Final Project2

December 2, 2019

[1]:
```
#Final Project - Dated 22 Nov 2019
```

[90]:
```
#required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import scipy.stats as st
import plotly.figure_factory as ff
from sklearn.preprocessing import StandardScaler
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans, DBSCAN
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
from IPython.display import IFrame
import folium
from folium import plugins
from folium.plugins import MarkerCluster, FastMarkerCluster, HeatMapWithTime

# make sure to run this line to
# conda install -c conda-forge folium
```

[2]:
```
#import datasets
columns = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
two1 = pd.read_csv('Chicago_Crimes_2001_to_2004.csv', usecols= columns)
two5 = pd.read_csv('Chicago_Crimes_2005_to_2007.csv', usecols= columns)
two8 = pd.read_csv('Chicago_Crimes_2008_to_2011.csv', usecols= columns)
two12 = pd.read_csv('Chicago_Crimes_2012_to_2017.csv', usecols= columns)
community_to_major_section = pd.read_csv('community_to_major_section.csv')
```

```
C:\Users\ibray\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3057: DtypeWarning:

Columns (9) have mixed types. Specify dtype option on import or set
```

low_memory=False.

[3]:
```
#Creating seasons variable

#splitting - 1
datetimestamp = two1['Date'].str.split(n=1, expand=True)
datetimestamp.columns = ['date_id{}'.format(x+1) for x in datetimestamp.columns]
two1 = two1.join(datetimestamp)

#splitting - 2
datetimestamp = two5['Date'].str.split(n=1, expand=True)
datetimestamp.columns = ['date_id{}'.format(x+1) for x in datetimestamp.columns]
two5 = two5.join(datetimestamp)

#splitting - 3
datetimestamp = two8['Date'].str.split(n=1, expand=True)
datetimestamp.columns = ['date_id{}'.format(x+1) for x in datetimestamp.columns]
two8 = two8.join(datetimestamp)

#splitting - 4
datetimestamp = two12['Date'].str.split(n=1, expand=True)
datetimestamp.columns = ['date_id{}'.format(x+1) for x in datetimestamp.columns]
two12 = two12.join(datetimestamp)

#seasons
```

[4]:
```
two1[['day','month','year']] = two1.date_id1.str.split('/', expand=True)
two5[['day','month','year']] = two5.date_id1.str.split('/', expand=True)
two8[['day','month','year']] = two8.date_id1.str.split('/', expand=True)
two12[['day','month','year']] = two12.date_id1.str.split('/', expand=True)
```

[5]:
```
def replace_check(row):
    if row == '06' or row == '07' or row == '08':
        val = 'Summer'
    elif row == '03' or row == '04' or row == '05':
        val = 'Spring'
    elif row == '12' or row == '01' or row == '02':
        val = 'Winter'
    else:
        val = 'Fall'
    return val
```

[6]:
```
#Seasons Initialize

two1['Seasons'] = two1.month.apply(replace_check)
two5['Seasons'] = two5.month.apply(replace_check)
two8['Seasons'] = two8.month.apply(replace_check)
two12['Seasons'] = two12.month.apply(replace_check)
```

```
#Seasons Done
```

[7]:
```
#Time Splitting - 1
two1[['hour','minute','second']] = two1.date_id2.str.split(':', expand=True)

#Time Splitting - 2
two5[['hour','minute','second']] = two5.date_id2.str.split(':', expand=True)

#Time Splitting - 3
two8[['hour','minute','second']] = two8.date_id2.str.split(':', expand=True)

#Time Splitting - 4
two12[['hour','minute','second']] = two12.date_id2.str.split(':', expand=True)
```

[8]:
```
#TimeSplit2 - 1
two1[['second', 'dayperiod']] = two1.second.str.split(' ', expand=True)

#TimeSplit2 - 2
two5[['second', 'dayperiod']] = two5.second.str.split(' ', expand=True)

#TimeSplit2 - 3
two8[['second', 'dayperiod']] = two8.second.str.split(' ', expand=True)

#TimeSplit2 - 4
two12[['second', 'dayperiod']] = two12.second.str.split(' ', expand=True)
```

