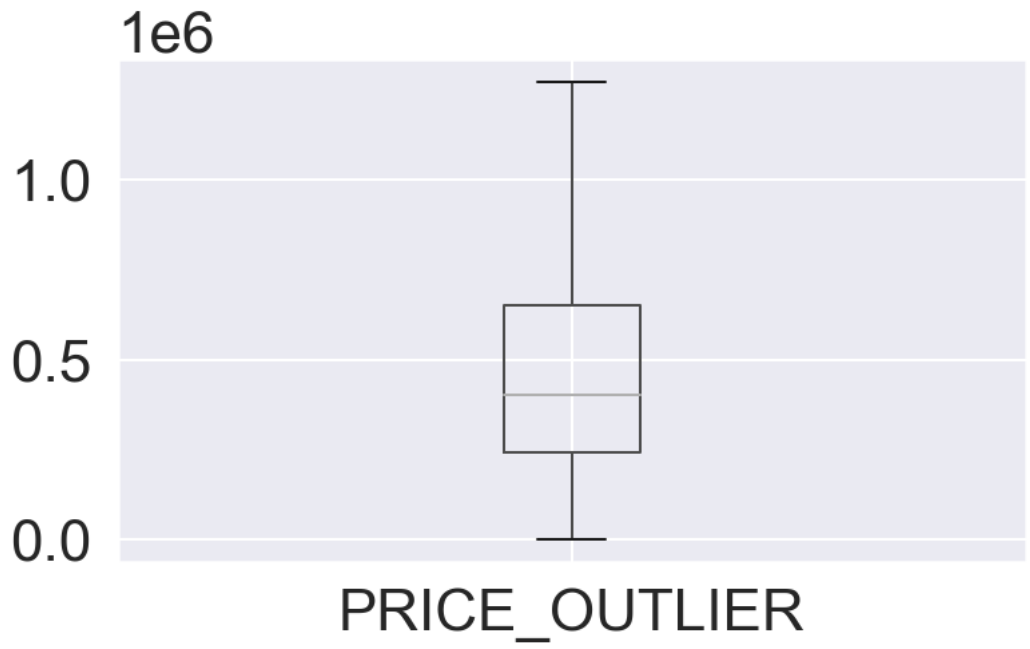
[267]:
```
properties[["PRICE"]].boxplot()
plt.show()
```

[267]:
```
<AxesSubplot:>
```



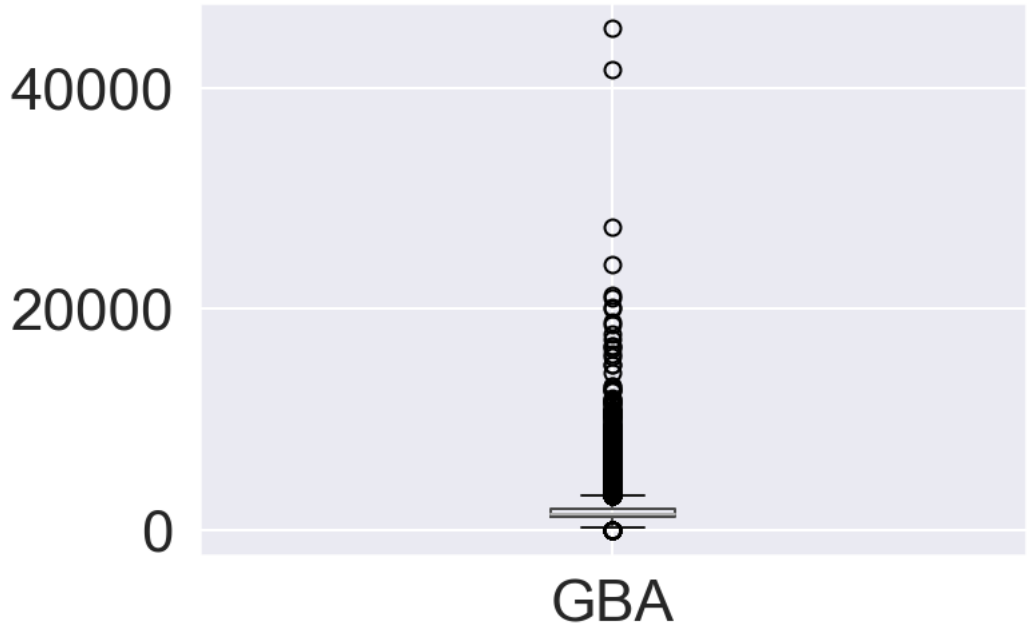
[268]:
```
up = properties["PRICE"].quantile(0.75)
lo = properties["PRICE"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["PRICE_OUTLIER"]=properties["PRICE"].apply(price_outlier)
properties[["PRICE_OUTLIER"]].boxplot()
plt.show()
```

[268]: <AxesSubplot:>



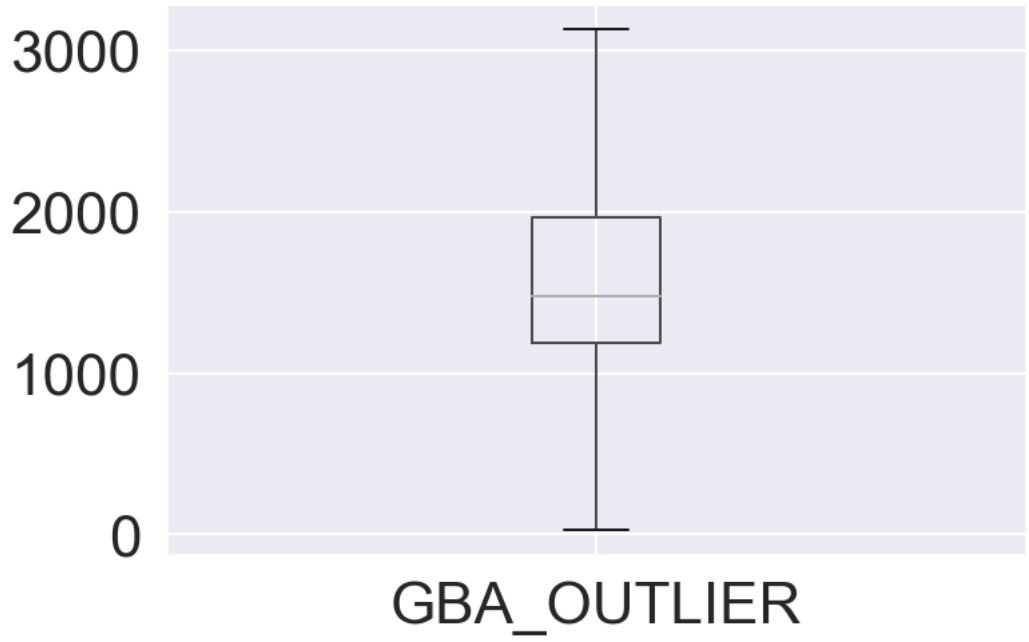
[269]:
```
properties[["GBA"]].boxplot()
plt.show()
```

[269]: <AxesSubplot:>



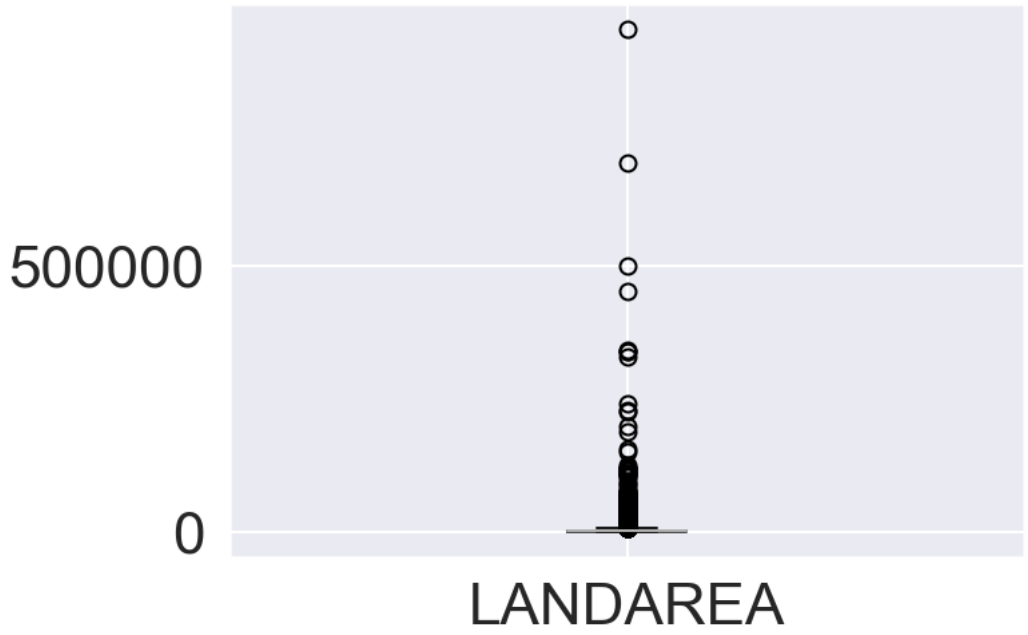
[270]:
```
up = properties["GBA"].quantile(0.75)
lo = properties["GBA"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["GBA_OUTLIER"]=properties["GBA"].apply(price_outlier)
properties[["GBA_OUTLIER"]].boxplot()
plt.show()
```

[270]: <AxesSubplot:>



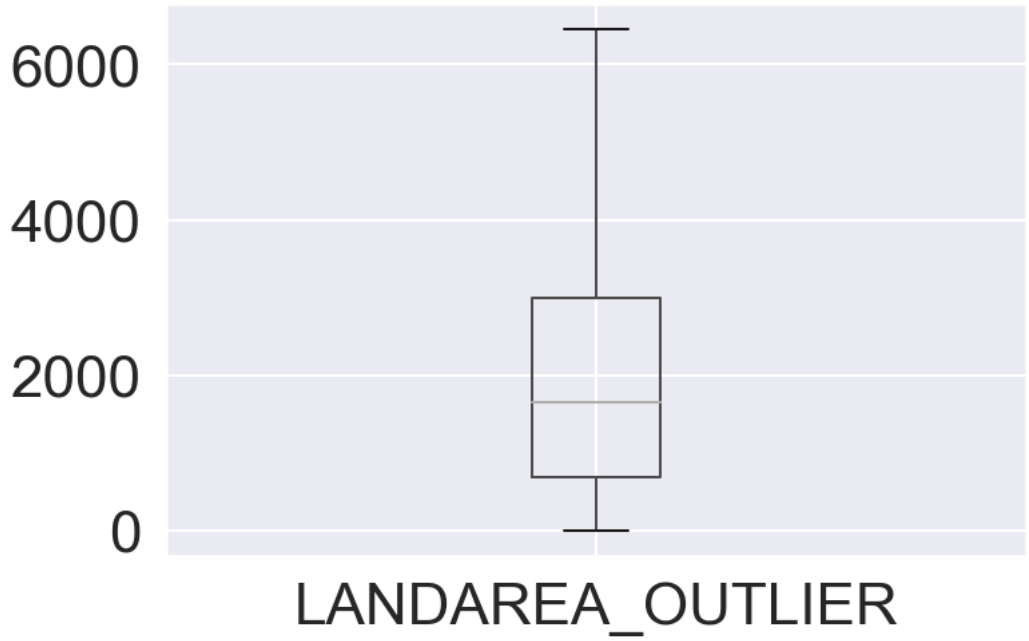
[271]:
```
properties[["LANDAREA"]].boxplot()
plt.show()
```

[271]: <AxesSubplot:>



[272]:
```
up = properties["LANDAREA"].quantile(0.75)
lo = properties["LANDAREA"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["LANDAREA_OUTLIER"]=properties["LANDAREA"].apply(price_outlier)
properties[["LANDAREA_OUTLIER"]].boxplot()
plt.show()
```

[272]: <AxesSubplot:>

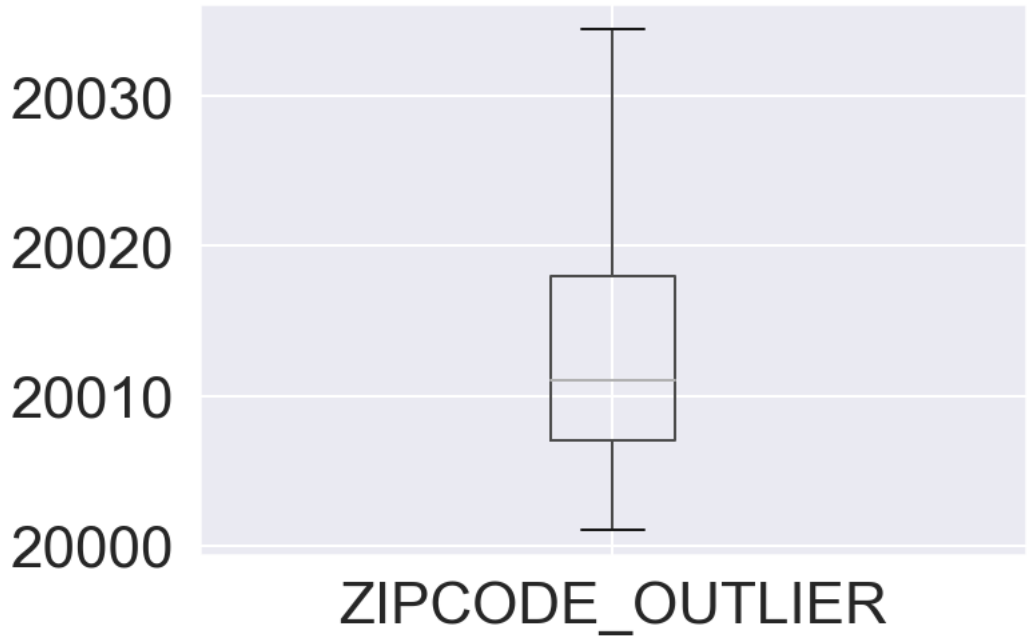


[273]:
```
properties[["ZIPCODE"]].boxplot()
plt.show()
```

[273]: <AxesSubplot:>

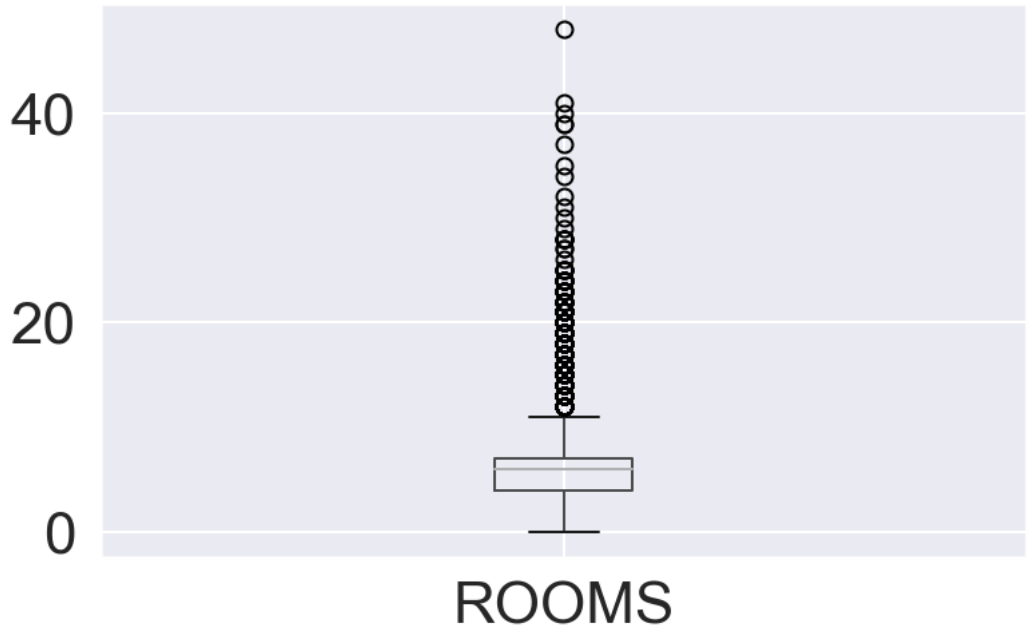
[274]:
```
up = properties["ZIPCODE"].quantile(0.75)
lo = properties["ZIPCODE"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["ZIPCODE_OUTLIER"]=properties["ZIPCODE"].apply(price_outlier)
properties[["ZIPCODE_OUTLIER"]].boxplot()
plt.show()
```

[274]: <AxesSubplot:>



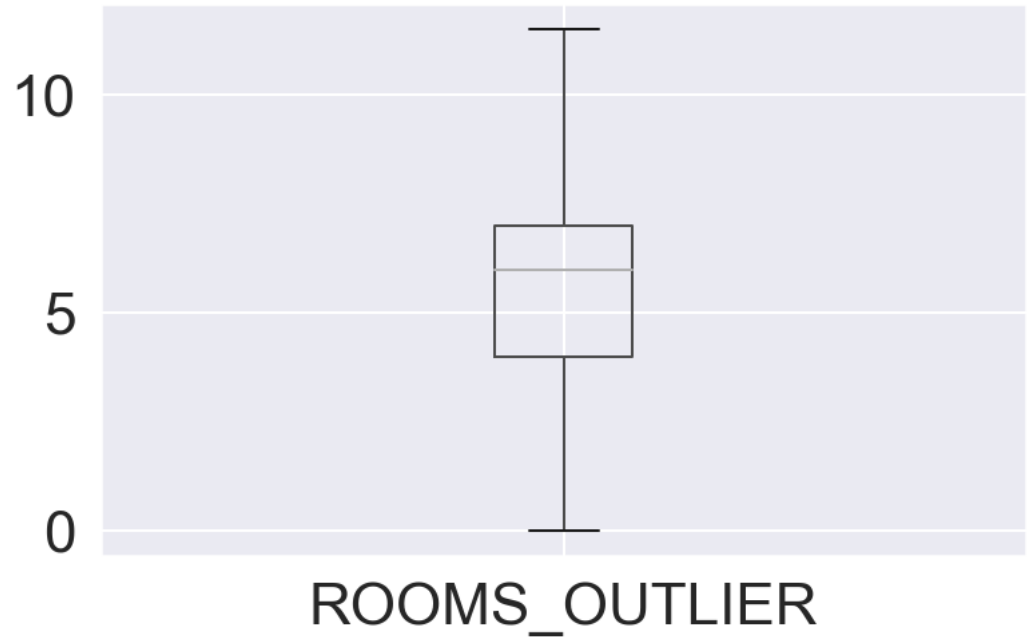
[275]:
```
properties[["ROOMS"]].boxplot()
plt.show()
```

[275]: <AxesSubplot:>



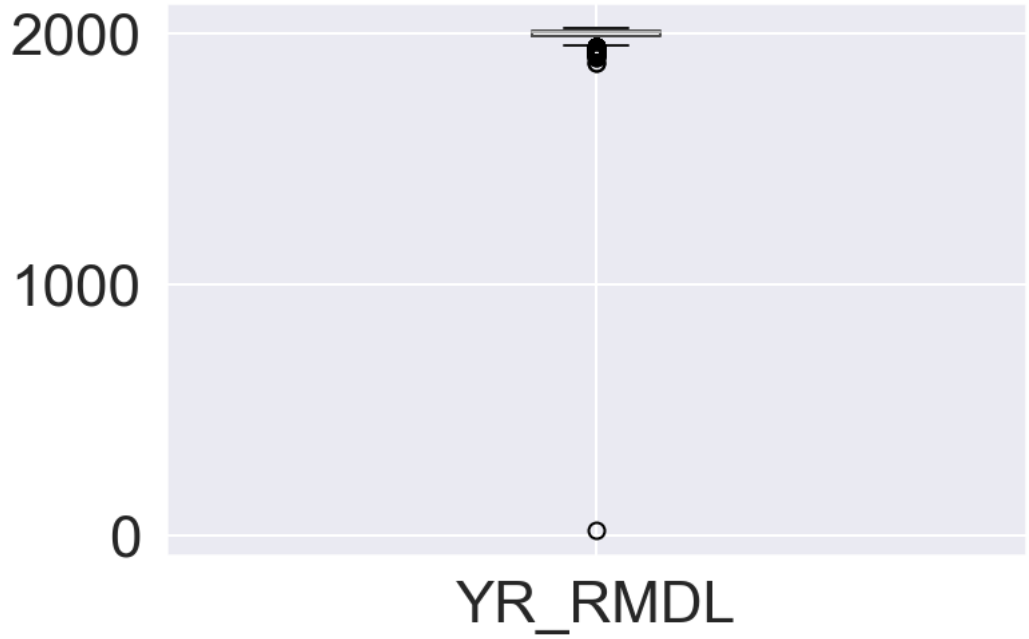
[276]:
```
up = properties["ROOMS"].quantile(0.75)
lo = properties["ROOMS"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["ROOMS_OUTLIER"]=properties["ROOMS"].apply(price_outlier)
properties[["ROOMS_OUTLIER"]].boxplot()
plt.show()
```

[276]: <AxesSubplot:>



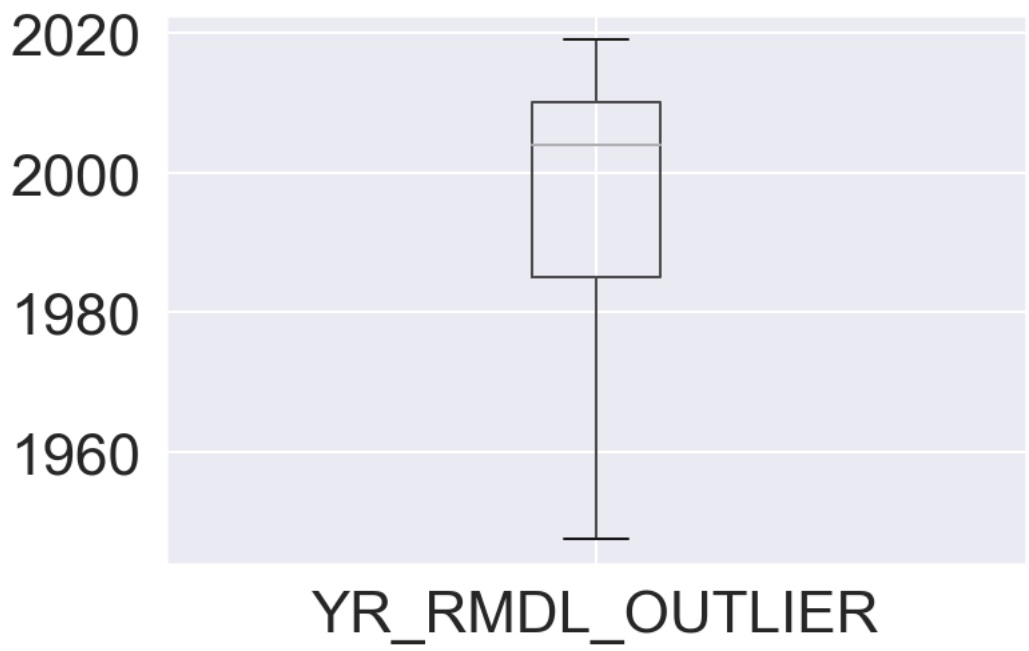
[277]:
```
properties[["YR_RMDL"]].boxplot()
plt.show()
```

[277]: <AxesSubplot:>



[278]:
```
up = properties["YR_RMDL"].quantile(0.75)
lo = properties["YR_RMDL"].quantile(0.25)
dif=up-lo
up=up+1.5*dif
lo=lo-1.5*dif
def price_outlier(x):
    if x>up:
        return up
    elif x<lo:
        return lo
    else:
        return x
properties["YR_RMDL_OUTLIER"]=properties["YR_RMDL"].apply(price_outlier)
properties[["YR_RMDL_OUTLIER"]].boxplot()
plt.show()
```

[278]: <AxesSubplot:>



create new dataframe properties_new to store features that may to be used in regression fill the missed values and transfer object type values into integer type

[279]:
```
properties_new=properties.drop(["CMPLX_NUM"], axis=1)
properties_new=properties_new.drop(["LIVING_GBA"],axis=1)
properties_new=properties_new.drop(["SALEDATE"],axis=1)
properties_new=properties_new.drop(["FIREPLACES"], axis=1)
```

```
properties_new["YR_RMDL"].fillna(1947, inplace=True)
properties_new["YR_RMDL_OUTLIER"].fillna(1947, inplace=True)

properties_new["ZIPCODE_OUTLIER"].fillna(properties_new["ZIPCODE_OUTLIER"].
 ↪mean(),inplace=True)
properties_new["LATITUDE"].fillna(properties_new["LATITUDE"].
 ↪mean(),inplace=True)
properties_new["LONGITUDE"].fillna(properties_new["LONGITUDE"].
 ↪mean(),inplace=True)
properties_new["X"].fillna(properties_new["X"].mean(),inplace=True)
properties_new["Y"].fillna(properties_new["Y"].mean(),inplace=True)
properties_new["CENSUS_TRACT"].fillna(properties_new["CENSUS_TRACT"].
 ↪mean(),inplace=True)

def discrete_AC(x):
    if x=='Y':
        return 1
    if x=='N':
        return -1
    if x=='0':
        return 0

def discrete_GRADE(x):
    if x=='Exceptional-A':
        return 1
    elif x=='Exceptional-B':
        return 2
    elif x=='Exceptional-C':
        return 3
    elif x=='Exceptional-D':
        return 4
    elif x=='Fair Quality':
        return 5
    elif x=='Avereage':
        return 6
    elif x=='Above Average':
        return 7
    elif x=='Good Quality':
        return 8
    elif x=='Very Good':
        return 9
    elif x=='Excellent':
        return 10
    elif x=='Superior':
        return 11
    else:
        return 6
```

```
def discrete_CNDTN(x):
    if x=='Poor':
        return 1
    elif x=='Fair':
        return 2
    elif x=='Good':
        return 3
    elif x=='Very Good':
        return 4
    elif x=='Excellent':
        return 5
    else:
        return 2


properties_new['GRADE_V']= properties['GRADE'].apply(discrete_GRADE)
properties_new['AC_V'] = properties['AC'].apply(discrete_AC)
properties_new['CNDTN_V'] = properties['CNDTN'].apply(discrete_CNDTN)

properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158957 entries, 0 to 158956
Data columns (total 53 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   BATHRM              158957 non-null  int64
 1   HF_BATHRM           158957 non-null  int64
 2   HEAT                158957 non-null  object
 3   AC                  158957 non-null  object
 4   NUM_UNITS           106696 non-null  float64
 5   ROOMS               158957 non-null  int64
 6   BEDRM               158957 non-null  int64
 7   AYB                 158686 non-null  float64
 8   YR_RMDL             158957 non-null  float64
 9   EYB                 158957 non-null  int64
 10  STORIES             106652 non-null  float64
 11  PRICE               98216 non-null   float64
 12  QUALIFIED           158957 non-null  object
 13  SALE_NUM            158957 non-null  int64
 14  GBA                 106696 non-null  float64
 15  BLDG_NUM            158957 non-null  int64
 16  STYLE               106696 non-null  object
 17  STRUCT              106696 non-null  object
 18  GRADE               106696 non-null  object
 19  CNDTN               106696 non-null  object
 20  EXTWALL             106696 non-null  object
```

```
 21  ROOF                106696 non-null  object
 22  INTWALL             106696 non-null  object
 23  KITCHENS            106695 non-null  float64
 24  USECODE             158957 non-null  int64
 25  LANDAREA            158957 non-null  int64
 26  GIS_LAST_MOD_DTTM   158957 non-null  object
 27  SOURCE              158957 non-null  object
 28  FULLADDRESS         106040 non-null  object
 29  CITY                106051 non-null  object
 30  STATE               106051 non-null  object
 31  ZIPCODE             158956 non-null  float64
 32  NATIONALGRID        106051 non-null  object
 33  LATITUDE            158957 non-null  float64
 34  LONGITUDE           158957 non-null  float64
 35  ASSESSMENT_NBHD     158956 non-null  object
 36  ASSESSMENT_SUBNBHD  126406 non-null  object
 37  CENSUS_TRACT        158957 non-null  float64
 38  CENSUS_BLOCK        106051 non-null  object
 39  WARD                158956 non-null  object
 40  SQUARE              158957 non-null  object
 41  X                   158957 non-null  float64
 42  Y                   158957 non-null  float64
 43  QUADRANT            158720 non-null  object
 44  PRICE_OUTLIER       98216 non-null   float64
 45  GBA_OUTLIER         106696 non-null  float64
 46  LANDAREA_OUTLIER    158957 non-null  float64
 47  ZIPCODE_OUTLIER     158957 non-null  float64
 48  ROOMS_OUTLIER       158957 non-null  float64
 49  YR_RMDL_OUTLIER     158957 non-null  float64
 50  GRADE_V             158957 non-null  int64
 51  AC_V                158957 non-null  int64
 52  CNDTN_V             158957 non-null  int64
dtypes: float64(19), int64(12), object(22)
memory usage: 65.5+ MB
```

display correlation between features selected in properties_new

[280]:
```
corrMatrix=properties_new.corr()
corrMatrix
```

[280]:

	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	\
BATHRM	1.000000	0.248759	0.403431	0.677751	0.655563	
HF_BATHRM	0.248759	1.000000	-0.151393	0.353825	0.375568	
NUM_UNITS	0.403431	-0.151393	1.000000	0.524724	0.330551	
ROOMS	0.677751	0.353825	0.524724	1.000000	0.841309	
BEDRM	0.655563	0.375568	0.330551	0.841309	1.000000	
AYB	-0.039087	-0.009372	-0.104280	-0.219367	-0.242955	
YR_RMDL	0.177417	0.038846	0.045385	0.009082	0.034834	

```
EYB               0.269444   0.235645  -0.069551  0.158847  0.168913
STORIES           0.042349   0.033270   0.019268  0.039352  0.040859
PRICE            -0.006536  -0.013113  -0.000469 -0.038241 -0.043331
SALE_NUM          0.068399   0.002193   0.013986 -0.034828 -0.024835
GBA               0.682977   0.272598   0.264009  0.679725  0.631736
BLDG_NUM         -0.000207  -0.009322   0.010247 -0.003255 -0.006528
KITCHENS          0.423178  -0.132895   0.930610  0.525796  0.346781
USECODE           0.121070  -0.229586   0.797330 -0.042795 -0.131676
LANDAREA          0.287648   0.212534  -0.038679  0.348184  0.321066
ZIPCODE          -0.063410  -0.020164  -0.099460 -0.053109 -0.071962
LATITUDE          0.130588   0.126221  -0.117124  0.120236  0.144971
LONGITUDE        -0.184810  -0.058719   0.037277  0.006756  0.028338
CENSUS_TRACT     -0.196311  -0.072888   0.060406 -0.075593 -0.072028
X                -0.184808  -0.058759   0.037265  0.006394  0.028006
Y                 0.129690   0.126109  -0.116872  0.119610  0.144630
PRICE_OUTLIER     0.521705   0.316287   0.003934  0.335612  0.369623
GBA_OUTLIER       0.693548   0.252602   0.354935  0.692509  0.653769
LANDAREA_OUTLIER  0.408990   0.349420  -0.059979  0.576120  0.581545
ZIPCODE_OUTLIER  -0.086604  -0.009391  -0.100817 -0.050811 -0.068618
ROOMS_OUTLIER     0.655036   0.388276   0.442279  0.969619  0.849709
YR_RMDL_OUTLIER   0.180057   0.039816   0.046424  0.009977  0.036230
GRADE_V           0.350494   0.297197  -0.004338  0.388408  0.404105
AC_V              0.133513   0.121314  -0.090888 -0.165388 -0.144600
CNDTN_V           0.436109   0.362503  -0.056158  0.363536  0.373965

                      AYB   YR_RMDL       EYB   STORIES     PRICE  …  \
BATHRM          -0.039087  0.177417  0.269444  0.042349 -0.006536  …
HF_BATHRM       -0.009372  0.038846  0.235645  0.033270 -0.013113  …
NUM_UNITS       -0.104280  0.045385 -0.069551  0.019268 -0.000469  …
ROOMS           -0.219367  0.009082  0.158847  0.039352 -0.038241  …
BEDRM           -0.242955  0.034834  0.168913  0.040859 -0.043331  …
AYB              1.000000 -0.300029  0.660668 -0.011098  0.056041  …
YR_RMDL         -0.300029  1.000000 -0.128281  0.015851  0.060862  …
EYB              0.660668 -0.128281  1.000000  0.040214  0.049627  …
STORIES         -0.011098  0.015851  0.040214  1.000000  0.042337  …
PRICE            0.056041  0.060862  0.049627  0.042337  1.000000  …
SALE_NUM         0.017096  0.177472  0.053879  0.012937 -0.015942  …
GBA              0.006706  0.129676  0.311207  0.056026  0.600576  …
BLDG_NUM         0.005310 -0.009148  0.014750 -0.003790  0.003048  …
KITCHENS        -0.108947  0.068411 -0.048890  0.021013  0.045213  …
USECODE          0.078079  0.081037 -0.059968  0.020700  0.026404  …
LANDAREA        -0.040503 -0.010193  0.081796 -0.009711 -0.005457  …
ZIPCODE          0.175098 -0.101543  0.041599 -0.026886  0.001467  …
LATITUDE        -0.122697  0.033892 -0.082219 -0.003654  0.044505  …
LONGITUDE        0.032979 -0.089952  0.005107 -0.031842 -0.095247  …
CENSUS_TRACT     0.115762 -0.095996  0.036704 -0.025058 -0.069743  …
X                0.033352 -0.089836  0.005331 -0.031811 -0.095255  …
```

```
Y                -0.122575  0.033434 -0.082202 -0.003587  0.044578  …
PRICE_OUTLIER    -0.056893  0.193318  0.240964  0.041493  0.200547  …
GBA_OUTLIER      -0.022830  0.152182  0.299226  0.061558  0.475272  …
LANDAREA_OUTLIER -0.109334 -0.047793  0.112064 -0.031338 -0.024632  …
ZIPCODE_OUTLIER   0.257436 -0.140304  0.039923 -0.027249  0.007717  …
ROOMS_OUTLIER    -0.240916  0.008794  0.171077  0.042877 -0.043977  …
YR_RMDL_OUTLIER  -0.305762  0.984948 -0.130467  0.016210  0.062260  …
GRADE_V          -0.161388  0.042623  0.188069  0.034808 -0.015916  …
AC_V              0.274723  0.279374  0.234621  0.011967  0.039829  …
CNDTN_V           0.017594  0.241744  0.363853  0.034276 -0.013148  …

                         Y  PRICE_OUTLIER  GBA_OUTLIER  LANDAREA_OUTLIER  \
BATHRM            0.129690       0.521705     0.693548          0.408990
HF_BATHRM         0.126109       0.316287     0.252602          0.349420
NUM_UNITS        -0.116872       0.003934     0.354935         -0.059979
ROOMS             0.119610       0.335612     0.692509          0.576120
BEDRM             0.144630       0.369623     0.653769          0.581545
AYB              -0.122575      -0.056893    -0.022830         -0.109334
YR_RMDL           0.033434       0.193318     0.152182         -0.047793
EYB              -0.082202       0.240964     0.299226          0.112064
STORIES          -0.003587       0.041493     0.061558         -0.031338
PRICE             0.044578       0.200547     0.475272         -0.024632
SALE_NUM         -0.012931       0.233897    -0.010961         -0.090166
GBA               0.172020       0.508968     0.875709          0.402785
BLDG_NUM         -0.013135       0.022988    -0.007458          0.032659
KITCHENS         -0.105825       0.044597     0.359786         -0.051927
USECODE          -0.110956      -0.026169     0.280514         -0.317411
LANDAREA          0.129156       0.241556     0.243575          0.502116
ZIPCODE          -0.126755      -0.110564    -0.118829          0.067926
LATITUDE          0.997937       0.217636     0.209745          0.252059
LONGITUDE        -0.499302      -0.404566    -0.445808         -0.001197
CENSUS_TRACT     -0.563146      -0.339541    -0.385577         -0.082175
X                -0.499799      -0.404282    -0.445638         -0.002448
Y                 1.000000       0.217537     0.208824          0.251299
PRICE_OUTLIER     0.217537       1.000000     0.519047          0.225177
GBA_OUTLIER       0.208824       0.519047     1.000000          0.422898
LANDAREA_OUTLIER  0.251299       0.225177     0.422898          1.000000
ZIPCODE_OUTLIER  -0.238783      -0.186485    -0.119809          0.183790
ROOMS_OUTLIER     0.141212       0.356765     0.710776          0.605935
YR_RMDL_OUTLIER   0.034566       0.197658     0.155856         -0.048277
GRADE_V           0.196839       0.396466     0.386785          0.393203
AC_V             -0.052132       0.215735     0.154512         -0.121965
CNDTN_V           0.036512       0.362584     0.222876          0.281408

                  ZIPCODE_OUTLIER  ROOMS_OUTLIER  YR_RMDL_OUTLIER   GRADE_V  \
BATHRM                  -0.086604       0.655036         0.180057  0.350494
HF_BATHRM               -0.009391       0.388276         0.039816  0.297197
```

NUM_UNITS	-0.100817	0.442279	0.046424	-0.004338
ROOMS	-0.050811	0.969619	0.009977	0.388408
BEDRM	-0.068618	0.849709	0.036230	0.404105
AYB	0.257436	-0.240916	-0.305762	-0.161388
YR_RMDL	-0.140304	0.008794	0.984948	0.042623
EYB	0.039923	0.171077	-0.130467	0.188069
STORIES	-0.027249	0.042877	0.016210	0.034808
PRICE	0.007717	-0.043977	0.062260	-0.015916
SALE_NUM	-0.055612	-0.037411	0.180299	-0.035956
GBA	-0.098597	0.640273	0.132755	0.255623
BLDG_NUM	0.001500	-0.003462	-0.009319	0.004025
KITCHENS	-0.105917	0.445626	0.069938	0.001697
USECODE	-0.055603	-0.106652	0.082015	-0.159118
LANDAREA	0.073617	0.308890	-0.010287	0.162152
ZIPCODE	0.628241	-0.058317	-0.103310	-0.066222
LATITUDE	-0.239380	0.141808	0.035029	0.196656
LONGITUDE	0.110920	0.007378	-0.091386	-0.297789
CENSUS_TRACT	0.236600	-0.084903	-0.097935	-0.300740
X	0.110164	0.007085	-0.091258	-0.297827
Y	-0.238783	0.141212	0.034566	0.196839
PRICE_OUTLIER	-0.186485	0.356765	0.197658	0.396466
GBA_OUTLIER	-0.119809	0.710776	0.155856	0.386785
LANDAREA_OUTLIER	0.183790	0.605935	-0.048277	0.393203
ZIPCODE_OUTLIER	1.000000	-0.056629	-0.142729	-0.087200
ROOMS_OUTLIER	-0.056629	1.000000	0.009753	0.439794
YR_RMDL_OUTLIER	-0.142729	0.009753	1.000000	0.043604
GRADE_V	-0.087200	0.439794	0.043604	1.000000
AC_V	0.044368	-0.168075	0.283590	-0.001344
CNDTN_V	-0.016598	0.389742	0.245948	0.312834

	AC_V	CNDTN_V
BATHRM	0.133513	0.436109
HF_BATHRM	0.121314	0.362503
NUM_UNITS	-0.090888	-0.056158
ROOMS	-0.165388	0.363536
BEDRM	-0.144600	0.373965
AYB	0.274723	0.017594
YR_RMDL	0.279374	0.241744
EYB	0.234621	0.363853
STORIES	0.011967	0.034276
PRICE	0.039829	-0.013148
SALE_NUM	0.133720	0.145534
GBA	0.150065	0.240516
BLDG_NUM	0.002653	0.017796
KITCHENS	-0.072496	-0.033797
USECODE	0.125884	-0.196753
LANDAREA	-0.026001	0.173798

```
ZIPCODE           0.043157 -0.026896
LATITUDE         -0.051921  0.036934
LONGITUDE        -0.142779 -0.002746
CENSUS_TRACT     -0.033512 -0.030310
X                -0.142227 -0.002501
Y                -0.052132  0.036512
PRICE_OUTLIER     0.215735  0.362584
GBA_OUTLIER       0.154512  0.222876
LANDAREA_OUTLIER -0.121965  0.281408
ZIPCODE_OUTLIER   0.044368 -0.016598
ROOMS_OUTLIER    -0.168075  0.389742
YR_RMDL_OUTLIER   0.283590  0.245948
GRADE_V          -0.001344  0.312834
AC_V              1.000000  0.229010
CNDTN_V           0.229010  1.000000

[31 rows x 31 columns]
```

display correlation between PRICE_OUTLIER and other features, just to prepare before selecting features

[281]:
```
corrMatrix["PRICE_OUTLIER"].sort_values(ascending=False)
```

[281]:
```
PRICE_OUTLIER       1.000000
BATHRM              0.521705
GBA_OUTLIER         0.519047
GBA                 0.508968
GRADE_V             0.396466
BEDRM               0.369623
CNDTN_V             0.362584
ROOMS_OUTLIER       0.356765
ROOMS               0.335612
HF_BATHRM           0.316287
LANDAREA            0.241556
EYB                 0.240964
SALE_NUM            0.233897
LANDAREA_OUTLIER    0.225177
LATITUDE            0.217636
Y                   0.217537
AC_V                0.215735
PRICE               0.200547
YR_RMDL_OUTLIER     0.197658
YR_RMDL             0.193318
KITCHENS            0.044597
STORIES             0.041493
BLDG_NUM            0.022988
NUM_UNITS           0.003934
USECODE            -0.026169
```

```
AYB                -0.056893
ZIPCODE            -0.110564
ZIPCODE_OUTLIER    -0.186485
CENSUS_TRACT       -0.339541
X                  -0.404282
LONGITUDE          -0.404566
Name: PRICE_OUTLIER, dtype: float64
```

drop features not used in regression

[282]:
```
properties_new.drop(["GBA"], axis=1, inplace=True)
properties_new.drop(["ROOMS"], axis=1, inplace=True)
properties_new.drop(["PRICE"], axis=1, inplace=True)
properties_new.drop(["YR_RMDL"], axis=1, inplace=True)
properties_new.drop(["STORIES"], axis=1, inplace=True)
properties_new.drop(["KITCHENS"], axis=1, inplace=True)
properties_new.drop(["BLDG_NUM"], axis=1, inplace=True)
properties_new.drop(["NUM_UNITS"], axis=1, inplace=True)
properties_new.drop(["USECODE"], axis=1, inplace=True)
properties_new.drop(["AYB"], axis=1, inplace=True)
properties_new.drop(["ZIPCODE"], axis=1, inplace=True)
properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158957 entries, 0 to 158956
Data columns (total 42 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   BATHRM              158957 non-null  int64
 1   HF_BATHRM           158957 non-null  int64
 2   HEAT                158957 non-null  object
 3   AC                  158957 non-null  object
 4   BEDRM               158957 non-null  int64
 5   EYB                 158957 non-null  int64
 6   QUALIFIED           158957 non-null  object
 7   SALE_NUM            158957 non-null  int64
 8   STYLE               106696 non-null  object
 9   STRUCT              106696 non-null  object
 10  GRADE               106696 non-null  object
 11  CNDTN               106696 non-null  object
 12  EXTWALL             106696 non-null  object
 13  ROOF                106696 non-null  object
 14  INTWALL             106696 non-null  object
 15  LANDAREA            158957 non-null  int64
 16  GIS_LAST_MOD_DTTM   158957 non-null  object
 17  SOURCE              158957 non-null  object
 18  FULLADDRESS         106040 non-null  object
 19  CITY                106051 non-null  object
```

```
 20  STATE                106051 non-null  object
 21  NATIONALGRID         106051 non-null  object
 22  LATITUDE             158957 non-null  float64
 23  LONGITUDE            158957 non-null  float64
 24  ASSESSMENT_NBHD      158956 non-null  object
 25  ASSESSMENT_SUBNBHD   126406 non-null  object
 26  CENSUS_TRACT         158957 non-null  float64
 27  CENSUS_BLOCK         106051 non-null  object
 28  WARD                 158956 non-null  object
 29  SQUARE               158957 non-null  object
 30  X                    158957 non-null  float64
 31  Y                    158957 non-null  float64
 32  QUADRANT             158720 non-null  object
 33  PRICE_OUTLIER        98216 non-null   float64
 34  GBA_OUTLIER          106696 non-null  float64
 35  LANDAREA_OUTLIER     158957 non-null  float64
 36  ZIPCODE_OUTLIER      158957 non-null  float64
 37  ROOMS_OUTLIER        158957 non-null  float64
 38  YR_RMDL_OUTLIER      158957 non-null  float64
 39  GRADE_V              158957 non-null  int64
 40  AC_V                 158957 non-null  int64
 41  CNDTN_V              158957 non-null  int64
dtypes: float64(11), int64(9), object(22)
memory usage: 52.1+ MB
```

transfer QUADRANT into integers

[283]:
```
properties_temp=pd.DataFrame()
properties_temp["QUADRANT"]=properties["QUADRANT"]
properties_temp=pd.get_dummies(properties_temp, prefix = properties_temp.
 ↪columns).astype(int)
properties_temp.value_counts()
```