[9]:
```
def period_replace_check(dayperiod,hour,minute):
    try:
        if dayperiod == "AM":
            if int(hour) in [5,6,7,8,9,10,11] and int(minute) in range(0,60):
                val = 'Morning'
            elif int(hour) in [12,1,2,3,4] and int(minute) in range(0,60):
                val = 'Night'
        else:
            if int(hour) in [5,6,7,8] and int(minute) in range(0,60):
                val = 'Evening'
            elif int(hour) in [12,1,2,3,4] and int(minute) in range(0,60):
                val = 'Noon'
            elif int(hour) in [9,10,11] and int(minute) in range(0,60):
                val = 'Night'
    except:
        val = ''
    return val
```

[10]:
```
#TimePeriod - Initialize
```

[11]:
```
two1['TimePeriod'] = two1.apply(lambda x: period_replace_check(x.dayperiod, x.
↪hour, x.minute), axis=1)
```

[12]:
```
two5['TimePeriod'] = two5.apply(lambda x: period_replace_check(x.dayperiod, x.
 ↪hour, x.minute), axis=1)
```

[13]:
```
two8['TimePeriod'] = two8.apply(lambda x: period_replace_check(x.dayperiod, x.
 ↪hour, x.minute), axis=1)
```

[14]:
```
two12['TimePeriod'] = two12.apply(lambda x: period_replace_check(x.dayperiod, x.
 ↪hour, x.minute), axis=1)
```

[15]:
```
#TimePeriod - End
```

[16]:
```
# Major section - Begin
```

[17]:
```
# create dictionary of community areas to major section of chicago using␣
 ↪community_to_major_section csv file
major_section_dic = community_to_major_section.set_index('Community␣
 ↪Area')['Major Section'].to_dict()
```

[18]:
```
two1['Major Section'] = two1['Community Area'].replace(major_section_dic)
```

[19]:
```
two5['Major Section'] = two5['Community Area'].replace(major_section_dic)
```

[20]:
```
two8['Major Section'] = two8['Community Area'].replace(major_section_dic)
```

[21]:
```
two12['Major Section'] = two12['Community Area'].replace(major_section_dic)
```

[22]:
```
# Major section - End
```

[23]:
```
major_section_dic
```

[23]:
```
{1: 'Far North side',
 2: 'Far North side',
 3: 'Far North side',
 4: 'Far North side',
 5: 'North side',
 6: 'North side',
 7: 'North side',
 8: 'Central',
 9: 'Far North side',
 10: 'Far North side',
 11: 'Far North side',
 12: 'Far North side',
 13: 'Far North side',
 14: 'Far North side',
 15: 'Northwest side',
 16: 'Northwest side',
 17: 'Northwest side',
 18: 'Northwest side',
 19: 'Northwest side',
 20: 'Northwest side',
 21: 'North side',
 22: 'North side',
```

```
23: 'West side',
24: 'West side',
25: 'West side',
26: 'West side',
27: 'West side',
28: 'West side',
29: 'West side',
30: 'West side',
31: 'West side',
32: 'Central',
33: 'Central',
34: 'South side',
35: 'South side',
36: 'South side',
37: 'South side',
38: 'South side',
39: 'South side',
40: 'South side',
41: 'South side',
42: 'South side',
43: 'South side',
44: 'Far Southeast side',
45: 'Far Southeast side',
46: 'Far Southeast side',
47: 'Far Southeast side',
48: 'Far Southeast side',
49: 'Far Southeast side',
50: 'Far Southeast side',
51: 'Far Southeast side',
52: 'Far Southeast side',
53: 'Far Southeast side',
54: 'Far Southeast side',
55: 'Far Southeast side',
56: 'Southwest side',
57: 'Southwest side',
58: 'Southwest side',
59: 'Southwest side',
60: 'South side',
61: 'Southwest side',
62: 'Southwest side',
63: 'Southwest side',
64: 'Southwest side',
65: 'Southwest side',
66: 'Southwest side',
67: 'Southwest side',
68: 'Southwest side',
69: 'South side',
```

```
 70: 'Far Southwest side',
 71: 'Far Southwest side',
 72: 'Far Southwest side',
 73: 'Far Southwest side',
 74: 'Far Southwest side',
 75: 'Far Southwest side',
 76: 'Far North side',
 77: 'Far North side'}
```

[24]:
```
#merge datasets
finaldatasets = pd.concat([two1, two5, two8, two12])
finaldatasets.isnull().sum()

#finaldatasets = finaldatasets.sample(n=200)
```

[24]:
```
Date                        0
Primary Type                0
Location Description     1990
Arrest                      0
District                   91
Community Area          702092
Year                        1
date_id1                    0
date_id2                    1
day                         0
month                       1
year                        1
Seasons                     0
hour                        1
minute                      1
second                      1
dayperiod                   1
TimePeriod                  0
Major Section           702092
dtype: int64
```

[25]:
```
finaldatasets.head()
```

[25]:

	Date	Primary Type	Location Description	Arrest	District	Community Area	Year	date_id1	date_id2	day	month
0	01/01/2004 12:01:00 AM	THEFT	RESIDENCE	False	4.0	46.0	2004.0	01/01/2004	12:01:00 AM	01	01
1	03/01/2003 12:00:00 AM	OTHER OFFENSE	RESIDENCE	False	9.0	61.0	2003.0	03/01/2003	12:00:00 AM	03	01
2	06/20/2004 11:00:00 AM	OFFENSE INVOLVING CHILDREN	RESIDENCE	False	14.0	22.0	2004.0	06/20/2004	11:00:00 AM	06	20
3	12/30/2004 08:00:00 PM	THEFT	OTHER								
4	05/01/2003 01:00:00 AM	THEFT	RESIDENCE								

```
3  False      25.0            20.0  2004.0  12/30/2004  08:00:00 PM  12    30
4  False      22.0            49.0  2003.0  05/01/2003  01:00:00 AM  05    01

   year Seasons hour minute second dayperiod TimePeriod       Major Section
0  2004  Winter   12     01     00        AM      Night  Far Southeast side
1  2003  Winter   12     00     00        AM      Night      Southwest side
2  2004    Fall   11     00     00        AM    Morning          North side
3  2004    Fall   08     00     00        PM    Evening      Northwest side
4  2003  Winter   01     00     00        AM      Night  Far Southeast side
```

[26]:
```
# finaldatasets.to_excel(index = False)
#export_csv = finaldatasets.to_csv␣
 ↪(r'\Users\mustafahabeeb\Desktop\export_dataframe.csv', index = None,␣
 ↪header=True)
```

[27]:
```
finaldatasets['Community Area'].unique()
```

[27]:
```
array([46., 61., 22., 20., 49., 29., 50., 73.,  8., 77., 65., 43., 59.,
       66., 23., 62., 67., 32., 70., 10., 25., 19., 44., 45., 28., 68.,
       30., 40., 11.,  3., 71., 42., 17., 34., 63., 37., 69., 55., 35.,
       27., 48., 24., 72., 18., 15., 12.,  6.,  7., 52., 60., 26., 58.,
       74., 64.,  5.,  2., 53., 56., 21., 31., 51.,  4., 33., 39., 38.,
       16., 41.,  1., 75., 14., 57., 36., 13., 76., 47.,  9., nan, 54.,
        0.])
```

[28]:
```
len(finaldatasets['Primary Type'].unique())
```

[28]:
```
36
```

[29]:
```
finaldatasets['Primary Type'].unique()
```

[29]:
```
array(['THEFT', 'OTHER OFFENSE', 'OFFENSE INVOLVING CHILDREN',
       'CRIM SEXUAL ASSAULT', 'MOTOR VEHICLE THEFT', 'SEX OFFENSE',
       'DECEPTIVE PRACTICE', 'BATTERY', 'BURGLARY', 'WEAPONS VIOLATION',
       'PUBLIC PEACE VIOLATION', 'NARCOTICS', 'GAMBLING', 'PROSTITUTION',
       'LIQUOR LAW VIOLATION', 'INTERFERENCE WITH PUBLIC OFFICER',
       'CRIMINAL DAMAGE', 'ASSAULT', 'STALKING', 'ARSON',
       'CRIMINAL TRESPASS', 'HOMICIDE', 'ROBBERY', 'OBSCENITY',
       'KIDNAPPING', 'INTIMIDATION', 'RITUALISM', 'DOMESTIC VIOLENCE',
       'OTHER NARCOTIC VIOLATION', 'PUBLIC INDECENCY', 'NON-CRIMINAL',
       'False', 'HUMAN TRAFFICKING', 'CONCEALED CARRY LICENSE VIOLATION',
       'NON - CRIMINAL', 'NON-CRIMINAL (SUBJECT SPECIFIED)'], dtype=object)
```