[283]:
```
QUADRANT_NE  QUADRANT_NW  QUADRANT_SE  QUADRANT_SW
0            1            0            0              89736
1            0            0            0              37675
0            0            1            0              27224
                          0            1               4085
                                       0                237
dtype: int64
```

fill missing data of QUADRANT
QUADRANT_NW has most 1, so fill missing data with 0100

[284]:
```
properties_new["QUADRANT_NE"] = properties_temp["QUADRANT_NE"]
properties_new["QUADRANT_NW"] = properties_temp["QUADRANT_NW"]
properties_new["QUADRANT_SE"] = properties_temp["QUADRANT_SE"]
properties_new["QUADRANT_SW"] = properties_temp["QUADRANT_SW"]
```

```
properties_new["QUADRANT_NE"].fillna(0, inplace=True)
properties_new["QUADRANT_NW"].fillna(1, inplace=True)
properties_new["QUADRANT_SE"].fillna(0, inplace=True)
properties_new["QUADRANT_SW"].fillna(0, inplace=True)

properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158957 entries, 0 to 158956
Data columns (total 46 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   BATHRM               158957 non-null  int64
 1   HF_BATHRM            158957 non-null  int64
 2   HEAT                 158957 non-null  object
 3   AC                   158957 non-null  object
 4   BEDRM                158957 non-null  int64
 5   EYB                  158957 non-null  int64
 6   QUALIFIED            158957 non-null  object
 7   SALE_NUM             158957 non-null  int64
 8   STYLE                106696 non-null  object
 9   STRUCT               106696 non-null  object
 10  GRADE                106696 non-null  object
 11  CNDTN                106696 non-null  object
 12  EXTWALL              106696 non-null  object
 13  ROOF                 106696 non-null  object
 14  INTWALL              106696 non-null  object
 15  LANDAREA             158957 non-null  int64
 16  GIS_LAST_MOD_DTTM    158957 non-null  object
 17  SOURCE               158957 non-null  object
 18  FULLADDRESS          106040 non-null  object
 19  CITY                 106051 non-null  object
 20  STATE                106051 non-null  object
 21  NATIONALGRID         106051 non-null  object
 22  LATITUDE             158957 non-null  float64
 23  LONGITUDE            158957 non-null  float64
 24  ASSESSMENT_NBHD      158956 non-null  object
 25  ASSESSMENT_SUBNBHD   126406 non-null  object
 26  CENSUS_TRACT         158957 non-null  float64
 27  CENSUS_BLOCK         106051 non-null  object
 28  WARD                 158956 non-null  object
 29  SQUARE               158957 non-null  object
 30  X                    158957 non-null  float64
 31  Y                    158957 non-null  float64
 32  QUADRANT             158720 non-null  object
 33  PRICE_OUTLIER        98216 non-null   float64
 34  GBA_OUTLIER          106696 non-null  float64
```

```
 35  LANDAREA_OUTLIER    158957 non-null  float64
 36  ZIPCODE_OUTLIER     158957 non-null  float64
 37  ROOMS_OUTLIER       158957 non-null  float64
 38  YR_RMDL_OUTLIER     158957 non-null  float64
 39  GRADE_V             158957 non-null  int64
 40  AC_V                158957 non-null  int64
 41  CNDTN_V             158957 non-null  int64
 42  QUADRANT_NE         158957 non-null  int32
 43  QUADRANT_NW         158957 non-null  int32
 44  QUADRANT_SE         158957 non-null  int32
 45  QUADRANT_SW         158957 non-null  int32
dtypes: float64(11), int32(4), int64(9), object(22)
memory usage: 54.6+ MB
```

select only features after preprocess

[285]:
```
properties_new.
↪drop(['HEAT','AC','QUALIFIED','STYLE','STRUCT','GRADE','CNDTN','EXTWALL','ROOF','INTWALL','
                    ␣
↪'FULLADDRESS','CITY','STATE','NATIONALGRID','ASSESSMENT_NBHD','ASSESSMENT_SUBNBHD','CENSUS_
                     'QUADRANT'], axis=1, inplace=True)
```

[286]:
```
properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 158957 entries, 0 to 158956
Data columns (total 24 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   BATHRM            158957 non-null  int64
 1   HF_BATHRM         158957 non-null  int64
 2   BEDRM             158957 non-null  int64
 3   EYB               158957 non-null  int64
 4   SALE_NUM          158957 non-null  int64
 5   LANDAREA          158957 non-null  int64
 6   LATITUDE          158957 non-null  float64
 7   LONGITUDE         158957 non-null  float64
 8   CENSUS_TRACT      158957 non-null  float64
 9   X                 158957 non-null  float64
 10  Y                 158957 non-null  float64
 11  PRICE_OUTLIER     98216 non-null   float64
 12  GBA_OUTLIER       106696 non-null  float64
 13  LANDAREA_OUTLIER  158957 non-null  float64
 14  ZIPCODE_OUTLIER   158957 non-null  float64
 15  ROOMS_OUTLIER     158957 non-null  float64
 16  YR_RMDL_OUTLIER   158957 non-null  float64
 17  GRADE_V           158957 non-null  int64
 18  AC_V              158957 non-null  int64
```

```
 19  CNDTN_V           158957 non-null  int64
 20  QUADRANT_NE       158957 non-null  int32
 21  QUADRANT_NW       158957 non-null  int32
 22  QUADRANT_SE       158957 non-null  int32
 23  QUADRANT_SW       158957 non-null  int32
dtypes: float64(11), int32(4), int64(9)
memory usage: 27.9 MB
```

drop rows if with missing data features

[287]:
```
properties_new.dropna(inplace=True)
properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57900 entries, 0 to 106695
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            57900 non-null  int64
 1   HF_BATHRM         57900 non-null  int64
 2   BEDRM             57900 non-null  int64
 3   EYB               57900 non-null  int64
 4   SALE_NUM          57900 non-null  int64
 5   LANDAREA          57900 non-null  int64
 6   LATITUDE          57900 non-null  float64
 7   LONGITUDE         57900 non-null  float64
 8   CENSUS_TRACT      57900 non-null  float64
 9   X                 57900 non-null  float64
 10  Y                 57900 non-null  float64
 11  PRICE_OUTLIER     57900 non-null  float64
 12  GBA_OUTLIER       57900 non-null  float64
 13  LANDAREA_OUTLIER  57900 non-null  float64
 14  ZIPCODE_OUTLIER   57900 non-null  float64
 15  ROOMS_OUTLIER     57900 non-null  float64
 16  YR_RMDL_OUTLIER   57900 non-null  float64
 17  GRADE_V           57900 non-null  int64
 18  AC_V              57900 non-null  int64
 19  CNDTN_V           57900 non-null  int64
 20  QUADRANT_NE       57900 non-null  int32
 21  QUADRANT_NW       57900 non-null  int32
 22  QUADRANT_SE       57900 non-null  int32
 23  QUADRANT_SW       57900 non-null  int32
dtypes: float64(11), int32(4), int64(9)
memory usage: 10.2 MB
```

display correlation between PRICE_OUTLIER and other features

[288]:
```
corrMatrix=properties_new.corr()
corrMatrix["PRICE_OUTLIER"].sort_values()
```

[288]:
```
LONGITUDE          -0.573791
X                  -0.573364
CENSUS_TRACT       -0.478770
ZIPCODE_OUTLIER    -0.253607
QUADRANT_NE        -0.246451
QUADRANT_SE        -0.236096
QUADRANT_SW        -0.062208
LANDAREA_OUTLIER    0.194777
LANDAREA            0.227070
Y                   0.254135
LATITUDE            0.254140
SALE_NUM            0.277428
HF_BATHRM           0.286911
YR_RMDL_OUTLIER     0.299789
AC_V                0.317515
EYB                 0.344025
ROOMS_OUTLIER       0.362077
BEDRM               0.365048
CNDTN_V             0.395281
QUADRANT_NW         0.432567
GRADE_V             0.438569
BATHRM              0.517514
GBA_OUTLIER         0.519047
PRICE_OUTLIER       1.000000
Name: PRICE_OUTLIER, dtype: float64
```

divide into train set and test set, by ratio of 8:2

[289]:
```
from sklearn import model_selection
x_train,x_test,y_train,y_test=model_selection.
 ↪train_test_split(properties_new,properties_new["PRICE_OUTLIER"],test_size=0.
 ↪2,random_state=1234)

x_train.drop(["PRICE_OUTLIER"],axis=1, inplace=True)
x_test.drop(["PRICE_OUTLIER"],axis=1, inplace=True)
x_train.info()
x_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 46320 entries, 40565 to 48491
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            46320 non-null  int64
 1   HF_BATHRM         46320 non-null  int64
 2   BEDRM             46320 non-null  int64
 3   EYB               46320 non-null  int64
 4   SALE_NUM          46320 non-null  int64
```

```
 5   LANDAREA          46320 non-null  int64
 6   LATITUDE          46320 non-null  float64
 7   LONGITUDE         46320 non-null  float64
 8   CENSUS_TRACT      46320 non-null  float64
 9   X                 46320 non-null  float64
 10  Y                 46320 non-null  float64
 11  GBA_OUTLIER       46320 non-null  float64
 12  LANDAREA_OUTLIER  46320 non-null  float64
 13  ZIPCODE_OUTLIER   46320 non-null  float64
 14  ROOMS_OUTLIER     46320 non-null  float64
 15  YR_RMDL_OUTLIER   46320 non-null  float64
 16  GRADE_V           46320 non-null  int64
 17  AC_V              46320 non-null  int64
 18  CNDTN_V           46320 non-null  int64
 19  QUADRANT_NE       46320 non-null  int32
 20  QUADRANT_NW       46320 non-null  int32
 21  QUADRANT_SE       46320 non-null  int32
 22  QUADRANT_SW       46320 non-null  int32
dtypes: float64(10), int32(4), int64(9)
memory usage: 7.8 MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11580 entries, 96365 to 35580
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            11580 non-null  int64
 1   HF_BATHRM         11580 non-null  int64
 2   BEDRM             11580 non-null  int64
 3   EYB               11580 non-null  int64
 4   SALE_NUM          11580 non-null  int64
 5   LANDAREA          11580 non-null  int64
 6   LATITUDE          11580 non-null  float64
 7   LONGITUDE         11580 non-null  float64
 8   CENSUS_TRACT      11580 non-null  float64
 9   X                 11580 non-null  float64
 10  Y                 11580 non-null  float64
 11  GBA_OUTLIER       11580 non-null  float64
 12  LANDAREA_OUTLIER  11580 non-null  float64
 13  ZIPCODE_OUTLIER   11580 non-null  float64
 14  ROOMS_OUTLIER     11580 non-null  float64
 15  YR_RMDL_OUTLIER   11580 non-null  float64
 16  GRADE_V           11580 non-null  int64
 17  AC_V              11580 non-null  int64
 18  CNDTN_V           11580 non-null  int64
 19  QUADRANT_NE       11580 non-null  int32
 20  QUADRANT_NW       11580 non-null  int32
 21  QUADRANT_SE       11580 non-null  int32
 22  QUADRANT_SW       11580 non-null  int32
```

```
dtypes: float64(10), int32(4), int64(9)
memory usage: 1.9 MB
```

using linear regression model of sklearn package

[290]:
```
from sklearn import linear_model

properties_train_X=properties_new.drop(["PRICE_OUTLIER"], axis=1)
properties_train_Y=properties_new["PRICE_OUTLIER"]

regr = linear_model.LinearRegression()
regr.fit(properties_train_X, properties_train_Y)

print(regr.intercept_)
print(regr.coef_)
print(regr.score(properties_train_X, properties_train_Y))
```

[290]:
```
LinearRegression()
```

```
-66824789.4460919
[ 3.98627067e+04  3.47997852e+04  3.88420208e+03  1.33099634e+03
  5.85840307e+04  4.26388980e+00 -1.41621947e+06 -2.45191398e+06
 -9.80733859e+00 -5.44745493e+05  6.58065525e+05  9.67161343e+01
  3.75271135e+00 -6.95857123e+03 -6.55757699e+03  1.02722144e+03
  2.50903291e+04  2.09752529e+04  6.53633427e+04  3.23522344e+04
 -3.32648336e+04  3.80028359e+03 -3.74215118e+04]
0.6532255397530919
```

[291]:
```
from sklearn import linear_model

properties_train_X=properties_new.drop(["PRICE_OUTLIER"], axis=1)
properties_train_Y=properties_new["PRICE_OUTLIER"]

regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)

print(regr.intercept_)
print(regr.coef_)
print(regr.score(x_test, y_test))
```

[291]:
```
LinearRegression()
```

```
-67188593.2432264
[ 4.04371020e+04  3.56243929e+04  5.08135570e+03  1.34638158e+03
  5.78198892e+04  4.34098941e+00 -1.79680084e+06 -3.22577474e+06
 -9.56895349e+00  2.00183874e+05  1.02479591e+06  9.43300566e+01
  3.89112427e+00 -7.02509765e+03 -6.98858626e+03  1.02725782e+03
  2.53024645e+04  1.96701701e+04  6.64054123e+04  5.52214402e+03
 -6.15466329e+04 -2.35085361e+04 -6.55485150e+04]
```

0.6552550279028013

Conclusion: we can see that the variance score for linear regression is about 0.66.

for more details on correlation, we use lasso regression.

[292]:
```
from sklearn import model_selection
from sklearn.linear_model import Lasso,LassoCV
from sklearn.metrics import mean_squared_error

Lambdas=np.logspace(-5,2,200)

Lambdas=np.logspace(-5,2,200)
lasso_cv=LassoCV(alphas=Lambdas,normalize=True,cv=10,max_iter=10000)
lasso_cv.fit(x_train,y_train)

lasso=Lasso(alpha=lasso_cv.alpha_,normalize=True,max_iter=10000)
lasso.fit(x_train,y_train)
print(pd.Series(index=['Intercept']+x_train.columns.tolist(),data=[lasso.
 ↪intercept_]+lasso.coef_.tolist()))
```

[292]:
```
LassoCV(alphas=array([1.00000000e-05, 1.08436597e-05, 1.17584955e-05,
1.27505124e-05,
       1.38262217e-05, 1.49926843e-05, 1.62575567e-05, 1.76291412e-05,
       1.91164408e-05, 2.07292178e-05, 2.24780583e-05, 2.43744415e-05,
       2.64308149e-05, 2.86606762e-05, 3.10786619e-05, 3.37006433e-05,
       3.65438307e-05, 3.96268864e-05, 4.29700470e-05, 4.65952567e-05,
       5.05263107e-05, 5.47890118e-0…
       1.55222536e+01, 1.68318035e+01, 1.82518349e+01, 1.97916687e+01,
       2.14614120e+01, 2.32720248e+01, 2.52353917e+01, 2.73644000e+01,
       2.96730241e+01, 3.21764175e+01, 3.48910121e+01, 3.78346262e+01,
       4.10265811e+01, 4.44878283e+01, 4.82410870e+01, 5.23109931e+01,
       5.67242607e+01, 6.15098579e+01, 6.66991966e+01, 7.23263390e+01,
       7.84282206e+01, 8.50448934e+01, 9.22197882e+01, 1.00000000e+02]),
        cv=10, max_iter=10000, normalize=True)
```

[292]:
```
Lasso(alpha=0.020255019392306665, max_iter=10000, normalize=True)
```

```
Intercept          -6.719373e+07
BATHRM              4.043402e+04
HF_BATHRM           3.562490e+04
BEDRM               5.069428e+03
EYB                 1.346498e+03
SALE_NUM            5.781757e+04
LANDAREA            4.338672e+00
LATITUDE           -1.583752e+06
LONGITUDE          -3.015875e+06
CENSUS_TRACT       -9.554477e+00
X                  -9.591567e+03
```

```
Y                   8.115465e+05
GBA_OUTLIER         9.432834e+01
LANDAREA_OUTLIER    3.893630e+00
ZIPCODE_OUTLIER    -7.024474e+03
ROOMS_OUTLIER      -6.978377e+03
YR_RMDL_OUTLIER     1.027004e+03
GRADE_V             2.530271e+04
AC_V                1.966533e+04
CNDTN_V             6.641126e+04
QUADRANT_NE         1.568715e+04
QUADRANT_NW        -5.125978e+04
QUADRANT_SE        -1.333779e+04
QUADRANT_SW        -5.536840e+04
dtype: float64
```

[293]:
```
lasso_score=lasso.score(x_test, y_test)
lasso_score
```

[293]: 0.6552703233436432

Conclusion:

there is a slight improvement on variance score comparing with linear regression.

after normalize data and using lasso regression, we find that Latitude and Longitude have great influence on price.

Quadrant has 4 values, while NE, NW, SE have same level influence on price, SW has almost no influence.

sale_num has positive influence on price, which represents expensive houses are sold for more times.

comparing with bedroom and rooms numbers, bathroom number has greater influence on price.

grade and AC values also have great influence on price.

Y has no attribute in lasso regression, and is considered as 0.

[ ]:

[ ]:

[ ]:

[ ]:

[294]:
```
crime=pd.read_csv('DC_Crime.csv')
crime
```

[294]:
```
       NEIGHBORHOOD_CLUSTER  CENSUS_TRACT offensegroup  LONGITUDE  \
0                cluster 21        8702.0     property -77.003574
1                cluster 16        1600.0     property -77.026557
```

```
2                cluster 8        4702.0     property -77.020913
3               cluster 31        7808.0     property -76.919601
4               cluster 39       10900.0     property -77.003927
…                    …           …           …        …
449198          cluster 22       11100.0     property -76.977167
449199           cluster 6        5201.0     property -77.035546
449200          cluster 23        8802.0     property -76.982015
449201          cluster 23        8802.0     property -76.990857
449202           cluster 3        3400.0     property -77.017297

                     END_DATE         offense-text     SHIFT    YBLOCK  \
0       2017-04-29T08:00:23.000         theft f/auto       day  138139.0
1       2017-04-29T08:30:37.000         theft f/auto       day  146051.0
2       2017-04-29T11:10:57.000          theft/other       day  137185.0
3       2017-04-28T09:30:33.000          theft/other       day  135903.0
4       2017-04-29T13:42:11.000          theft/other       day  128340.0
…                           …                   …       …       …
449198  2021-03-17T20:20:07.000          theft/other   evening  139268.0
449199  2021-03-17T17:20:33.000  motor vehicle theft   evening  137976.0
449200  2021-03-17T23:09:20.000          theft/other  midnight  137150.0
449201  2021-03-17T22:40:24.000  motor vehicle theft  midnight  137569.0
449202  2021-03-25T13:00:28.000         theft f/auto       day  138891.0

        DISTRICT  WARD  …                                        BLOCK  \
0            5.0   5.0  …                150 - 299 block of q street ne
1            4.0   4.0  …       7600  - 7699 block of georgia avenue nw
2            1.0   6.0  …                600 - 699 block of k street nw
3            6.0   7.0  …          5715 5739  block of blaine street ne
4            7.0   8.0  …     4610 - 4659 block of south capitol street
…          …   …   …                                             …
449198       5.0   5.0  …         1815 - 1999 block of bryant street ne
449199       2.0   2.0  …              1500 - 1599 block of p street nw
449200       5.0   5.0  …        900 - 999 block of bladensburg road ne
449201       5.0   5.0  …  1300 - 1399 block of west virginia avenue ne
449202       3.0   1.0  …           300 - 399 block of oakdale place nw

                    START_DATE       CCN              OFFENSE  \
0       2017-04-29T01:30:14.000  17070672         theft f/auto
1       2017-04-29T02:30:10.000  17070675         theft f/auto
2       2017-04-29T10:43:33.000  17070714          theft/other
3       2017-04-28T09:15:27.000  17070736          theft/other
4       2017-04-29T13:03:40.000  17070780          theft/other
…                           …       …                   …
449198  2021-03-17T20:15:08.000  21034375          theft/other
449199  2021-03-16T21:00:41.000  21034386  motor vehicle theft
449200  2021-03-17T22:21:45.000  21034407          theft/other
449201  2021-03-17T22:38:11.000  21034425  motor vehicle theft
```

```
449202  2021-03-18T18:00:05.000  21037812         theft f/auto

        OCTO_RECORD_ID  ANC                REPORT_DAT  METHOD  \
0          17070672-01   5E  2017-04-29T13:49:31.000Z  others
1          17070675-01   4A  2017-04-29T14:38:59.000Z  others
2          17070714-01   6E  2017-04-29T15:19:02.000Z  others
3          17070736-01   7C  2017-04-29T16:11:44.000Z  others
4          17070780-01   8D  2017-04-29T18:17:15.000Z  others
…                  …  …                        …     …
449198     21034375-01   5C   2021-03-17T21:07:30.000  others
449199     21034386-01   2B   2021-03-17T21:38:51.000  others
449200     21034407-01   5D   2021-03-18T01:12:46.000  others
449201     21034425-01   5D   2021-03-18T00:57:07.000  others
449202     21037812-01   1B   2021-03-25T14:41:36.000  others

                                      location   LATITUDE
0       38.911121322949178,-77.003576581965632  38.911114
1       38.982391883146363,-77.026559339798794  38.982384
2       38.902525540064957,-77.020915170313728  38.902518
3       38.890951021927407,-76.919603310082607  38.890943
4       38.822847890448664,-77.003929146312586  38.822840
…                                          …        …
449198     38.9212817635688,-76.9771673661107  38.921282
449199     38.9096398205834,-77.0355462229667  38.909640
449200     38.9022029472732,-76.9820154636605  38.902203
449201      38.905978473251,-76.9908573768314  38.905978
449202     38.9178865750065,-77.0172966263538  38.917887

[449203 rows x 29 columns]
```