[30]:
```
finaldatasets['Primary Type'] = finaldatasets['Primary Type'].str.lower()
```

[31]:
```
primary_type_dict = {
  "theft": "THEFT",
  "other offense": "OFFENSE",
  "offense involving children": "OFFENSE",
  "crim sexual assault": "ASSAULT",
  "motor vehicle theft": "THEFT",
```

```
  "sex offense": "OFFENSE",
  "deceptive practice": "OTHER",
  "battery": "ASSAULT",
  "burglary": "THEFT",
  "weapons violation": "VIOLATION",
  "public peace violation": "VIOLATION",
  "gambling": "WHITE COLLAR CRIME",
  "prostitution": "WHITE COLLAR CRIME",
  "liquor law violation": "VIOLATION",
  "interference with public officer": "VIOLATION",
  "criminal damage": "DAMAGES",
  "assault": "ASSAULT",
  "stalking": "VIOLATION",
  "arson": "DAMAGES",
  "criminal trespass": "VIOLATION",
  "homicide": "MURDER",
  "obscenity": "OTHER",
  "kidnapping": "KIDNAPPING",
  "intimidation": "ASSAULT",
  "ritualism": "OTHER",
  "domestic violence": "ASSAULT",
  "other narcotic violation": "WHITE COLLAR CRIME",
  "public indecency": "VIOLATION",
  "non-criminal": "OTHER",
  "false": "OTHER",
  "human trafficking": "KIDNAPPING",
  "concealed carry license violation": "OTHER",
  "non - criminal": "OTHER",
  "non-criminal (subject specified)": "OTHER",
  "robbery":"THEFT",
  "narcotics":"WHITE COLLAR CRIME"
}
```

[32]:
```
finaldatasets['Primary Crime Type'] = finaldatasets['Primary Type'].
↪replace(primary_type_dict)
```

[33]:
```
finaldatasets['Location Description'].unique()
```

[33]:
```
array(['RESIDENCE', 'OTHER', 'APARTMENT', 'RESIDENCE PORCH/HALLWAY',
       'GAS STATION', 'COMMERCIAL / BUSINESS OFFICE', 'STREET', 'BANK',
       'SMALL RETAIL STORE', 'DEPARTMENT STORE', 'SIDEWALK',
       'APPLIANCE STORE', 'HOTEL/MOTEL', 'MEDICAL/DENTAL OFFICE',
       'PARKING LOT/GARAGE(NON.RESID.)', 'ALLEY',
       'CHURCH/SYNAGOGUE/PLACE OF WORSHIP', 'DAY CARE CENTER',
       'RESTAURANT', 'COLLEGE/UNIVERSITY GROUNDS',
       'SCHOOL, PUBLIC, BUILDING', 'HOSPITAL BUILDING/GROUNDS',
       'WAREHOUSE', 'FACTORY/MANUFACTURING BUILDING',
       'SCHOOL, PRIVATE, GROUNDS', 'GROCERY FOOD STORE', 'CHA APARTMENT',
```

'SCHOOL, PUBLIC, GROUNDS', 'VEHICLE NON-COMMERCIAL',
'GOVERNMENT BUILDING/PROPERTY', 'AIRPORT/AIRCRAFT',
'ATM (AUTOMATIC TELLER MACHINE)', 'VACANT LOT/LAND',
'POLICE FACILITY/VEH PARKING LOT', 'TAVERN/LIQUOR STORE',
'CHA HALLWAY/STAIRWELL/ELEVATOR', 'RESIDENCE-GARAGE',
'PARK PROPERTY', 'CHA PARKING LOT/GROUNDS', 'ABANDONED BUILDING',
'SCHOOL, PRIVATE, BUILDING', 'CURRENCY EXCHANGE', 'BARBERSHOP',
'NURSING HOME/RETIREMENT HOME', 'CHA STAIRWELL', 'AUTO',
'BASEMENT', 'ANIMAL HOSPITAL', 'RESIDENTIAL YARD (FRONT/BACK)',
'JAIL / LOCK-UP FACILITY', 'RETAIL STORE', 'TAVERN',
'GAS STATION DRIVE/PROP.', 'FEDERAL BUILDING', 'HOTEL', 'HALLWAY',
'TRUCK', 'GANGWAY', 'POOL ROOM', 'PARKING LOT', 'HOUSE',
'COACH HOUSE', 'PORCH', 'CLUB', 'VACANT LOT', 'ATHLETIC CLUB',
'YARD', 'AIRPORT BUILDING NON-TERMINAL - SECURE AREA', 'CAR WASH',
'CHA PARKING LOT', 'LOADING DOCK', 'CHA ELEVATOR', 'LAKE',
'RAILROAD PROPERTY', 'CTA GARAGE / OTHER PROPERTY', 'VESTIBULE',
'CHA HALLWAY', 'AIRPORT TERMINAL UPPER LEVEL - SECURE AREA',
'DUMPSTER', 'GARAGE', 'FOREST PRESERVE', 'BAR OR TAVERN',
'COLLEGE/UNIVERSITY RESIDENCE HALL', 'CHA PLAY LOT', 'CHA GROUNDS',
'HOSPITAL', 'RIVER', 'FIRE STATION', 'DRUG STORE', 'CTA BUS',
'CTA PLATFORM', 'HIGHWAY/EXPRESSWAY', 'CLEANING STORE',
'DRIVEWAY - RESIDENTIAL', 'OTHER RAILROAD PROP / TRAIN DEPOT',
'CTA TRAIN', 'VEHICLE-COMMERCIAL',
'OTHER COMMERCIAL TRANSPORTATION', 'LIBRARY', 'DELIVERY TRUCK',
'CEMETARY', 'CONSTRUCTION SITE', 'BOAT/WATERCRAFT',
'SPORTS ARENA/STADIUM', 'LAKEFRONT/WATERFRONT/RIVERBANK',
'TAXICAB', 'WOODED AREA', 'COUNTY JAIL', 'STAIRWELL', 'YMCA',
'CHURCH PROPERTY', 'MOVIE HOUSE/THEATER', 'BOWLING ALLEY',
'COIN OPERATED MACHINE', 'SAVINGS AND LOAN', 'SEWER',
'LIVERY STAND OFFICE', 'GARAGE/AUTO REPAIR', 'CREDIT UNION',
'CHURCH', 'CHA BREEZEWAY', 'NEWSSTAND', 'BRIDGE', 'CHA LOBBY', nan,
'PRAIRIE', 'DRIVEWAY', 'PUBLIC GRAMMAR SCHOOL',
'JUNK YARD/GARBAGE DUMP', 'SCHOOL YARD', 'FUNERAL PARLOR',
'OFFICE', 'LIQUOR STORE', 'BARBER SHOP/BEAUTY SALON', 'TAXI CAB',
'CTA "L" TRAIN', 'PUBLIC HIGH SCHOOL', 'TRUCKING TERMINAL',
'FACTORY', 'TRAILER', 'MOTEL', 'CTA PROPERTY', 'CONVENIENCE STORE',
'LAUNDRY ROOM', 'PAWN SHOP', 'AIRPORT PARKING LOT',
'AIRPORT TERMINAL MEZZANINE - NON-SECURE AREA', 'LIVERY AUTO',
'RIVER BANK', 'BANQUET HALL', 'VEHICLE - DELIVERY TRUCK',
'ROOMING HOUSE', 'AIRCRAFT', 'CTA BUS STOP',
'AIRPORT TERMINAL LOWER LEVEL - SECURE AREA',
'AIRPORT EXTERIOR - SECURE AREA',
'AIRPORT EXTERIOR - NON-SECURE AREA',
'AIRPORT TERMINAL LOWER LEVEL - NON-SECURE AREA',
'AIRPORT TERMINAL UPPER LEVEL - NON-SECURE AREA',
'AIRPORT VENDING ESTABLISHMENT',
'AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA',

```
        'AIRPORT TRANSPORTATION SYSTEM (ATS)', '1134', 'NURSING HOME',
        'CTA "L" PLATFORM', 'CTA STATION', 'VEHICLE - OTHER RIDE SERVICE',
        'CTA TRACKS - RIGHT OF WAY', 'ELEVATOR', 'CLEANERS/LAUNDROMAT',
        'EXPRESSWAY EMBANKMENT', 'GOVERNMENT BUILDING', 'POOLROOM',
        'LAGOON'], dtype=object)
```