[295]:
```
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 449203 entries, 0 to 449202
Data columns (total 29 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   NEIGHBORHOOD_CLUSTER  443701 non-null  object
 1   CENSUS_TRACT          448011 non-null  float64
 2   offensegroup          449203 non-null  object
 3   LONGITUDE             449203 non-null  float64
 4   END_DATE              422407 non-null  object
 5   offense-text          449203 non-null  object
 6   SHIFT                 449203 non-null  object
 7   YBLOCK                449203 non-null  float64
 8   DISTRICT              448991 non-null  float64
 9   WARD                  449192 non-null  float64
 10  YEAR                  449203 non-null  int64
```

```
 11  offensekey            449203 non-null  object
 12  BID                   74761 non-null   object
 13  sector                448960 non-null  object
 14  PSA                   448960 non-null  float64
 15  ucr-rank              449203 non-null  int64
 16  BLOCK_GROUP           448011 non-null  object
 17  VOTING_PRECINCT       449135 non-null  object
 18  XBLOCK                449203 non-null  float64
 19  BLOCK                 449202 non-null  object
 20  START_DATE            449189 non-null  object
 21  CCN                   449203 non-null  int64
 22  OFFENSE               449203 non-null  object
 23  OCTO_RECORD_ID        449203 non-null  object
 24  ANC                   449203 non-null  object
 25  REPORT_DAT            449203 non-null  object
 26  METHOD                449203 non-null  object
 27  location              449203 non-null  object
 28  LATITUDE              449203 non-null  float64
dtypes: float64(8), int64(3), object(18)
memory usage: 99.4+ MB
```

[296]:
```
crime["YEAR"].value_counts()
```

[296]:
```
2014    38440
2015    37328
2016    37228
2013    35896
2012    35318
2008    34309
2019    33909
2018    33760
2011    33293
2017    33113
2010    31677
2009    31315
2020    27871
2021     5746
Name: YEAR, dtype: int64
```

[297]:
```
del crime['BID']
crime['START_DATE']=crime['START_DATE'].str[0:10].fillna(method='pad')
crime['END_DATE']=crime['END_DATE'].str[0:10].fillna(method='pad')
crime_filtered=crime.dropna()
crime_filtered.drop_duplicates(keep='first',inplace=True)
crime_filtered.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 442193 entries, 0 to 449202
```

```
Data columns (total 28 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   NEIGHBORHOOD_CLUSTER  442193 non-null  object
 1   CENSUS_TRACT          442193 non-null  float64
 2   offensegroup          442193 non-null  object
 3   LONGITUDE             442193 non-null  float64
 4   END_DATE              442193 non-null  object
 5   offense-text          442193 non-null  object
 6   SHIFT                 442193 non-null  object
 7   YBLOCK                442193 non-null  float64
 8   DISTRICT              442193 non-null  float64
 9   WARD                  442193 non-null  float64
 10  YEAR                  442193 non-null  int64
 11  offensekey            442193 non-null  object
 12  sector                442193 non-null  object
 13  PSA                   442193 non-null  float64
 14  ucr-rank              442193 non-null  int64
 15  BLOCK_GROUP           442193 non-null  object
 16  VOTING_PRECINCT       442193 non-null  object
 17  XBLOCK                442193 non-null  float64
 18  BLOCK                 442193 non-null  object
 19  START_DATE            442193 non-null  object
 20  CCN                   442193 non-null  int64
 21  OFFENSE               442193 non-null  object
 22  OCTO_RECORD_ID        442193 non-null  object
 23  ANC                   442193 non-null  object
 24  REPORT_DAT            442193 non-null  object
 25  METHOD                442193 non-null  object
 26  location              442193 non-null  object
 27  LATITUDE              442193 non-null  float64
dtypes: float64(8), int64(3), object(17)
memory usage: 97.8+ MB
```

divide crime data by years, while only take the data from 2008 to 2021

[298]:
```
crime_filtered_08=crime_filtered[crime_filtered.YEAR==2008]
crime_filtered_09=crime_filtered[crime_filtered.YEAR==2009]
crime_filtered_10=crime_filtered[crime_filtered.YEAR==2010]
crime_filtered_11=crime_filtered[crime_filtered.YEAR==2011]
crime_filtered_12=crime_filtered[crime_filtered.YEAR==2012]
crime_filtered_13=crime_filtered[crime_filtered.YEAR==2013]
crime_filtered_14=crime_filtered[crime_filtered.YEAR==2014]
crime_filtered_15=crime_filtered[crime_filtered.YEAR==2015]
crime_filtered_16=crime_filtered[crime_filtered.YEAR==2016]
crime_filtered_17=crime_filtered[crime_filtered.YEAR==2017]
crime_filtered_18=crime_filtered[crime_filtered.YEAR==2018]
crime_filtered_19=crime_filtered[crime_filtered.YEAR==2019]
```

```
crime_filtered_20=crime_filtered[crime_filtered.YEAR==2020]
crime_filtered_21=crime_filtered[crime_filtered.YEAR==2021]
```

create data set for model training.

[299]:
```
crime_train_08 = pd.DataFrame()
crime_train_08["LONGITUDE"] = crime_filtered_08["LONGITUDE"]
crime_train_08["LATITUDE"] = crime_filtered_08["LATITUDE"]

crime_train_09 = pd.DataFrame()
crime_train_09["LONGITUDE"] = crime_filtered_09["LONGITUDE"]
crime_train_09["LATITUDE"] = crime_filtered_09["LATITUDE"]

crime_train_10 = pd.DataFrame()
crime_train_10["LONGITUDE"] = crime_filtered_10["LONGITUDE"]
crime_train_10["LATITUDE"] = crime_filtered_10["LATITUDE"]

crime_train_11 = pd.DataFrame()
crime_train_11["LONGITUDE"] = crime_filtered_11["LONGITUDE"]
crime_train_11["LATITUDE"] = crime_filtered_11["LATITUDE"]

crime_train_12 = pd.DataFrame()
crime_train_12["LONGITUDE"] = crime_filtered_12["LONGITUDE"]
crime_train_12["LATITUDE"] = crime_filtered_12["LATITUDE"]

crime_train_13 = pd.DataFrame()
crime_train_13["LONGITUDE"] = crime_filtered_13["LONGITUDE"]
crime_train_13["LATITUDE"] = crime_filtered_13["LATITUDE"]

crime_train_14 = pd.DataFrame()
crime_train_14["LONGITUDE"] = crime_filtered_14["LONGITUDE"]
crime_train_14["LATITUDE"] = crime_filtered_14["LATITUDE"]

crime_train_15 = pd.DataFrame()
crime_train_15["LONGITUDE"] = crime_filtered_15["LONGITUDE"]
crime_train_15["LATITUDE"] = crime_filtered_15["LATITUDE"]

crime_train_16 = pd.DataFrame()
crime_train_16["LONGITUDE"] = crime_filtered_16["LONGITUDE"]
crime_train_16["LATITUDE"] = crime_filtered_16["LATITUDE"]

crime_train_17 = pd.DataFrame()
crime_train_17["LONGITUDE"] = crime_filtered_17["LONGITUDE"]
crime_train_17["LATITUDE"] = crime_filtered_17["LATITUDE"]

crime_train_18 = pd.DataFrame()
crime_train_18["LONGITUDE"] = crime_filtered_18["LONGITUDE"]
```

```
crime_train_18["LATITUDE"] = crime_filtered_18["LATITUDE"]

crime_train_19 = pd.DataFrame()
crime_train_19["LONGITUDE"] = crime_filtered_19["LONGITUDE"]
crime_train_19["LATITUDE"] = crime_filtered_19["LATITUDE"]

crime_train_20 = pd.DataFrame()
crime_train_20["LONGITUDE"] = crime_filtered_20["LONGITUDE"]
crime_train_20["LATITUDE"] = crime_filtered_20["LATITUDE"]

crime_train_21 = pd.DataFrame()
crime_train_21["LONGITUDE"] = crime_filtered_21["LONGITUDE"]
crime_train_21["LATITUDE"] = crime_filtered_21["LATITUDE"]

crime_train_08 = pd.DataFrame()
crime_train_08["LONGITUDE"] = crime_filtered_08["LONGITUDE"]
crime_train_08["LATITUDE"] = crime_filtered_08["LATITUDE"]
```

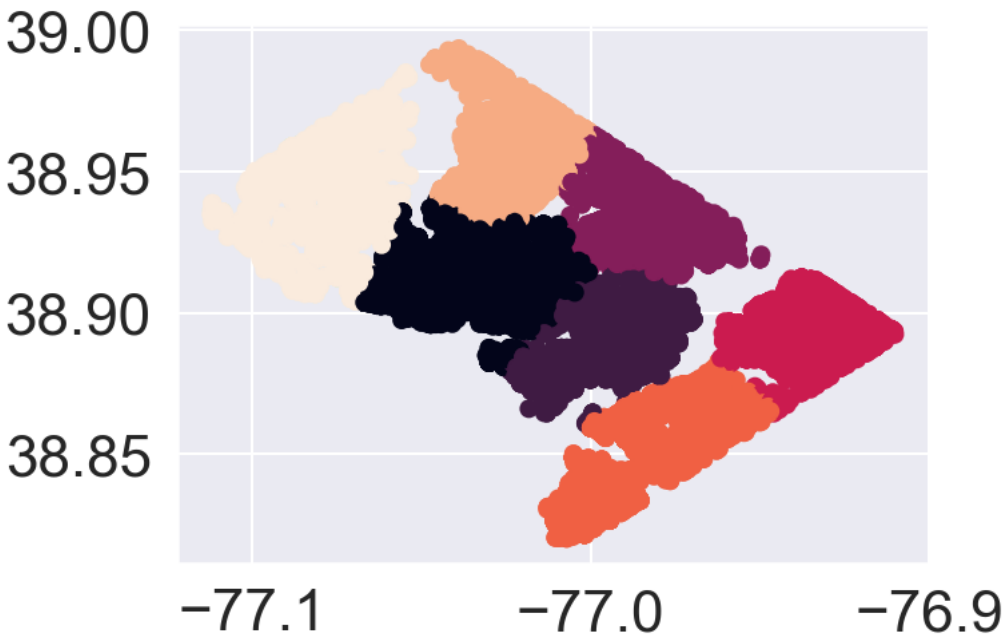
Using Gaussian Mixture clustering

[300]:
```
from sklearn.mixture import GaussianMixture


gmm_08 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_08)
y_pred = gmm_08.predict(crime_train_08)
x = crime_train_08.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_09 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_09)
y_pred = gmm_09.predict(crime_train_09)
x = crime_train_09.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_10 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_10)
y_pred = gmm_10.predict(crime_train_10)
x = crime_train_10.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_11 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_11)
y_pred = gmm_11.predict(crime_train_11)
```

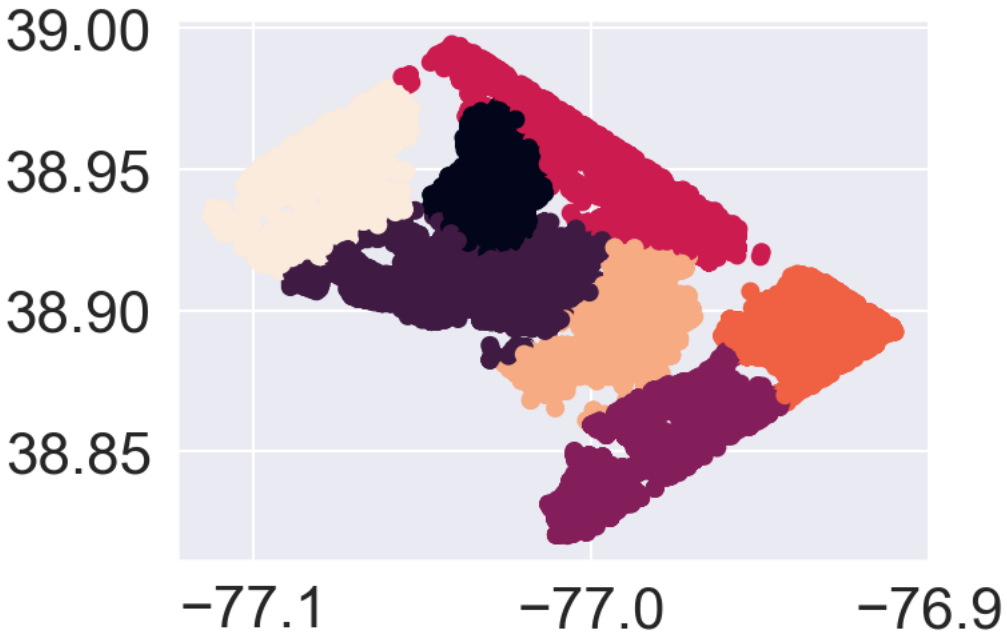
```
x = crime_train_11.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_12 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_12)
y_pred = gmm_12.predict(crime_train_12)
x = crime_train_12.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_13 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_13)
y_pred = gmm_13.predict(crime_train_13)
x = crime_train_13.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_14 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_14)
y_pred = gmm_14.predict(crime_train_14)
x = crime_train_14.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_15 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_15)
y_pred = gmm_15.predict(crime_train_15)
x = crime_train_15.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_16 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_16)
y_pred = gmm_16.predict(crime_train_16)
x = crime_train_16.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_17 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_17)
y_pred = gmm_17.predict(crime_train_17)
x = crime_train_17.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()
```

```
gmm_18 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_18)
y_pred = gmm_18.predict(crime_train_18)
x = crime_train_18.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_19 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_19)
y_pred = gmm_19.predict(crime_train_19)
x = crime_train_19.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_20 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_20)
y_pred = gmm_20.predict(crime_train_20)
x = crime_train_20.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()

gmm_21 = GaussianMixture(n_components=7, covariance_type='full').
 ↪fit(crime_train_21)
y_pred = gmm_21.predict(crime_train_21)
x = crime_train_21.values
plt.scatter(x[:, 0], x[:, 1], c=y_pred)
plt.show()
```

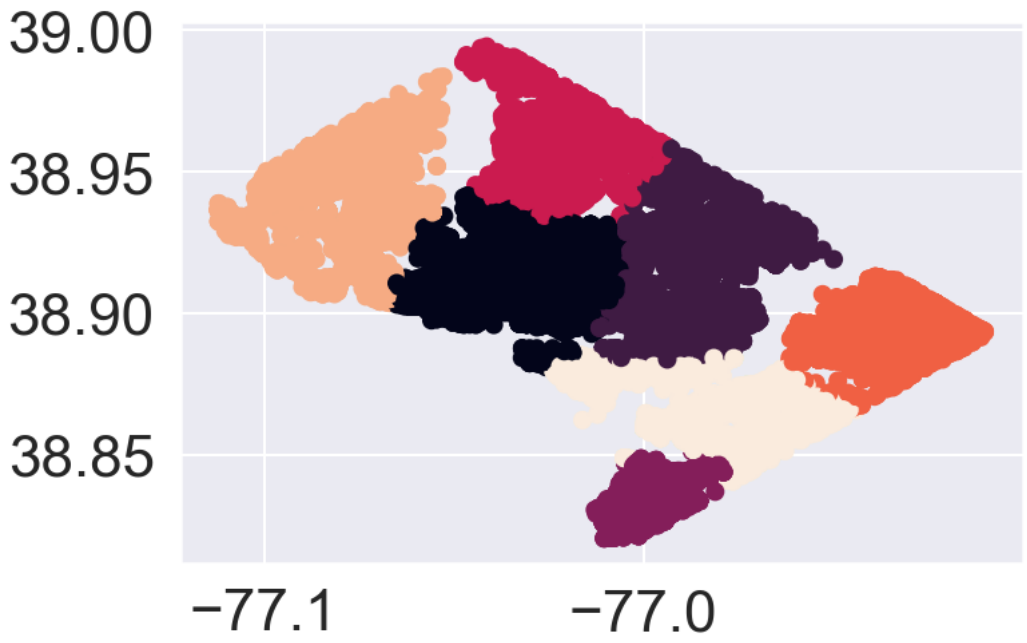
[300]: <matplotlib.collections.PathCollection at 0x2c55cf7b3d0>



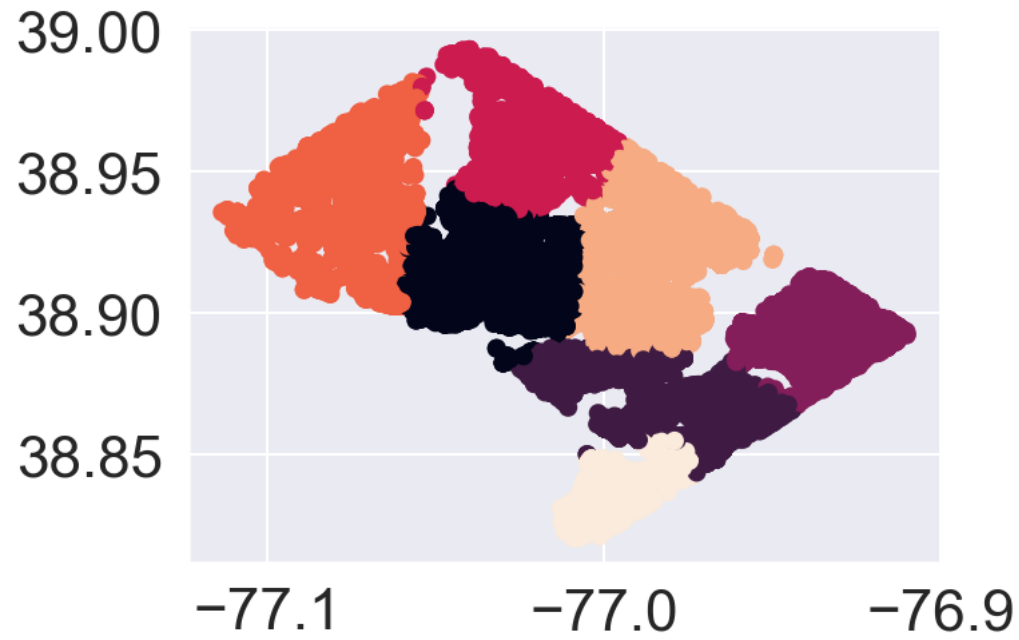
[300]: <matplotlib.collections.PathCollection at 0x2c3bfd78a90>



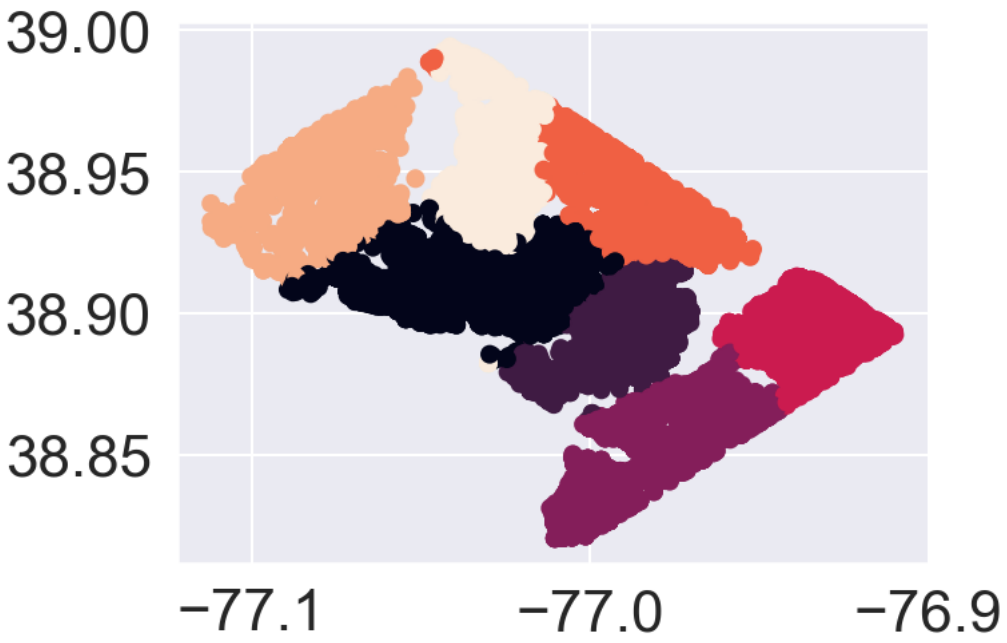
[300]: <matplotlib.collections.PathCollection at 0x2c58722eee0>