[34]:
```
finaldatasets['Location Description'] = finaldatasets['Location Description'].
↪str.lower()
```

[35]:
```
location_dict = {'residence': 'RESIDENTIAL',
 'other': 'OTHER',
 'apartment': 'RESIDENTIAL',
 'residence porch/hallway': 'RESIDENTIAL',
 'gas station': 'COMMERCIAL',
 'commercial / business office': 'COMMERCIAL',
 'street': 'PUBLIC',
 'bank': 'PUBLIC',
 'small retail store': 'COMMERCIAL',
 'department store': 'COMMERCIAL',
 'sidewalk': 'PUBLIC',
 'appliance store': 'COMMERCIAL',
 'hotel/motel': 'PUBLIC',
 'medical/dental office': 'HEALTHCARE FACILITY',
 'parking lot/garage(non.resid.)': 'PUBLIC',
 'alley': 'PUBLIC',
 'church/synagogue/place of worship': 'PLACE OF WORSHIP',
 'day care center': 'EDUCATION',
 'restaurant': 'COMMERCIAL',
 'college/university grounds': 'EDUCATION',
 'school, public, building': 'EDUCATION',
 'hospital building/grounds': 'HEALTHCARE FACILITY',
 'warehouse': 'PRIVATE',
 'factory/manufacturing building': 'PRIVATE',
 'school, private, grounds': 'EDUCATION',
 'grocery food store': 'COMMERCIAL',
 'cha apartment': 'RESIDENTIAL',
 'school, public, grounds': 'EDUCATION',
 'vehicle non-commercial': 'PRIVATE',
 'government building/property': 'GOVERNMENT',
 'airport/aircraft': 'PRIVATE',
 'atm (automatic teller machine)': 'PUBLIC',
 'vacant lot/land': 'PRIVATE',
 'police facility/veh parking lot': 'GOVERNMENT',
 'tavern/liquor store': 'COMMERCIAL',
 'cha hallway/stairwell/elevator': 'RESIDENTIAL',
 'residence-garage': 'RESIDENTIAL',
 'park property': 'GOVERNMENT',
 'cha parking lot/grounds': 'RESIDENTIAL',
```

```
'abandoned building': 'PRIVATE',
'school, private, building': 'EDUCATION',
'currency exchange': 'COMMERCIAL',
'barbershop': 'COMMERCIAL',
'nursing home/retirement home': 'PUBLIC',
'cha stairwell': 'RESIDENTIAL',
'auto': 'OTHER',
'basement': 'PRIVATE',
'animal hospital': 'HEALTHCARE FACILITY',
'residential yard (front/back)': 'RESIDENTIAL',
'jail / lock-up facility': 'GOVERNMENT',
'retail store': 'COMMERCIAL',
'tavern': 'COMMERCIAL',
'gas station drive/prop.': 'COMMERCIAL',
'federal building': 'GOVERNMENT',
'hotel': 'PUBLIC',
'hallway': 'PUBLIC',
'truck': 'PRIVATE',
'gangway': 'OTHER',
'pool room': 'PRIVATE',
'parking lot': 'PUBLIC',
'house': 'RESIDENTIAL',
'coach house': 'RESIDENTIAL',
'porch': 'RESIDENTIAL',
'club': 'COMMERCIAL',
'vacant lot': 'PRIVATE',
'athletic club': 'COMMERCIAL',
'yard': 'RESIDENTIAL',
'airport building non-terminal - secure area': 'PUBLIC',
'car wash': 'COMMERCIAL',
'cha parking lot': 'RESIDENTIAL',
'loading dock': 'PRIVATE',
'cha elevator': 'RESIDENTIAL',
'lake': 'PUBLIC',
'railroad property': 'PRIVATE',
'cta garage / other property': 'PUBLIC',
'vestibule': 'OTHER',
'cha hallway': 'RESIDENTIAL',
'airport terminal upper level - secure area': 'PUBLIC',
'dumpster': 'PUBLIC',
'garage': 'PUBLIC',
'forest preserve': 'PRIVATE',
'bar or tavern': 'COMMERCIAL',
'college/university residence hall': 'EDUCATION',
'cha play lot': 'RESIDENTIAL',