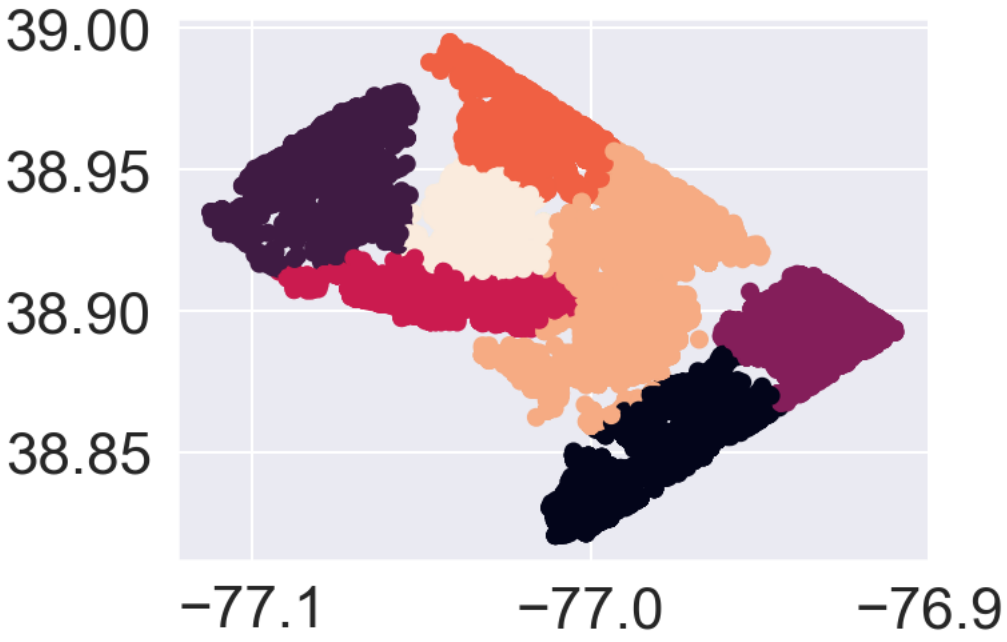
[300]: <matplotlib.collections.PathCollection at 0x2c59da88a30>



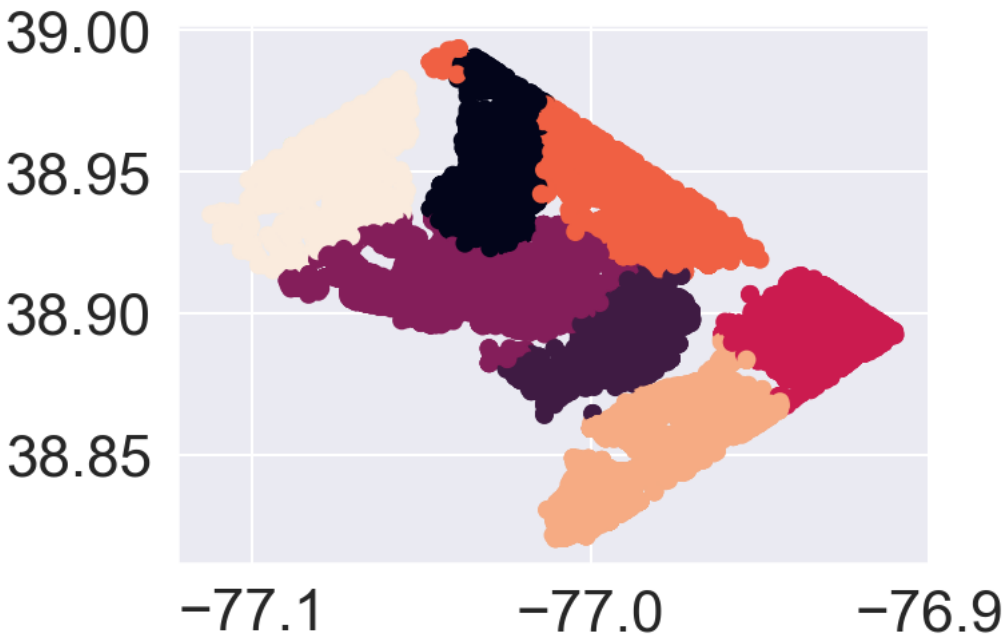
[300]: <matplotlib.collections.PathCollection at 0x2c3bfd60b20>



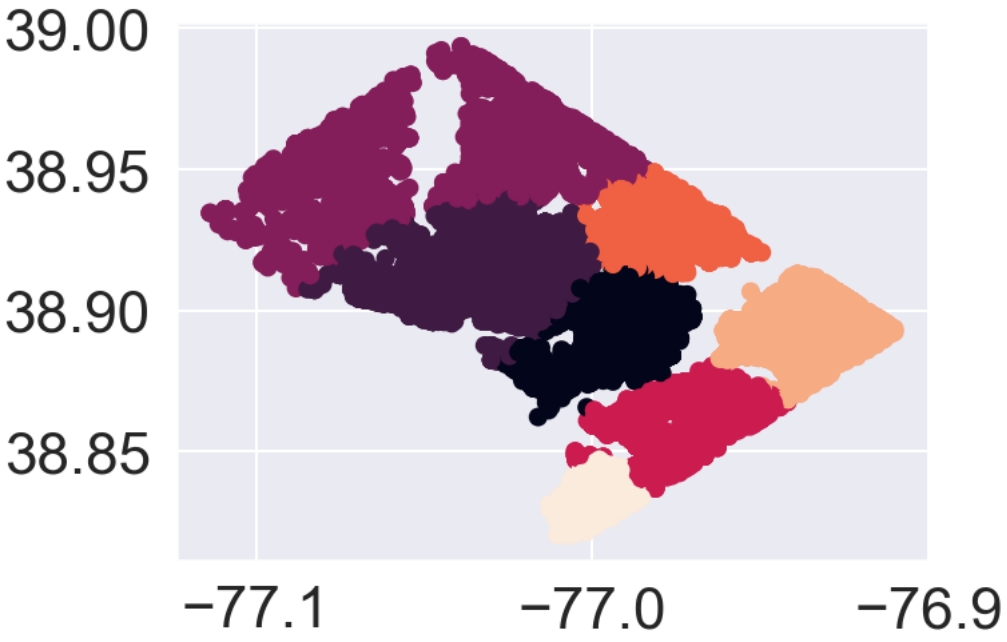
[300]: <matplotlib.collections.PathCollection at 0x2c3c0453c10>



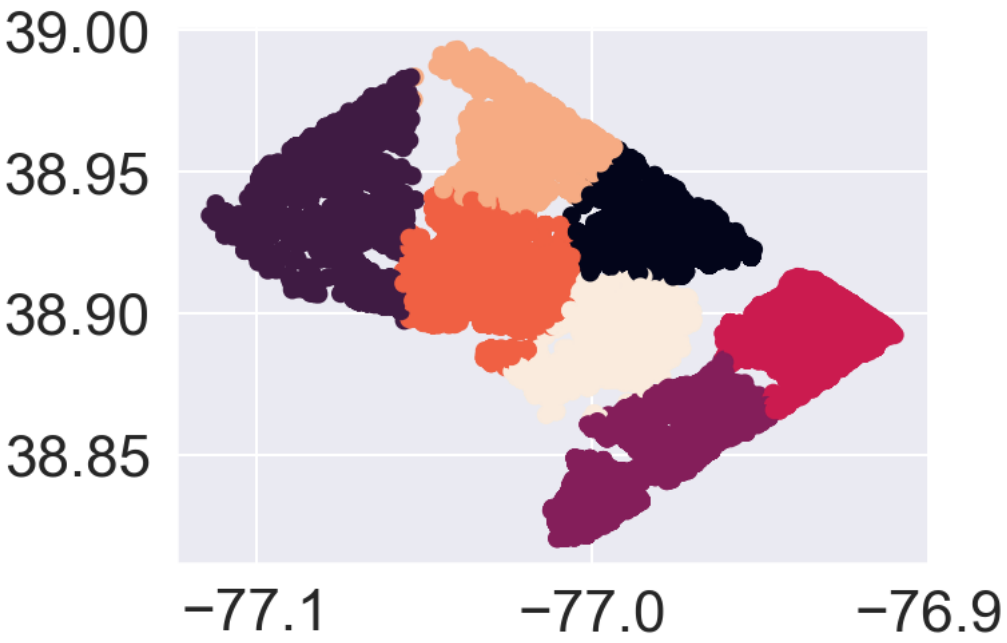
[300]: <matplotlib.collections.PathCollection at 0x2c59e33ad00>



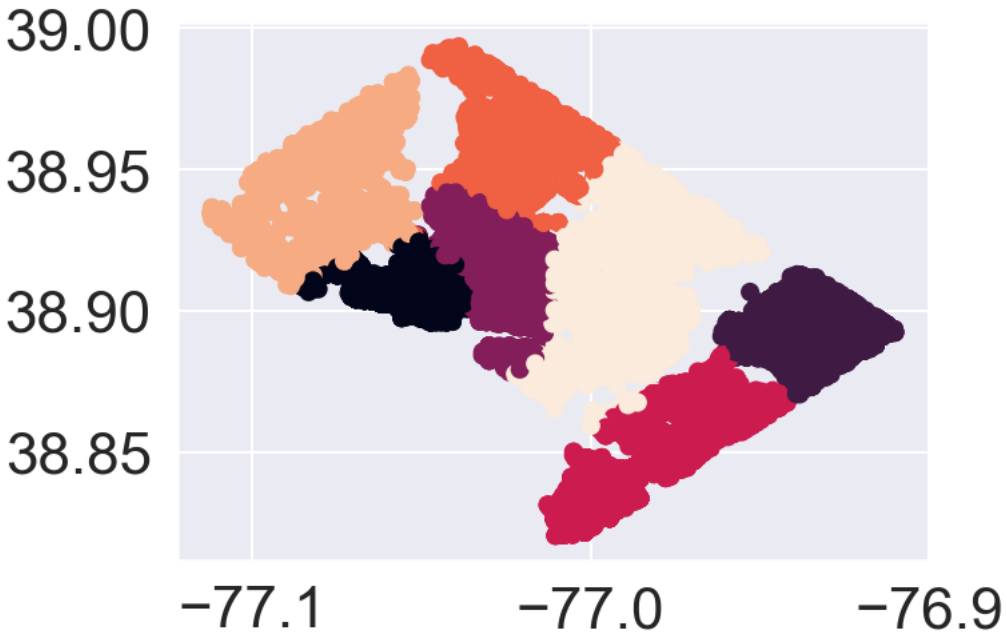
[300]: <matplotlib.collections.PathCollection at 0x2c6601f5df0>



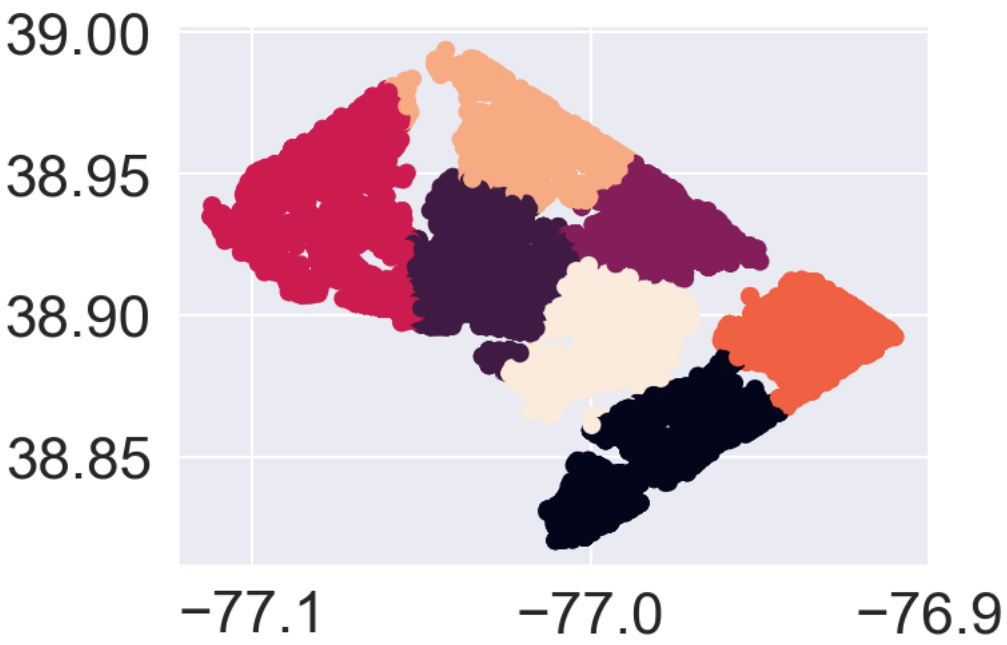
[300]: <matplotlib.collections.PathCollection at 0x2c407bf2ee0>



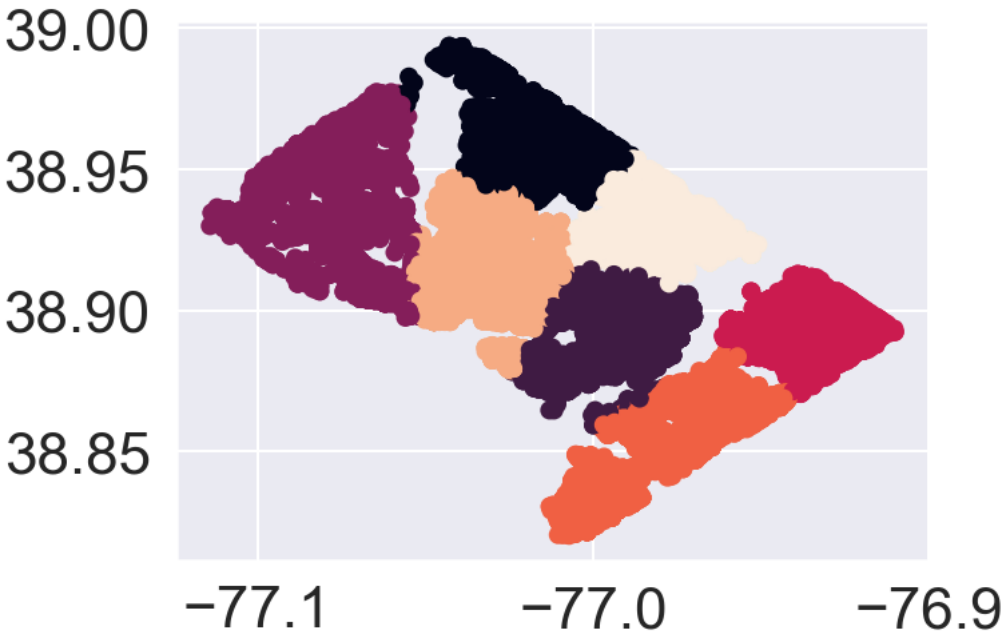
[300]: <matplotlib.collections.PathCollection at 0x2c505b53f70>



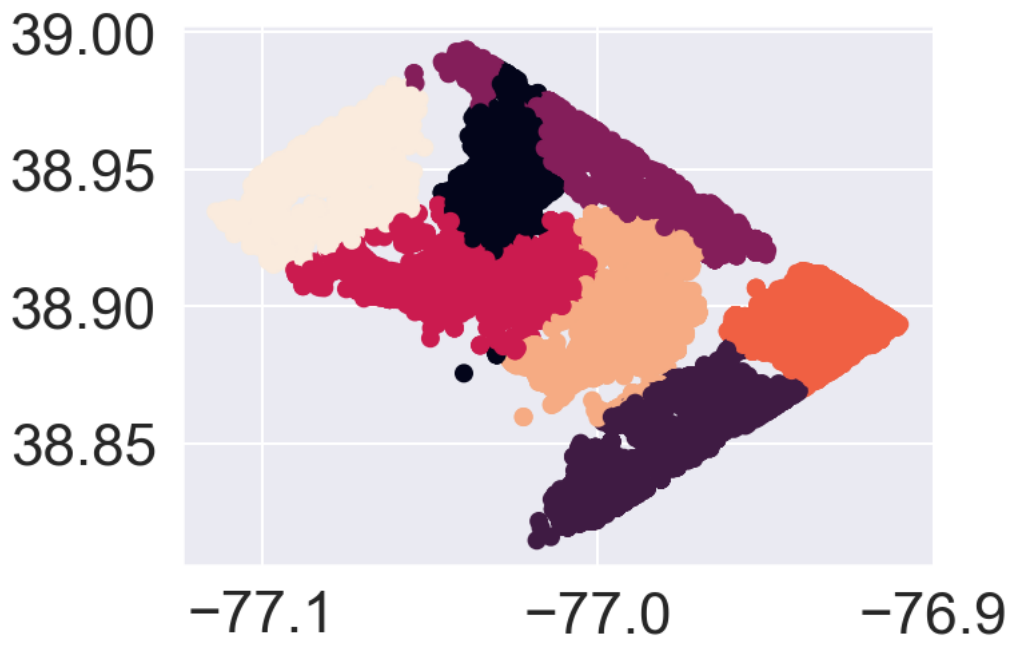
[300]: <matplotlib.collections.PathCollection at 0x2c59aed3100>



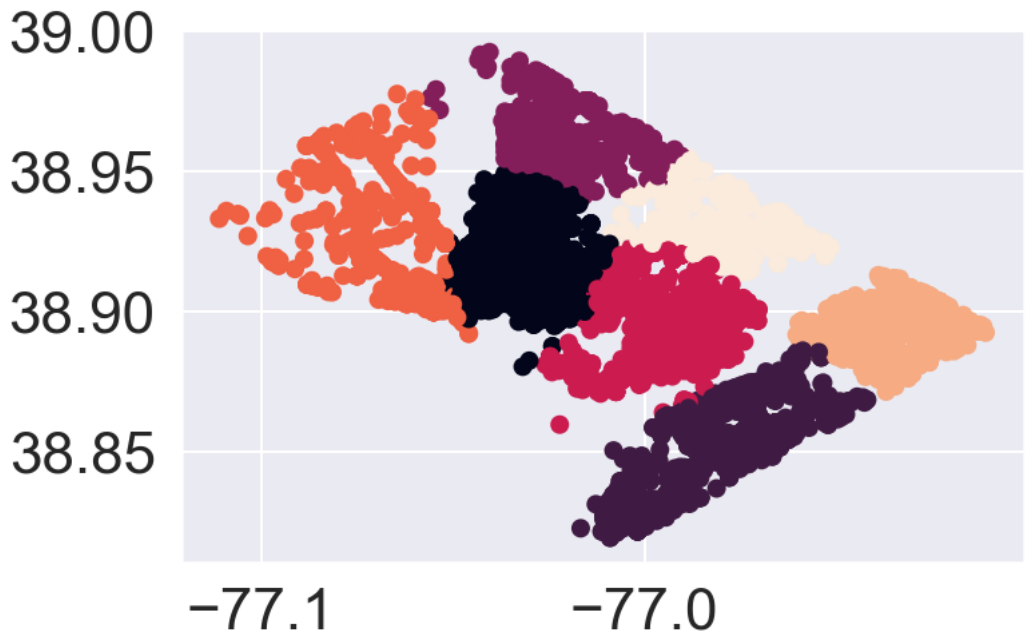
[300]: <matplotlib.collections.PathCollection at 0x2c55fc03220>



[300]: <matplotlib.collections.PathCollection at 0x2c59d82c2e0>



[300]: <matplotlib.collections.PathCollection at 0x2c65f6403d0>



Conclusion:

Gaussian Mixture can divide city into 7 distinct area, while each year is divided differently

[ ]:

[ ]:

[ ]:

[ ]:

[301]:
```
properties_new["YEAR"]=properties_new["YR_RMDL_OUTLIER"].astype(int)
properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57900 entries, 0 to 106695
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            57900 non-null  int64
 1   HF_BATHRM         57900 non-null  int64
 2   BEDRM             57900 non-null  int64
 3   EYB               57900 non-null  int64
 4   SALE_NUM          57900 non-null  int64
 5   LANDAREA          57900 non-null  int64
 6   LATITUDE          57900 non-null  float64
 7   LONGITUDE         57900 non-null  float64
 8   CENSUS_TRACT      57900 non-null  float64
 9   X                 57900 non-null  float64
 10  Y                 57900 non-null  float64
 11  PRICE_OUTLIER     57900 non-null  float64
 12  GBA_OUTLIER       57900 non-null  float64
 13  LANDAREA_OUTLIER  57900 non-null  float64
 14  ZIPCODE_OUTLIER   57900 non-null  float64
 15  ROOMS_OUTLIER     57900 non-null  float64
 16  YR_RMDL_OUTLIER   57900 non-null  float64
 17  GRADE_V           57900 non-null  int64
 18  AC_V              57900 non-null  int64
 19  CNDTN_V           57900 non-null  int64
 20  QUADRANT_NE       57900 non-null  int32
 21  QUADRANT_NW       57900 non-null  int32
 22  QUADRANT_SE       57900 non-null  int32
 23  QUADRANT_SW       57900 non-null  int32
 24  YEAR              57900 non-null  int32
dtypes: float64(11), int32(5), int64(9)
memory usage: 10.4 MB
```

For the ammount of crime is so large, we choose 20% of data as the traing set. then we divide the traing set in different years from 2008 to 2021

find the crimes that occured on the year that a house was sold which has the distance less than 0.0004

[302]:
```
p=pd.DataFrame()
def predict_crime(x,y,z) :
    p["LONGITUDE"]=[y]
    p["LATITUDE"]=[z]
    if x>=2020: return gmm_20.predict(p)
    elif x>=2019: return gmm_19.predict(p)
    elif x>=2018: return gmm_18.predict(p)
    elif x>=2017: return gmm_17.predict(p)
    elif x>=2016: return gmm_16.predict(p)
    elif x>=2015: return gmm_15.predict(p)
    elif x>=2014: return gmm_14.predict(p)
    elif x>=2013: return gmm_13.predict(p)
    elif x>=2012: return gmm_12.predict(p)
    elif x>=2011: return gmm_11.predict(p)
    elif x>=2010: return gmm_10.predict(p)
    elif x>=2009: return gmm_09.predict(p)
    elif x>=2008: return gmm_08.predict(p)
    else: return gmm_08.predict(p)

properties_new["CRIME"] = properties_new.apply(lambda a:predict_crime(a.YEAR, a.
 ↪LONGITUDE, a.LATITUDE),axis=1)

properties_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57900 entries, 0 to 106695
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            57900 non-null  int64
 1   HF_BATHRM         57900 non-null  int64
 2   BEDRM             57900 non-null  int64
 3   EYB               57900 non-null  int64
 4   SALE_NUM          57900 non-null  int64
 5   LANDAREA          57900 non-null  int64
 6   LATITUDE          57900 non-null  float64
 7   LONGITUDE         57900 non-null  float64
 8   CENSUS_TRACT      57900 non-null  float64
 9   X                 57900 non-null  float64
 10  Y                 57900 non-null  float64
 11  PRICE_OUTLIER     57900 non-null  float64
 12  GBA_OUTLIER       57900 non-null  float64
 13  LANDAREA_OUTLIER  57900 non-null  float64
 14  ZIPCODE_OUTLIER   57900 non-null  float64
 15  ROOMS_OUTLIER     57900 non-null  float64
 16  YR_RMDL_OUTLIER   57900 non-null  float64
 17  GRADE_V           57900 non-null  int64
 18  AC_V              57900 non-null  int64
```

```
 19  CNDTN_V           57900 non-null  int64
 20  QUADRANT_NE       57900 non-null  int32
 21  QUADRANT_NW       57900 non-null  int32
 22  QUADRANT_SE       57900 non-null  int32
 23  QUADRANT_SW       57900 non-null  int32
 24  YEAR              57900 non-null  int32
 25  CRIME             57900 non-null  object
dtypes: float64(11), int32(5), int64(9), object(1)
memory usage: 10.8+ MB
```

[303]:
```
properties_new["CRIME"].describe()
```

[303]:
```
count     57900
unique        7
top         [1]
freq      10734
Name: CRIME, dtype: object
```

[304]:
```
properties_combine_crime=properties_new[properties_new['YEAR']>2007]

from sklearn import model_selection
x_train,x_test,y_train,y_test=model_selection.
 ↪train_test_split(crime_filtered,crime_filtered["OFFENSE"],test_size=0.
 ↪8,random_state=1234)

train_filtered_08=x_train[x_train.YEAR==2008]
train_filtered_09=x_train[x_train.YEAR==2009]
train_filtered_10=x_train[x_train.YEAR==2010]
train_filtered_11=x_train[x_train.YEAR==2011]
train_filtered_12=x_train[x_train.YEAR==2012]
train_filtered_13=x_train[x_train.YEAR==2013]
train_filtered_14=x_train[x_train.YEAR==2014]
train_filtered_15=x_train[x_train.YEAR==2015]
train_filtered_16=x_train[x_train.YEAR==2016]
train_filtered_17=x_train[x_train.YEAR==2017]
train_filtered_18=x_train[x_train.YEAR==2018]
train_filtered_19=x_train[x_train.YEAR==2019]
train_filtered_20=x_train[x_train.YEAR==2020]
train_filtered_21=x_train[x_train.YEAR==2021]

idx=properties_combine_crime.index


properties_combine_crime['CRIME_NUM']=0
for i in idx:
    count=0
    x=properties_combine_crime.loc[i,'LONGITUDE']
```

```
    y=properties_combine_crime.loc[i,'LATITUDE']

    if properties_combine_crime.loc[i,'YEAR']==2008:
        for j in train_filtered_08.index:
            if (((train_filtered_08.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_08.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2009:
        for j in train_filtered_09.index:
            if (((train_filtered_09.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_09.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2010:
        for j in train_filtered_10.index:
            if (((train_filtered_10.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_10.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2011:
        for j in train_filtered_11.index:
            if (((train_filtered_11.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_11.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2012:
        for j in train_filtered_12.index:
            if (((train_filtered_12.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_12.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2013:
        for j in train_filtered_13.index:
            if (((train_filtered_13.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_13.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2014:
        for j in train_filtered_14.index:
            if (((train_filtered_14.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_14.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2015:
        for j in train_filtered_15.index:
```

```
            if (((train_filtered_15.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_15.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2016:
        for j in train_filtered_16.index:
            if (((train_filtered_16.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_16.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2017:
        for j in train_filtered_17.index:
            if (((train_filtered_17.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_17.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2018:
        for j in train_filtered_18.index:
            if (((train_filtered_18.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_18.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2019:
        for j in train_filtered_19.index:
            if (((train_filtered_19.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_19.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2020:
        for j in train_filtered_20.index:
            if (((train_filtered_20.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_20.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1
    elif properties_combine_crime.loc[i,'YEAR']==2021:
        for j in train_filtered_21.index:
            if (((train_filtered_21.
↪loc[j,'LONGITUDE']-x)**2)+((train_filtered_21.loc[j,'LATITUDE']-y)**2))<0.
↪00004:
                count=count+1

    properties_combine_crime.loc[i,'CRIME_NUM']=count
#     print(i,count,properties_combine_crime.loc[i,'CRIME_NUM'])

properties_combine_crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 16759 entries, 2 to 106695
Data columns (total 27 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   BATHRM            16759 non-null  int64
 1   HF_BATHRM         16759 non-null  int64
 2   BEDRM             16759 non-null  int64
 3   EYB               16759 non-null  int64
 4   SALE_NUM          16759 non-null  int64
 5   LANDAREA          16759 non-null  int64
 6   LATITUDE          16759 non-null  float64
 7   LONGITUDE         16759 non-null  float64
 8   CENSUS_TRACT      16759 non-null  float64
 9   X                 16759 non-null  float64
 10  Y                 16759 non-null  float64
 11  PRICE_OUTLIER     16759 non-null  float64
 12  GBA_OUTLIER       16759 non-null  float64
 13  LANDAREA_OUTLIER  16759 non-null  float64
 14  ZIPCODE_OUTLIER   16759 non-null  float64
 15  ROOMS_OUTLIER     16759 non-null  float64
 16  YR_RMDL_OUTLIER   16759 non-null  float64
 17  GRADE_V           16759 non-null  int64
 18  AC_V              16759 non-null  int64
 19  CNDTN_V           16759 non-null  int64
 20  QUADRANT_NE       16759 non-null  int32
 21  QUADRANT_NW       16759 non-null  int32
 22  QUADRANT_SE       16759 non-null  int32
 23  QUADRANT_SW       16759 non-null  int32
 24  YEAR              16759 non-null  int32
 25  CRIME             16759 non-null  object
 26  CRIME_NUM         16759 non-null  int64
dtypes: float64(11), int32(5), int64(10), object(1)
memory usage: 3.8+ MB
```

display the correlation matrix

[305]:
```
corrmatrix=properties_combine_crime.corr()
corrmatrix
```

[305]:

	BATHRM	HF_BATHRM	BEDRM	EYB	SALE_NUM	LANDAREA	\
BATHRM	1.000000	0.105255	0.657180	0.339263	0.004942	0.317109	
HF_BATHRM	0.105255	1.000000	0.145300	0.260042	-0.013244	0.143485	
BEDRM	0.657180	0.145300	1.000000	0.235421	-0.046807	0.330765	
EYB	0.339263	0.260042	0.235421	1.000000	0.041374	0.185574	
SALE_NUM	0.004942	-0.013244	-0.046807	0.041374	1.000000	-0.085424	
LANDAREA	0.317109	0.143485	0.330765	0.185574	-0.085424	1.000000	
LATITUDE	0.200254	0.173567	0.194610	0.092427	-0.064770	0.245112	
LONGITUDE	-0.319078	-0.240859	-0.279423	-0.327817	0.115887	-0.236953	

```
CENSUS_TRACT     -0.277473  -0.200899 -0.239459 -0.303818  0.092489 -0.206894
X                -0.319183  -0.240385 -0.279484 -0.327215  0.115641 -0.237507
Y                 0.199752   0.173357  0.194168  0.093191 -0.064610  0.241155
PRICE_OUTLIER     0.475724   0.308100  0.373632  0.460038  0.188393  0.202271
GBA_OUTLIER       0.635465   0.255845  0.663093  0.323698 -0.140694  0.427163
LANDAREA_OUTLIER  0.300006   0.089161  0.350703  0.124605 -0.096427  0.764207
ZIPCODE_OUTLIER  -0.064331  -0.133773 -0.000207 -0.046260  0.030948  0.253117
ROOMS_OUTLIER     0.615132   0.183692  0.652261  0.250170 -0.043900  0.285615
YR_RMDL_OUTLIER   0.042008   0.013430  0.010179  0.040801  0.308470  0.011246
GRADE_V           0.183039   0.148209  0.190859  0.248424 -0.084112  0.142715
AC_V              0.139365   0.138201  0.012821  0.223361  0.102117  0.028636
CNDTN_V           0.217655   0.144446  0.081658  0.282310  0.239372 -0.006544
QUADRANT_NE      -0.147861  -0.080043 -0.148629 -0.213047  0.060396 -0.132976
QUADRANT_NW       0.275290   0.188219  0.258794  0.269199 -0.088584  0.172065
QUADRANT_SE      -0.171749  -0.137450 -0.149907 -0.088385  0.038351 -0.067878
QUADRANT_SW      -0.029239  -0.038419 -0.029269 -0.010334  0.016304 -0.013759
YEAR              0.042008   0.013430  0.010179  0.040801  0.308470  0.011246
CRIME_NUM        -0.037024  -0.010922 -0.036771 -0.003310  0.014751 -0.387068

                  LATITUDE  LONGITUDE  CENSUS_TRACT         X  …  \
BATHRM            0.200254  -0.319078     -0.277473 -0.319183  …
HF_BATHRM         0.173567  -0.240859     -0.200899 -0.240385  …
BEDRM             0.194610  -0.279423     -0.239459 -0.279484  …
EYB               0.092427  -0.327817     -0.303818 -0.327215  …
SALE_NUM         -0.064770   0.115887      0.092489  0.115641  …
LANDAREA          0.245112  -0.236953     -0.206894 -0.237507  …
LATITUDE          1.000000  -0.547527     -0.584037 -0.547096  …
LONGITUDE        -0.547527   1.000000      0.801430  0.998726  …
CENSUS_TRACT     -0.584037   0.801430      1.000000  0.799520  …
X                -0.547096   0.998726      0.799520  1.000000  …
Y                 0.997492  -0.547516     -0.584917 -0.547972  …
PRICE_OUTLIER     0.270629  -0.601644     -0.484508 -0.600927  …
GBA_OUTLIER       0.251141  -0.479121     -0.390895 -0.478930  …
LANDAREA_OUTLIER  0.327490  -0.240055     -0.215016 -0.241252  …
ZIPCODE_OUTLIER  -0.138921   0.160304      0.149762  0.159052  …
ROOMS_OUTLIER     0.168822  -0.290320     -0.249082 -0.290179  …
YR_RMDL_OUTLIER   0.001601   0.088580      0.052361  0.088885  …
GRADE_V           0.208447  -0.496372     -0.415582 -0.496293  …
AC_V             -0.028485  -0.014819      0.005670 -0.015134  …
CNDTN_V          -0.016870  -0.002566      0.053694 -0.002613  …
QUADRANT_NE      -0.119171   0.502308      0.665542  0.503745  …
QUADRANT_NW       0.660322  -0.795973     -0.909070 -0.795674  …
QUADRANT_SE      -0.662502   0.423842      0.349592  0.424572  …
QUADRANT_SW      -0.199718  -0.004873      0.101801 -0.004565  …
YEAR              0.001601   0.088580      0.052361  0.088885  …
CRIME_NUM        -0.181523   0.047185      0.042781  0.048110  …
```

	YR_RMDL_OUTLIER	GRADE_V	AC_V	CNDTN_V	QUADRANT_NE	\
BATHRM	0.042008	0.183039	0.139365	0.217655	-0.147861	
HF_BATHRM	0.013430	0.148209	0.138201	0.144446	-0.080043	
BEDRM	0.010179	0.190859	0.012821	0.081658	-0.148629	
EYB	0.040801	0.248424	0.223361	0.282310	-0.213047	
SALE_NUM	0.308470	-0.084112	0.102117	0.239372	0.060396	
LANDAREA	0.011246	0.142715	0.028636	-0.006544	-0.132976	
LATITUDE	0.001601	0.208447	-0.028485	-0.016870	-0.119171	
LONGITUDE	0.088580	-0.496372	-0.014819	-0.002566	0.502308	
CENSUS_TRACT	0.052361	-0.415582	0.005670	0.053694	0.665542	
X	0.088885	-0.496293	-0.015134	-0.002613	0.503745	
Y	0.001663	0.209221	-0.027342	-0.016433	-0.118180	
PRICE_OUTLIER	0.059714	0.453274	0.177642	0.294163	-0.250550	
GBA_OUTLIER	-0.070247	0.374995	0.013414	0.012942	-0.251750	
LANDAREA_OUTLIER	0.048494	0.186587	0.014088	-0.063943	-0.148384	
ZIPCODE_OUTLIER	0.161929	-0.110072	-0.002553	-0.058960	-0.140253	
ROOMS_OUTLIER	0.006669	0.192209	0.019293	0.123403	-0.163659	
YR_RMDL_OUTLIER	1.000000	-0.060427	0.067316	0.224233	0.001909	
GRADE_V	-0.060427	1.000000	0.037324	-0.063695	-0.244178	
AC_V	0.067316	0.037324	1.000000	0.283280	0.016993	
CNDTN_V	0.224233	-0.063695	0.283280	1.000000	0.083505	
QUADRANT_NE	0.001909	-0.244178	0.016993	0.083505	1.000000	
QUADRANT_NW	-0.055633	0.354919	-0.019239	-0.056425	-0.656927	
QUADRANT_SE	0.056789	-0.152769	0.003980	-0.033050	-0.321423	
QUADRANT_SW	0.052676	-0.050264	0.008254	0.029722	-0.059971	
YEAR	1.000000	-0.060427	0.067316	0.224233	0.001909	
CRIME_NUM	-0.065528	-0.047802	-0.033245	0.015623	0.008178	

	QUADRANT_NW	QUADRANT_SE	QUADRANT_SW	YEAR	CRIME_NUM
BATHRM	0.275290	-0.171749	-0.029239	0.042008	-0.037024
HF_BATHRM	0.188219	-0.137450	-0.038419	0.013430	-0.010922
BEDRM	0.258794	-0.149907	-0.029269	0.010179	-0.036771
EYB	0.269199	-0.088385	-0.010334	0.040801	-0.003310
SALE_NUM	-0.088584	0.038351	0.016304	0.308470	0.014751
LANDAREA	0.172065	-0.067878	-0.013759	0.011246	-0.387068
LATITUDE	0.660322	-0.662502	-0.199718	0.001601	-0.181523
LONGITUDE	-0.795973	0.423842	-0.004873	0.088580	0.047185
CENSUS_TRACT	-0.909070	0.349592	0.101801	0.052361	0.042781
X	-0.795674	0.424572	-0.004565	0.088885	0.048110
Y	0.664340	-0.662876	-0.199898	0.001663	-0.179257
PRICE_OUTLIER	0.442997	-0.254356	-0.063637	0.059714	0.064520
GBA_OUTLIER	0.377598	-0.176728	-0.044850	-0.070247	-0.054840
LANDAREA_OUTLIER	0.187072	-0.069952	-0.009967	0.048494	-0.514323
ZIPCODE_OUTLIER	-0.116656	0.260770	0.219031	0.161929	-0.388950
ROOMS_OUTLIER	0.243871	-0.116607	-0.015096	0.006669	-0.039447
YR_RMDL_OUTLIER	-0.055633	0.056789	0.052676	1.000000	-0.065528
GRADE_V	0.354919	-0.152769	-0.050264	-0.060427	-0.047802

```
AC_V               -0.019239     0.003980     0.008254  0.067316  -0.033245
CNDTN_V            -0.056425    -0.033050     0.029722  0.224233   0.015623
QUADRANT_NE        -0.656927    -0.321423    -0.059971  0.001909   0.008178
QUADRANT_NW         1.000000    -0.476774    -0.088956 -0.055633   0.081317
QUADRANT_SE        -0.476774     1.000000    -0.043525  0.056789  -0.096744
QUADRANT_SW        -0.088956    -0.043525     1.000000  0.052676  -0.049811
YEAR               -0.055633     0.056789     0.052676  1.000000  -0.065528
CRIME_NUM           0.081317    -0.096744    -0.049811 -0.065528   1.000000

[26 rows x 26 columns]
```

create the quadratic, cube, and biquardrate number of CRIME_NUM

show the crrelation between those values and price

[306]:
```
properties_combine_crime["CRIME_NUM_2"]=properties_combine_crime["CRIME_NUM"]*properties_combi
properties_combine_crime["CRIME_NUM_3"]=properties_combine_crime["CRIME_NUM_2"]*properties_com
properties_combine_crime["CRIME_NUM_4"]=properties_combine_crime["CRIME_NUM_2"]*properties_com
corrmatrix=properties_combine_crime.corr()
corrmatrix["PRICE_OUTLIER"].sort_values()
```

[306]:
```
LONGITUDE          -0.601644
X                  -0.600927
CENSUS_TRACT       -0.484508
ZIPCODE_OUTLIER    -0.298018
QUADRANT_SE        -0.254356
QUADRANT_NE        -0.250550
QUADRANT_SW        -0.063637
YEAR                0.059714
YR_RMDL_OUTLIER     0.059714
CRIME_NUM           0.064520
CRIME_NUM_4         0.116446
CRIME_NUM_2         0.126739
CRIME_NUM_3         0.130076
LANDAREA_OUTLIER    0.160243
AC_V                0.177642
SALE_NUM            0.188393
LANDAREA            0.202271
LATITUDE            0.270629
Y                   0.270928
CNDTN_V             0.294163
HF_BATHRM           0.308100
BEDRM               0.373632
ROOMS_OUTLIER       0.374510
QUADRANT_NW         0.442997
GRADE_V             0.453274
EYB                 0.460038
BATHRM              0.475724
```

```
GBA_OUTLIER          0.533065
PRICE_OUTLIER        1.000000
Name: PRICE_OUTLIER, dtype: float64
```

we can see that the cube number has the greatest correlation, so we keep the cube number and drop others

[307]:
```
properties_combine_crime.drop(["CRIME_NUM"],axis=1,inplace=True)
properties_combine_crime.drop(["CRIME_NUM_2"],axis=1,inplace=True)
properties_combine_crime.drop(["CRIME_NUM_4"],axis=1,inplace=True)
properties_combine_crime.drop(["CRIME"],axis=1,inplace=True)
properties_combine_crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16759 entries, 2 to 106695
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   BATHRM             16759 non-null  int64
 1   HF_BATHRM          16759 non-null  int64
 2   BEDRM              16759 non-null  int64
 3   EYB                16759 non-null  int64
 4   SALE_NUM           16759 non-null  int64
 5   LANDAREA           16759 non-null  int64
 6   LATITUDE           16759 non-null  float64
 7   LONGITUDE          16759 non-null  float64
 8   CENSUS_TRACT       16759 non-null  float64
 9   X                  16759 non-null  float64
 10  Y                  16759 non-null  float64
 11  PRICE_OUTLIER      16759 non-null  float64
 12  GBA_OUTLIER        16759 non-null  float64
 13  LANDAREA_OUTLIER   16759 non-null  float64
 14  ZIPCODE_OUTLIER    16759 non-null  float64
 15  ROOMS_OUTLIER      16759 non-null  float64
 16  YR_RMDL_OUTLIER    16759 non-null  float64
 17  GRADE_V            16759 non-null  int64
 18  AC_V               16759 non-null  int64
 19  CNDTN_V            16759 non-null  int64
 20  QUADRANT_NE        16759 non-null  int32
 21  QUADRANT_NW        16759 non-null  int32
 22  QUADRANT_SE        16759 non-null  int32
 23  QUADRANT_SW        16759 non-null  int32
 24  YEAR               16759 non-null  int32
 25  CRIME_NUM_3        16759 non-null  int64
dtypes: float64(11), int32(5), int64(10)
memory usage: 3.6 MB
```

using linear regression model and lasso regression model

[308]:
```
from sklearn import linear_model
from sklearn import model_selection

x_train,x_test,y_train,y_test=model_selection.
 ↪train_test_split(properties_combine_crime,properties_combine_crime["PRICE_OUTLIER"],test_si
 ↪2,random_state=1234)

x_train.drop(["PRICE_OUTLIER"],axis=1, inplace=True)
x_test.drop(["PRICE_OUTLIER"],axis=1, inplace=True)

regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)

print(regr.intercept_)
print(regr.coef_)
print(regr.score(x_test, y_test))
```

[308]:
```
LinearRegression()

-44659384.64634759
[ 3.68208855e+04  3.53765225e+04  3.78054740e+03  3.81842234e+03
  4.26611118e+04  1.30323453e+00  5.20830887e+05 -1.62407913e+06
 -9.21565646e+00 -1.56151504e+06 -1.37197400e+06  1.02711838e+02
  8.59682085e+00 -9.21154169e+03 -5.57368326e+03  2.23205411e+03
  3.57966136e+04  2.33832724e+04  8.39680514e+04  7.47133752e+04
  3.20064605e+03  6.02906563e+04 -4.63369330e+03  2.23205411e+03
  4.29532685e-03]
0.6687359029601967
```

[309]:
```
from sklearn.linear_model import Lasso,LassoCV
from sklearn.metrics import mean_squared_error

Lambdas=np.logspace(-5,2,200)

Lambdas=np.logspace(-5,2,200)
lasso_cv=LassoCV(alphas=Lambdas,normalize=True,cv=10,max_iter=10000)
lasso_cv.fit(x_train,y_train)

lasso=Lasso(alpha=lasso_cv.alpha_,normalize=True,max_iter=10000)
lasso.fit(x_train,y_train)
print(pd.Series(index=['Intercept']+x_train.columns.tolist(),data=[lasso.
 ↪intercept_]+lasso.coef_.tolist()))
lasso_score=lasso.score(x_test, y_test)
lasso_score
```

[309]:
```
LassoCV(alphas=array([1.00000000e-05, 1.08436597e-05, 1.17584955e-05,
1.27505124e-05,
       1.38262217e-05, 1.49926843e-05, 1.62575567e-05, 1.76291412e-05,
```

```
       1.91164408e-05, 2.07292178e-05, 2.24780583e-05, 2.43744415e-05,
       2.64308149e-05, 2.86606762e-05, 3.10786619e-05, 3.37006433e-05,
       3.65438307e-05, 3.96268864e-05, 4.29700470e-05, 4.65952567e-05,
       5.05263107e-05, 5.47890118e-0…
       1.55222536e+01, 1.68318035e+01, 1.82518349e+01, 1.97916687e+01,
       2.14614120e+01, 2.32720248e+01, 2.52353917e+01, 2.73644000e+01,
       2.96730241e+01, 3.21764175e+01, 3.48910121e+01, 3.78346262e+01,
       4.10265811e+01, 4.44878283e+01, 4.82410870e+01, 5.23109931e+01,
       5.67242607e+01, 6.15098579e+01, 6.66991966e+01, 7.23263390e+01,
       7.84282206e+01, 8.50448934e+01, 9.22197882e+01, 1.00000000e+02]),
        cv=10, max_iter=10000, normalize=True)
```

[309]:
```
Lasso(alpha=0.9884959046625587, max_iter=10000, normalize=True)
```

```
Intercept          -4.416448e+07
BATHRM              3.675177e+04
HF_BATHRM           3.521461e+04
BEDRM               3.553360e+03
EYB                 3.822070e+03
SALE_NUM            4.260446e+04
LANDAREA            1.319206e+00
LATITUDE           -5.081469e+05
LONGITUDE          -2.564423e+06
CENSUS_TRACT       -8.798583e+00
X                  -6.114806e+05
Y                  -3.363122e+05
GBA_OUTLIER         1.026295e+02
LANDAREA_OUTLIER    8.426506e+00
ZIPCODE_OUTLIER    -9.205330e+03
ROOMS_OUTLIER      -5.323340e+03
YR_RMDL_OUTLIER     4.364467e+03
GRADE_V             3.585427e+04
AC_V                2.332007e+04
CNDTN_V             8.384831e+04
QUADRANT_NE         1.388643e+04
QUADRANT_NW        -5.408271e+04
QUADRANT_SE         0.000000e+00
QUADRANT_SW        -6.377325e+04
YEAR                5.814500e+01
CRIME_NUM_3         4.210469e-03
dtype: float64
```

[309]:
```
0.6687505643296618
```

Conclusion:

the cube of crime num occured nearby has a positive attribute to the price.