'cha grounds': 'RESIDENTIAL',
'hospital': 'HEALTHCARE FACILITY',
```

```
'river': 'GOVERNMENT',
'fire station': 'GOVERNMENT',
'drug store': 'COMMERCIAL',
'cta bus': 'PUBLIC',
'cta platform': 'PUBLIC',
'highway/expressway': 'PUBLIC',
'cleaning store': 'COMMERCIAL',
'driveway - residential': 'RESIDENTIAL',
'other railroad prop / train depot': 'PRIVATE',
'cta train': 'PUBLIC',
'vehicle-commercial': 'COMMERCIAL',
'other commercial transportation': 'COMMERCIAL',
'library': 'PUBLIC',
'delivery truck': 'PRIVATE',
'cemetary': 'PRIVATE',
'construction site': 'PRIVATE',
'boat/watercraft': 'PRIVATE',
'sports arena/stadium': 'PRIVATE',
'lakefront/waterfront/riverbank': 'GOVERNMENT',
'taxicab': 'PRIVATE',
'wooded area': 'PRIVATE',
'county jail': 'GOVERNMENT',
'stairwell': 'PUBLIC',
'ymca': 'PUBLIC',
'church property': 'PLACE OF WORSHIP',
'movie house/theater': 'COMMERCIAL',
'bowling alley': 'COMMERCIAL',
'coin operated machine': 'PUBLIC',
'savings and loan': 'PRIVATE',
'sewer': 'GOVERNMENT',
'livery stand office': 'OTHER',
'garage/auto repair': 'COMMERCIAL',
'credit union': 'PUBLIC',
'church': 'PLACE OF WORSHIP',
'cha breezeway': 'RESIDENTIAL',
'newsstand': 'COMMERCIAL',
'bridge': 'PUBLIC',
'cha lobby': 'RESIDENTIAL',
None: '',
'prairie': 'GOVERNMENT',
'driveway': 'PRIVATE',
'public grammar school': 'EDUCATION',
'junk yard/garbage dump': 'GOVERNMENT',
'school yard': 'EDUCATION',
'funeral parlor': 'COMMERCIAL',
'office': 'PRIVATE',
'liquor store': 'COMMERCIAL',
```

```
 'barber shop/beauty salon': 'COMMERCIAL',
 'taxi cab': 'PRIVATE',
 'cta "l" train': 'PUBLIC',
 'public high school': 'EDUCATION',
 'trucking terminal': 'PRIVATE',
 'factory': 'PRIVATE',
 'trailer': 'PRIVATE',
 'motel': 'PUBLIC',
 'cta property': 'GOVERNMENT',
 'convenience store': 'COMMERCIAL',
 'laundry room': 'COMMERCIAL',
 'pawn shop': 'COMMERCIAL',
 'airport parking lot': 'PUBLIC',
 'airport terminal mezzanine - non-secure area': 'PUBLIC',
 'livery auto': 'OTHER',
 'river bank': 'GOVERNMENT',
 'banquet hall': 'PRIVATE',
 'vehicle - delivery truck': 'PRIVATE',
 'rooming house': 'PUBLIC',
 'aircraft': 'PRIVATE',
 'cta bus stop': 'PUBLIC',
 'airport terminal lower level - secure area': 'PRIVATE',
 'airport exterior - secure area': 'PRIVATE',
 'airport exterior - non-secure area': 'PUBLIC',
 'airport terminal lower level - non-secure area': 'PUBLIC',
 'airport terminal upper level - non-secure area': 'PUBLIC',
 'airport vending establishment': 'PUBLIC',
 'airport building non-terminal - non-secure area': 'PUBLIC',
 'airport transportation system (ats)': 'PRIVATE',
 '1134': 'OTHER',
 'nursing home': 'PRIVATE',
 'cta "l" platform': 'PUBLIC',
 'cta station': 'PUBLIC',
 'vehicle - other ride service': 'PRIVATE',
 'cta tracks - right of way': 'PUBLIC',
 'elevator': 'PRIVATE',
 'cleaners/laundromat': 'COMMERCIAL',
 'expressway embankment': 'PUBLIC',
 'government building': 'GOVERNMENT',
 'poolroom': 'PRIVATE',
 'lagoon': 'GOVERNMENT'}
```