6 5 Extra data analysis(Task 5)

@task:Hereby we give you some suggestions. For instance, you may predict the number of crimes in the future (or for certain types of crimes). You can divide different prediction models according to the space-time grid. Either standard regression or other methods based on time series analysis can be used. You can also find the correlation between crime situation and local economic status/housing price (find datasets by yourself). We encourage you do this task by using PCA to reduce dimensionality or boosting method to reduce bias, etc.

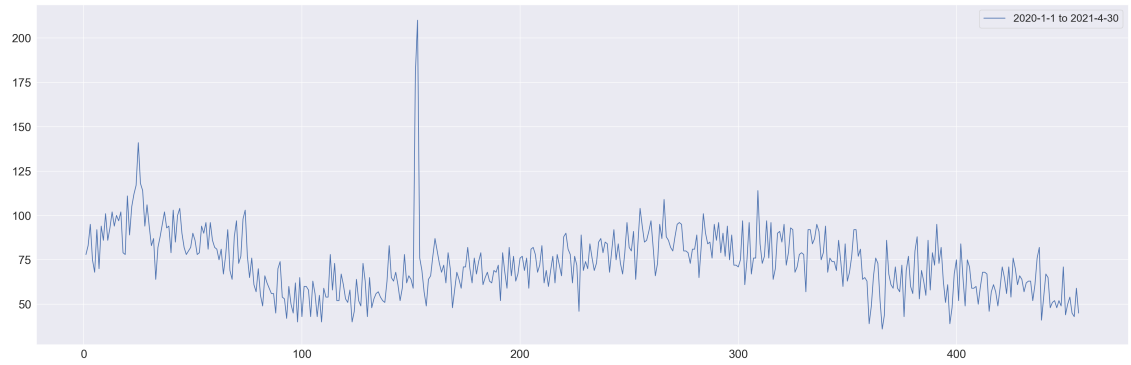
6.0.1 5-1 Predict the number of crimes in the future

Here we view the latter months of 2021 as the future since the crime data for 2021 is clearly incomplete, which can be inferred from our previous analysis.

[310]:
```
# time: 2020-1-1 to 2021-4-30
# num_20: number of crimes from 2020-1-1 to 2020-12-31
# num_21: number of crimes from 2021-1-1 to 2021-4-30
time=[]
num_20=[]
num_21=[]
for i in range(2020,2022):
    if i == 2020:
        for j in range(1,367):
            temp = crime_model[crime_model.YEAR==i][crime_model.dayofyear==j].
 ↪shape[0]
            if temp != 0:
                time.append([i,j])
                num_20.append(temp)
    if i == 2021:
        for j in range(1,121):
            temp = crime_model[crime_model.YEAR==i][crime_model.dayofyear==j].
 ↪shape[0]
            if temp != 0:
                time.append([i,366+j])
                num_21.append(temp)
num = num_20+num_21
# num: number of crimes from 2020-1-1 to 2021-4-30
```

[311]:
```
plt.figure(figsize=(30,10))
plt.plot(range(1,len(num)+1),num)
plt.legend(['2020-1-1 to 2021-4-30'],loc='upper right',fontsize=20)
plt.show()
# number of crimes     2020-1-1  to  2021-4-30
```

[311]:



[312]:
```
# future: from 2021-5-1 to 2021-6-1
future = []
for i in range(121,153):
    future.append([2021,366+i])
```

Neural network

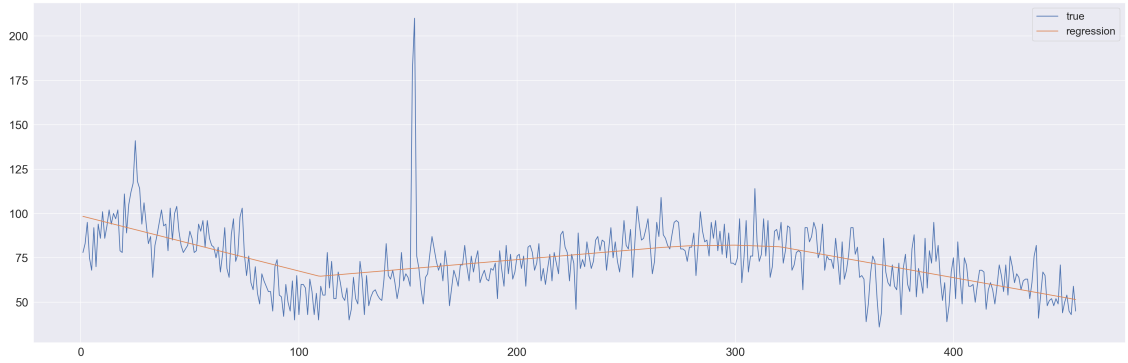
[313]:
```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor

X_train, X_test, y_train, y_test = train_test_split(time, num ,test_size = 1/3.
 ↪,random_state = 8)
regr = MLPRegressor(hidden_layer_sizes=(100,50),random_state=1,max_iter=500).
 ↪fit(X_train, y_train)
regr.score(X_train, y_train)
```

[313]: 0.36909583662334966

[314]:
```
num_p = regr.predict(time)
num_f = regr.predict(future)
num_p = num_p.tolist()
num_f = num_f.tolist()
```

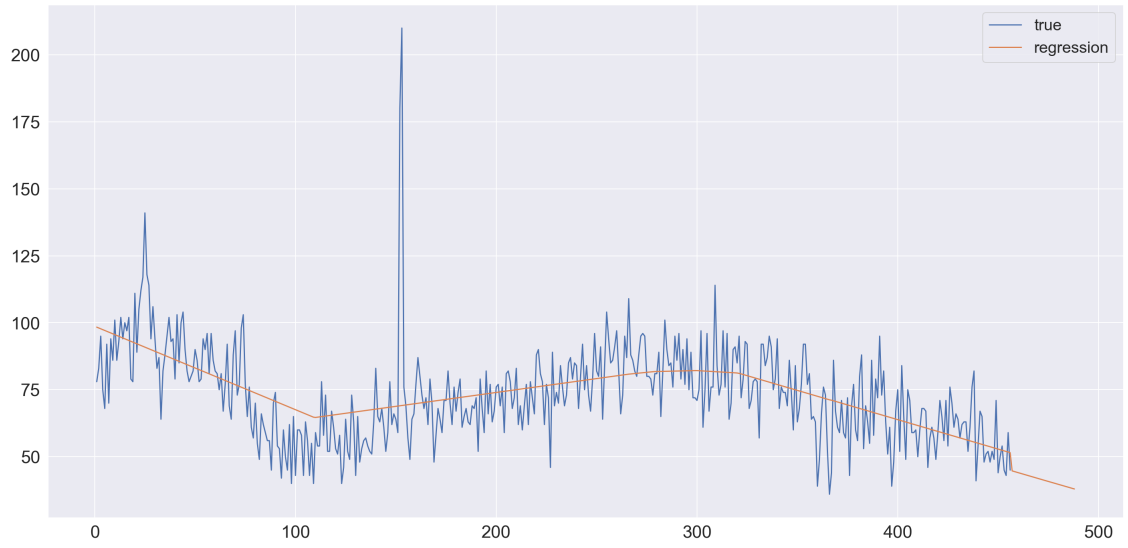
[315]:
```
plt.figure(figsize=(30,10))
plt.plot(range(1,len(num)+1),num)
plt.plot(range(1,len(num_p)+1),num_p)
plt.legend(['true','regression'],loc='upper right',fontsize=20)
plt.show()
# visulazition       2020-1-1    to    2020-4-30
```

[315]:



[316]:
```
num_f = num_p + num_f
plt.figure(figsize=(20,10))
plt.plot(range(1,len(num)+1),num)
plt.plot(range(1,len(num_f)+1),num_f)
plt.legend(['true','regression'],loc='upper right',fontsize=20)
plt.show()
# visulazition       2021-5-1   to   2021-6-1
```

[316]:



*Conclusion: the number of crimes would decrease in May 2021. Prediction works well in the short future, but for a long future, the prediction is probably not very accurate.

linear model

[317]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression(normalize=True,n_jobs=2)
scores =␣
 ↪cross_val_score(lr,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
print('meas square error: ',scores.mean())
```

```
meas square error:  -290.59461705474814
```

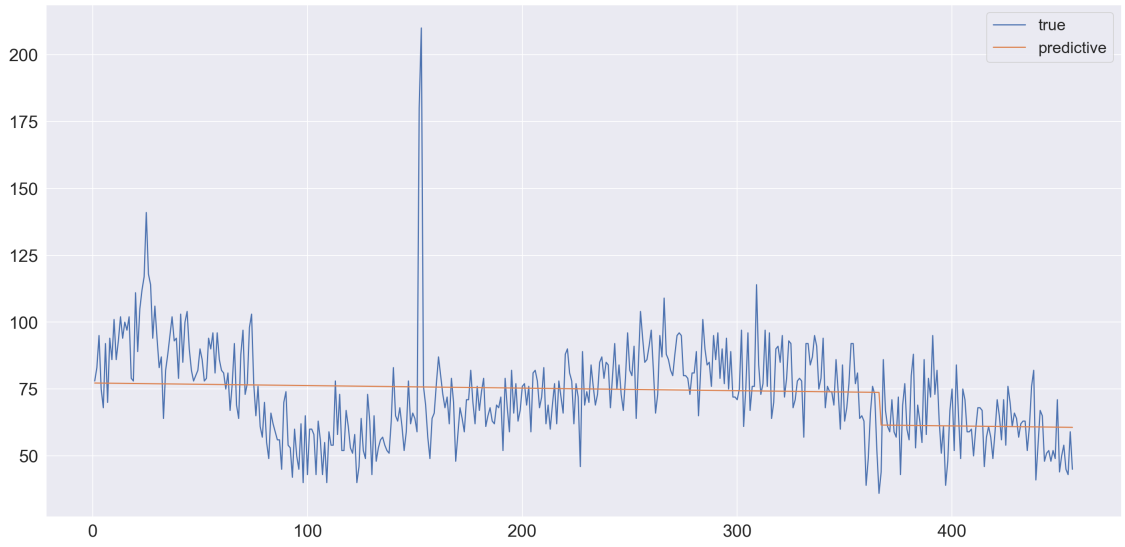
[318]:
```
lr.fit(X_train,y_train)
lr.score(X_test,y_test)
```

[318]:
```
LinearRegression(n_jobs=2, normalize=True)
```

[318]:
```
0.08431180990119491
```

[319]:
```
num_p = lr.predict(time)
num_f = lr.predict(future)
num_p = num_p.tolist()
num_f = num_f.tolist()
plt.figure(figsize=(20,10))
plt.plot(range(1,len(num)+1),num)
plt.plot(range(1,len(num_p)+1),num_p)
plt.legend(['true','predictive'],loc='upper right',fontsize=20)
plt.show()
```

[319]:



*Conclusion: linear model has bad performance. The regression curve is almost flat and does not fit well.

6.0.2 5-2 Find the correlation between crime situation and bank locations

Dataset about the location of banks in DC. https://opendata.dc.gov/datasets/DCGIS::atm-banking/about

[320]:
```
bank = pd.read_csv('ATM_banking.csv')
bank
```

[320]:
```
              X          Y  OBJECTID                      NAME  \
0   -77.021733  38.915743         1                Wells Fargo
1   -77.030663  38.917218         2         Bank of Georgetown
2   -77.021716  38.915189         3            Bank of America
3   -77.019156  38.924101         4            Bank of America
4   -77.020789  38.927280         5            Bank of America
..         …        …       …                            …
271 -77.026692  38.906798       272  ATM Services Corporation
272 -77.025529  38.941036       273  ATM Services Corporation
273 -77.021623  38.920616       274  ATM Services Corporation
274 -76.951080  38.894844       275  ATM Services Corporation
275 -76.974377  38.898309       276  ATM Services Corporation

                     ADDRESS   ID  ZIPCODE  WARD
0         1901 7TH STREET NW    1    20001     1
1           1301 U STREET NW    2    20009     1
2         1845 7TH STREET NW    3    20001     1
3         2397 6TH STREET NW    4    20059     1
4       511 GRESHAM PLACE NW    5    20059     1
..                       …  …       …   …
271     1231 11th Street NW   161    20001     2
272     4100 Georgia Ave NW   156    20011     4
273  2301 f georgia ave nw    159    20001     1
274    3701 Benning Road NE   157    20019     7
275  2033 BENNING ROAD, NE    144    20002     2

[276 rows x 8 columns]
```

[321]:
```
bank.isnull().sum()
```

[321]:
```
X           0
Y           0
OBJECTID    0
NAME        0
ADDRESS     0
ID          0
ZIPCODE     0
WARD        0
dtype: int64
```

PCA——dimension reduction.

[322]:
```
from sklearn.decomposition import PCA
```

[323]:
```
# all feature in crime_model exclude 'DATE'
features = ['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD', 'YEAR',
       'ucr-rank', 'VOTING_PRECINCT', 'LATITUDE', 'OFFENSE_code',
       'SPAN', 'TIME_TO_REPORT', 'MONTH', 'DAY', 'hour', 'dayofyear',
       'week', 'weekofyear', 'dayofweek', 'weekday', 'quarter',
       'OFFENSE_arson', 'OFFENSE_assault w/dangerous weapon',
       'OFFENSE_burglary', 'OFFENSE_homicide', 'OFFENSE_motor vehicle theft',
       'OFFENSE_robbery', 'OFFENSE_sex abuse', 'OFFENSE_theft f/auto',
       'OFFENSE_theft/other', 'offensegroup_property', 'offensegroup_violent',
       'SHIFT_day', 'SHIFT_evening', 'SHIFT_midnight', 'METHOD_gun',
       'METHOD_knife', 'METHOD_others']
```

[324]:
```
# PCA
x = crime_model[features]
pca = PCA()
pca.fit(crime_model[features])
```

[324]:
```
PCA()
```

[325]:
```
pca.fit(x)
n_component = 0
while sum(pca.explained_variance_ratio_[0: n_component]) < 0.9:
    n_component += 1
pca = PCA(n_components=n_component)
x = pca.fit_transform(x)
```

[325]:
```
PCA()
```

Cluster the data after dimension reduction

[326]:
```
km = KMeans(n_clusters=5).fit(x)
crime_KMeans=crime_model.copy()
crime_KMeans['cluster'] = km.labels_
crime_KMeans.groupby('cluster').mean()
```

[326]:

cluster	CENSUS_TRACT	LONGITUDE	DISTRICT	WARD	YEAR	ucr-rank	\
0	6196.777792	-77.008252	3.716259	4.414206	2014.095172	5.949528	
1	6271.076923	-77.001484	3.846154	4.692308	2018.692308	3.307692	
2	5021.047619	-77.009392	3.476190	4.523810	2014.523810	5.857143	
3	6296.400369	-77.011810	3.455720	4.337638	2012.839483	5.887454	
4	5668.779070	-77.017219	3.627907	3.825581	2014.976744	3.220930	

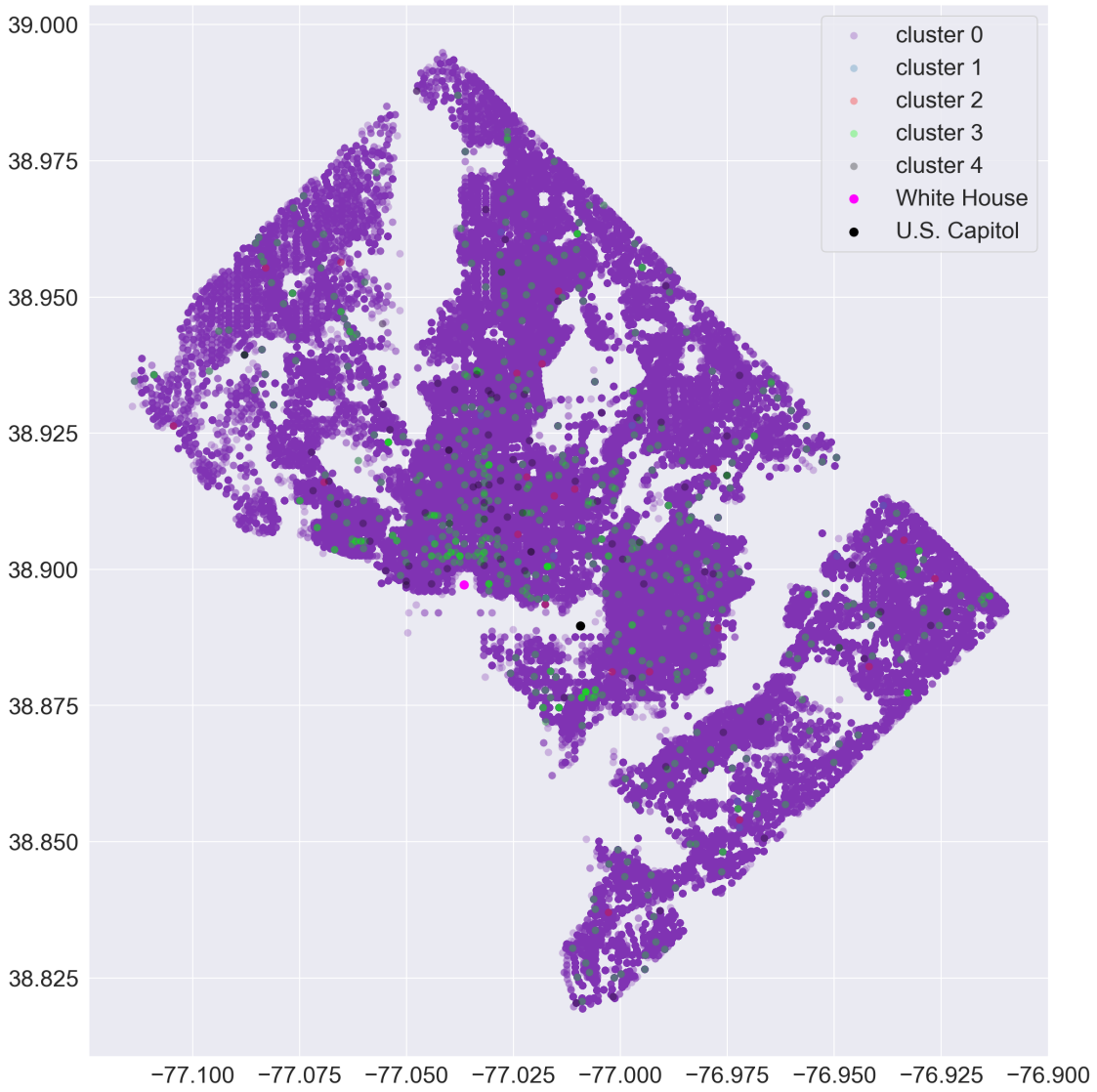
cluster	VOTING_PRECINCT	LATITUDE	SPAN	TIME_TO_REPORT	…	\
0	70.127699	38.906895	7.602266e+04	1.181744e+05	…	

```
1              75.230769  38.908752 -1.107821e+08    2.977522e+08  …
2              73.714286  38.908187  2.294879e+08   -9.176335e+07  …
3              70.271218  38.907893  3.865934e+07   -3.123582e+06  …
4              59.325581  38.909512 -3.150786e+06    9.303001e+07  …

         OFFENSE_theft f/auto  OFFENSE_theft/other  offensegroup_property  \
cluster
0                    0.266594             0.391038               0.831999
1                    0.000000             0.153846               0.307692
2                    0.428571             0.333333               0.857143
3                    0.138376             0.627306               0.898524
4                    0.104651             0.186047               0.302326

         offensegroup_violent  SHIFT_day  SHIFT_evening  SHIFT_midnight  \
cluster
0                    0.168001   0.377861       0.427655        0.194484
1                    0.692308   0.153846       0.461538        0.384615
2                    0.142857   0.380952       0.428571        0.190476
3                    0.101476   0.520295       0.369004        0.110701
4                    0.697674   0.267442       0.395349        0.337209

         METHOD_gun  METHOD_knife  METHOD_others
cluster
0          0.059655      0.030038       0.910306
1          0.153846      0.000000       0.846154
2          0.000000      0.000000       1.000000
3          0.029520      0.007380       0.963100
4          0.069767      0.023256       0.906977

[5 rows x 36 columns]
```

[327]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==0],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==0], s=50, alpha=0.3,␣
 ↪color=[0.5,0.2,0.7], lw=0, label='cluster 0')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==1],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==1], s=50, alpha=0.3,␣
 ↪color=[0.2,0.5,0.7], lw=0, label='cluster 1')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==2],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==2], s=50, alpha=0.3,␣
 ↪color=[1,0,0], lw=0, label='cluster 2')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==3],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==3], s=50, alpha=0.3,␣
 ↪color=[0,1,0], lw=0, label='cluster 3')
```

```
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==4],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==4], s=50, alpha=0.3,␣
 ↪color=[0,0,0], lw=0, label='cluster 4')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
```

[327]:



The vast majority of crimes belong to one cluser that is purple, which we presume might represent minor, less harmful crimes, such as theft.
Hide this cluster and let's have a clearer look at the other types.

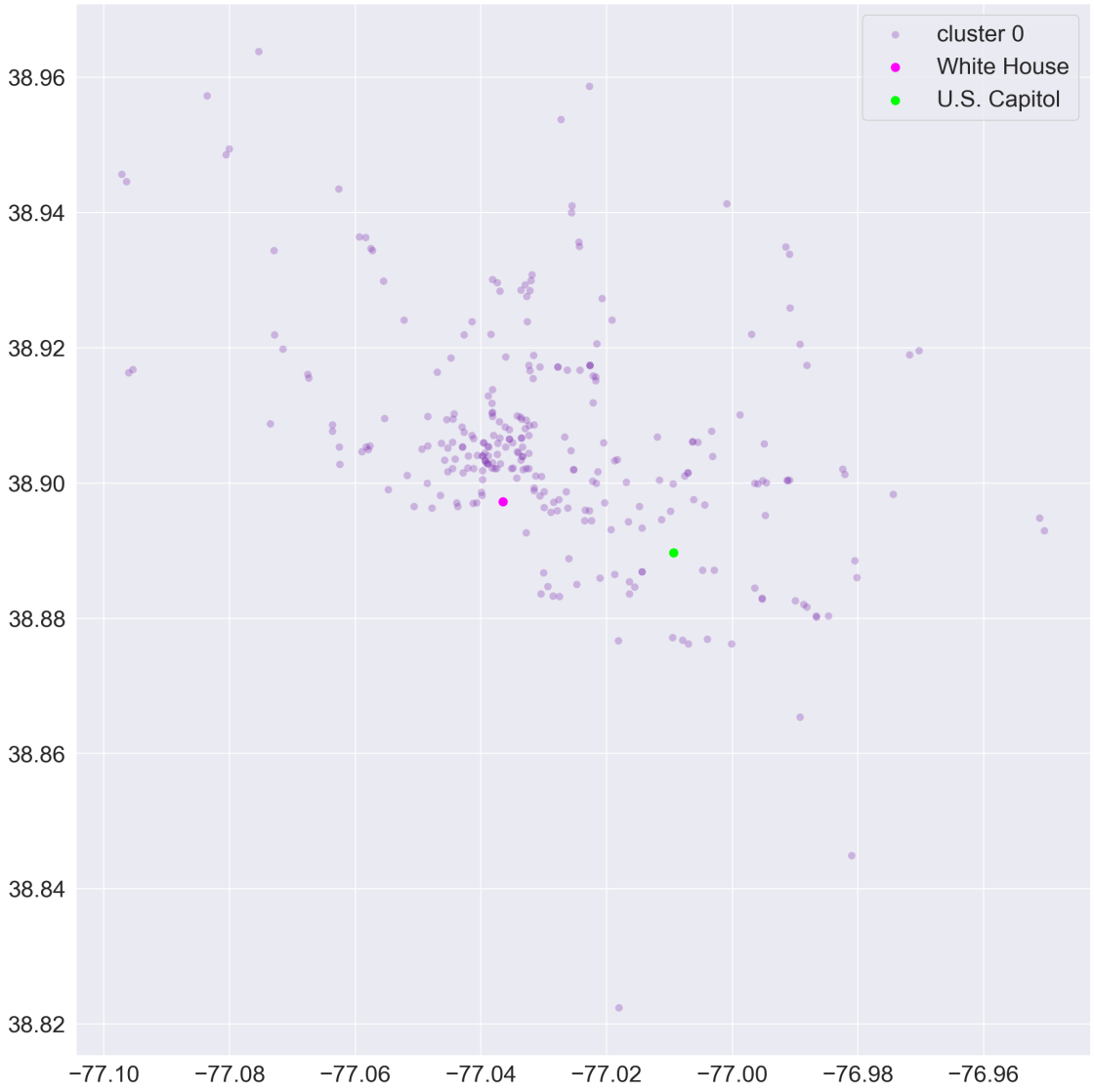
[328]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
#plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==0],␣
↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==0], s=50, alpha=0.3,␣
↪color=[0.5,0.2,0.7], lw=0, label='cluster 0')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==1],␣
↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==1], s=50, alpha=0.3,␣
↪color=[1,1,1], lw=0, label='cluster 1')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==2],␣
↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==2], s=50, alpha=0.3,␣
↪color=[1,0,0], lw=0, label='cluster 2')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==3],␣
↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==3], s=50, alpha=0.3,␣
↪color=[0,1,0], lw=0, label='cluster 3')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==4],␣
↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==4], s=50, alpha=0.3,␣
↪color=[0,0,0], lw=0, label='cluster 4')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,0,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
```

[328]:

To find the correlation between crimes and banks, let's plot the map for the location of banks.

[329]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(bank['X'],bank['Y'], s=50, alpha=0.3, color=[0.5,0.2,0.7], lw=0,␣
↪label='cluster 0')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,1,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
plt.show()
```

[329]:



We infer that the distribution of cluster that is black may have some correlation with the distribution of bank locations. We plot them in one map.

[330]:
```
plt.figure(figsize=(15, 15))
sns.set(font_scale=2)
plt.scatter(bank['X'],bank['Y'], s=50, alpha=0.3, color=[0.5,0.2,0.7], lw=0,␣
 ↪label='cluster 0')
plt.scatter(crime_KMeans['LONGITUDE'][crime_KMeans['cluster']==4],␣
 ↪crime_KMeans['LATITUDE'][crime_KMeans['cluster']==4], s=50, alpha=0.3,␣
 ↪color=[0,0,0], lw=0, label='cluster 1')
plt.scatter(-77.03654,38.89722,s=60, color=[1,0,1], lw=1, label='White House')
plt.scatter(-77.00937,38.88968,s=60, color=[0,1,0], lw=1, label='U.S. Capitol')
plt.legend(loc='upper right')
```

```
plt.show()
```

[330]:

*Conclusion: crimes of cluster that is black may have some correlation with bank. In particular, inner-city crime, almost all of which takes place in the location of the bank, and the more densely the bank, the more crime.

6.0.3 5-3 Some interesting classifications

Here we do two kinds of classification. One is binary classification which views 'offensegroup' as label. The other is a classification that views 'OFFENSE' as label.

offensegroup as label binary classification)

[331]:
```
crime_model['offensegroup']=crime_filtered_newfeature['offensegroup']
variables = ['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD',
             'YEAR', 'sector','ucr-rank', 'VOTING_PRECINCT',
             'ANC', 'LATITUDE', 'SPAN', 'TIME_TO_REPORT',
             'MONTH', 'DAY', 'hour', 'dayofyear','week',
             'weekofyear', 'dayofweek', 'weekday', 'quarter',
             'OFFENSE_arson', 'OFFENSE_assault w/dangerous weapon',
             'OFFENSE_burglary', 'OFFENSE_homicide',
             'OFFENSE_motor vehicle theft','OFFENSE_robbery',
             'OFFENSE_sex abuse', 'OFFENSE_theft f/auto',
             'OFFENSE_theft/other','SHIFT_day', 'SHIFT_evening',
             'SHIFT_midnight', 'METHOD_gun','METHOD_knife',
             'METHOD_others']
x=crime_model[variables]
y=crime_model['offensegroup']
```

[332]:
```
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[333]:
```
dtree =␣
↪DecisionTreeClassifier(criterion='entropy',max_depth=7,min_samples_leaf=11,min_samples_spli
```

[334]:
```
dtree.fit(x_train,y_train)
```

[334]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=11,
                       min_samples_split=10)
```

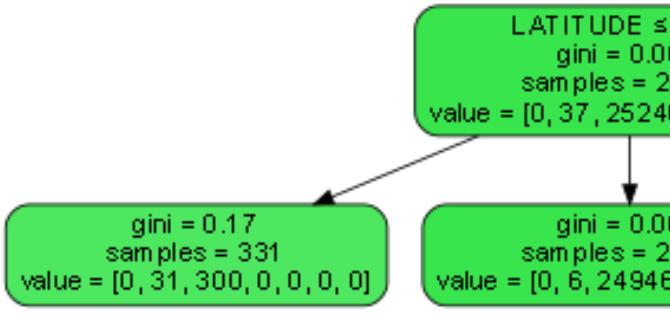
[335]:
```
y_pred = dtree.predict(x_test)
```

[336]:
```
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[91505     0]
 [    0 18554]]
              precision    recall  f1-score   support

    property       1.00      1.00      1.00     91505
     violent       1.00      1.00      1.00     18554

    accuracy                           1.00    110059
   macro avg       1.00      1.00      1.00    110059
weighted avg       1.00      1.00      1.00    110059
```

[337]:
```
tree.export_graphviz(dtree, out_file = dot_data, feature_names =␣
↪crime_model[variables].columns,
                     filled = True, rounded = True, special_characters = True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_png("decisiontree_offensegroup_01.png")
Image(graph[0].create_png())
```

[337]:


LATITUDE ≤
gini = 0.0
samples = 2
value = [0, 37, 2524
gini = 0.17
samples = 331
value = [0, 31, 300, 0, 0, 0, 0]
gini = 0.0
samples = 2
value = [0, 6, 24946

Clearly, we can completely derive offensegroup from ucr-rank. What if we delete ucr-rank and other nature information of crimes?

[338]:
```
variables = ['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT',
             'WARD', 'YEAR', 'sector','VOTING_PRECINCT',
             'ANC', 'LATITUDE', 'SPAN', 'TIME_TO_REPORT',
             'MONTH', 'DAY', 'hour', 'dayofyear','week',
             'weekofyear', 'dayofweek', 'weekday', 'quarter']
x=crime_model[variables]
y=crime_model['offensegroup']
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[339]:
```
dtree =␣
 ↪DecisionTreeClassifier(criterion='entropy',max_depth=7,min_samples_leaf=11,min_samples_spli
dtree.fit(x_train,y_train)
y_pred = dtree.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

[339]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=11,
                       min_samples_split=10)
```

```
[[87744  3761]
 [12811  5743]]
              precision    recall  f1-score   support

    property       0.87      0.96      0.91     91505
     violent       0.60      0.31      0.41     18554

    accuracy                           0.85    110059
   macro avg       0.74      0.63      0.66    110059
weighted avg       0.83      0.85      0.83    110059
```

[340]:
```
dot_data = StringIO()
tree.export_graphviz(dtree, out_file = dot_data, feature_names =␣
 ↪crime_model[variables].columns,
                     filled = True, rounded = True, special_characters = True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_png("decisiontree_offensegroup_02.png")
```

```
Image(graph[0].create_png())
```

[340]:

OFFENSE as label

[341]:
```
variables = ['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD', 'YEAR', 'sector',
       'ucr-rank', 'VOTING_PRECINCT', 'ANC', 'LATITUDE',
       'SPAN', 'TIME_TO_REPORT', 'MONTH', 'DAY', 'hour', 'dayofyear',
       'week', 'weekofyear', 'dayofweek', 'weekday', 'quarter',
        'offensegroup_property', 'offensegroup_violent',
       'SHIFT_day', 'SHIFT_evening', 'SHIFT_midnight', 'METHOD_gun',
       'METHOD_knife', 'METHOD_others']
x = crime_model[variables]
y = crime_model['OFFENSE_code']
```

[342]:
```
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

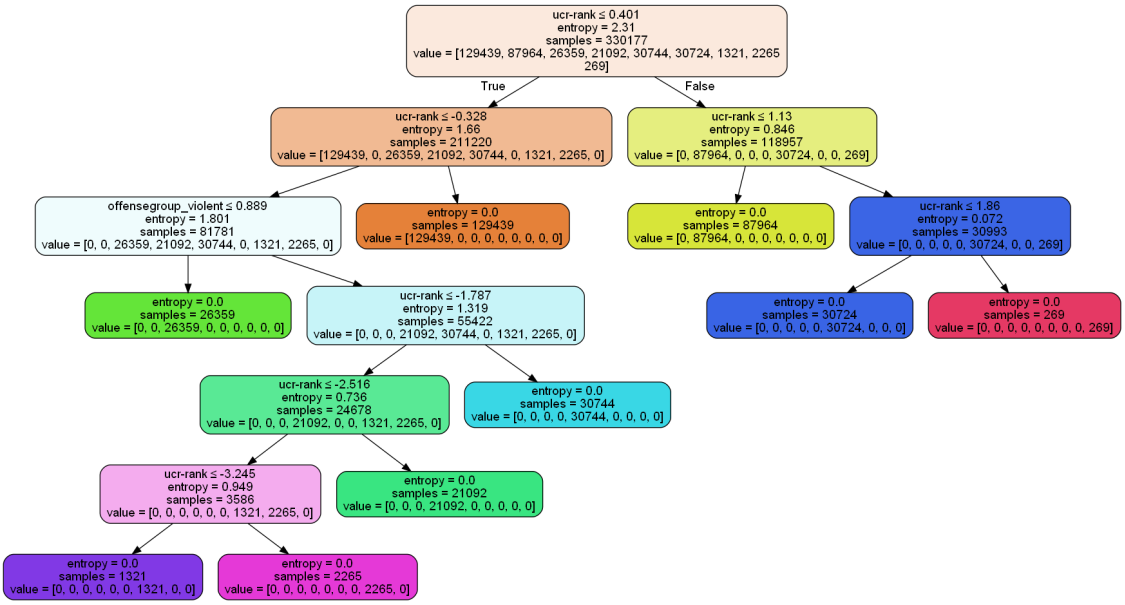
[343]:
```
dtree =␣
 ↪DecisionTreeClassifier(criterion='entropy',max_depth=9,min_samples_leaf=15,min_samples_spli
dtree.fit(x_train,y_train)
y_pred = dtree.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

[343]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=9, min_samples_leaf=15,
                       min_samples_split=14)

[[42816     0     0     0     0     0     0     0     0]
 [    0 29317     0     0     0     0     0     0     0]
 [    0     0  9003     0     0     0     0     0     0]
 [    0     0     0  7127     0     0     0     0     0]
 [    0     0     0     0 10228     0     0     0     0]
 [    0     0     0     0     0 10294     0     0     0]
 [    0     0     0     0     0     0   449     0     0]
 [    0     0     0     0     0     0     0   750     0]
 [    0     0     0     0     0     0     0     0    75]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00     42816
           2       1.00      1.00      1.00     29317
           3       1.00      1.00      1.00      9003
```

```
           4       1.00      1.00      1.00      7127
           5       1.00      1.00      1.00     10228
           6       1.00      1.00      1.00     10294
           7       1.00      1.00      1.00       449
           8       1.00      1.00      1.00       750
           9       1.00      1.00      1.00        75

    accuracy                           1.00    110059
   macro avg       1.00      1.00      1.00    110059
weighted avg       1.00      1.00      1.00    110059
```

[344]:
```
dot_data = StringIO()
tree.export_graphviz(dtree, out_file = dot_data, feature_names =␣
 ↪crime_model[variables].columns,
                     filled = True, rounded = True, special_characters = True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_png("decisiontree_offense_01.png")
Image(graph[0].create_png())
```

[344]:



Clearly, we can completely derive offense from ucr-rank. What if we delete ucr-rank?

[345]:
```
variables = ['CENSUS_TRACT', 'LONGITUDE', 'DISTRICT', 'WARD', 'YEAR', 'sector',
       'VOTING_PRECINCT', 'ANC', 'LATITUDE',
       'SPAN', 'TIME_TO_REPORT', 'MONTH', 'DAY', 'hour', 'dayofyear',
       'week', 'weekofyear', 'dayofweek', 'weekday', 'quarter',
        'offensegroup_property', 'offensegroup_violent',
       'SHIFT_day', 'SHIFT_evening', 'SHIFT_midnight', 'METHOD_gun',
```

```
       'METHOD_knife', 'METHOD_others']
x = crime_model[variables]
y = crime_model['OFFENSE_code']
```

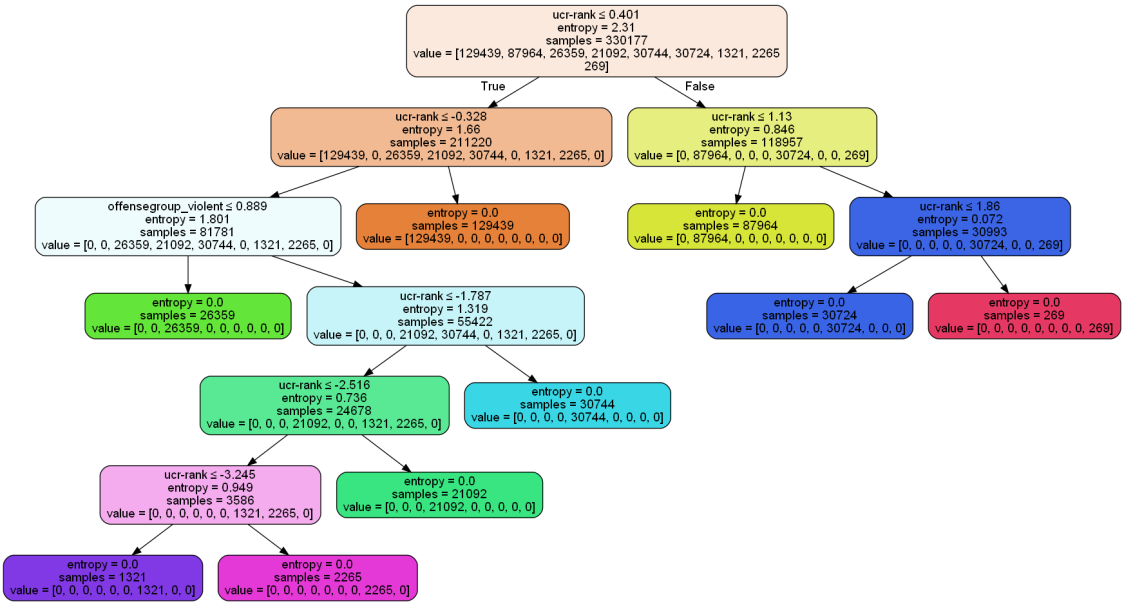
[346]:
```
x_train, x_test, y_train, y_test =  train_test_split(x, y,random_state=42)
x_train.dropna(inplace=True)
x_test.dropna(inplace=True)
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
```

[347]:
```
dtree =␣
 ↪DecisionTreeClassifier(criterion='entropy',max_depth=9,min_samples_leaf=15,min_samples_spli
dtree.fit(x_train,y_train)
y_pred = dtree.predict(x_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

[347]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=9, min_samples_leaf=15,
                       min_samples_split=14)

[[33487  8465   487     0     0   377     0     0     0]
 [11737 16865   423     0     0   292     0     0     0]
 [ 4745  2971  1070     0     0   217     0     0     0]
 [    0     0     0  2774  4302     0    20    31     0]
 [    0     0     0   761  9383     0    21    63     0]
 [ 5429  3767   468     0     0   630     0     0     0]
 [    0     0     0    38   160     0   246     5     0]
 [    0     0     0    61   490     0     8   191     0]
 [   65     6     4     0     0     0     0     0     0]]
              precision    recall  f1-score   support

           1       0.60      0.78      0.68     42816
           2       0.53      0.58      0.55     29317
           3       0.44      0.12      0.19      9003
           4       0.76      0.39      0.52      7127
           5       0.65      0.92      0.76     10228
           6       0.42      0.06      0.11     10294
           7       0.83      0.55      0.66       449
           8       0.66      0.25      0.37       750
           9       0.00      0.00      0.00        75

    accuracy                           0.59    110059
   macro avg       0.54      0.41      0.43    110059
weighted avg       0.57      0.59      0.55    110059
```

[348]:
```
tree.export_graphviz(dtree, out_file = dot_data, feature_names =␣
 ↪crime_model[variables].columns,
                     filled = True, rounded = True, special_characters = True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_png("decisiontree_offense_02.png")
Image(graph[0].create_png())
```

[348]:



7 6 Summary

Through the data mining technology, we have carried out quite in-depth analysis on the data set, including dealing with missing values, detecting and deleting outliers, drawing features bar plots, analyzing data correlation and correlation with time changes, classification task, clustering task, regression task, housing price prediction problem, PCA dimension reduction, crime and bank correlation analysis.

We came to a lot of interesting conclusions, which made us feel quite accomplished. It turned out to be such a wonderful thing to explore data with data mining techniques. We enjoyed this journey of data science.