[36]:
```
finaldatasets['Loc Type'] = finaldatasets['Location Description'].
↪replace(location_dict)
```

[37]:
```
finaldatasets['Loc Type'].unique()
```

[37]:
```
array(['RESIDENTIAL', 'OTHER', 'COMMERCIAL', 'PUBLIC',
       'HEALTHCARE FACILITY', 'PLACE OF WORSHIP', 'EDUCATION', 'PRIVATE',
       'GOVERNMENT', ''], dtype=object)
```

[38]:
```
finaldatasets.describe()
```

[38]:

	District	Community Area	Year
count	7.941195e+06	7.239194e+06	7.941285e+06
mean	1.131215e+01	3.774790e+01	2.007672e+03
std	6.944523e+00	2.156597e+01	4.123451e+00
min	1.000000e+00	0.000000e+00	4.178983e+01
25%	6.000000e+00	2.300000e+01	2.005000e+03
50%	1.000000e+01	3.200000e+01	2.008000e+03
75%	1.700000e+01	5.800000e+01	2.010000e+03
max	3.100000e+01	7.700000e+01	2.017000e+03

[39]:
```
finaldatasets.head()
finaldatasets.columns
```

[39]:
```
Index(['Date', 'Primary Type', 'Location Description', 'Arrest', 'District',
       'Community Area', 'Year', 'date_id1', 'date_id2', 'day', 'month',
       'year', 'Seasons', 'hour', 'minute', 'second', 'dayperiod',
       'TimePeriod', 'Major Section', 'Primary Crime Type', 'Loc Type'],
      dtype='object')
```

[40]:
```
finaldatasets.iloc[1]
```

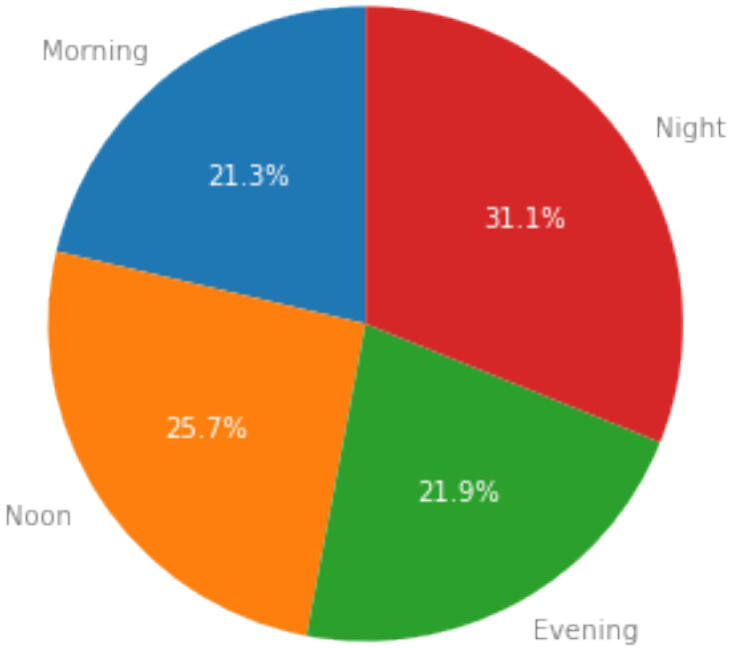
[40]:
```
Date                    03/01/2003 12:00:00 AM
Primary Type                       other offense
Location Description                  residence
Arrest                                    False
District                                      9
Community Area                               61
Year                                       2003
date_id1                             03/01/2003
date_id2                           12:00:00 AM
day                                          03
month                                        01
year                                       2003
Seasons                                  Winter
hour                                         12
minute                                       00
second                                       00
dayperiod                                    AM
TimePeriod                                Night
Major Section                   Southwest side
Primary Crime Type                      OFFENSE
Loc Type                             RESIDENTIAL
Name: 1, dtype: object
```

[41]:
```
finaldatasets.info()
#The columns that are used are
#[Date, Year, TimePeriod, Primary Crime Type, Arrest, Community Area, District,␣
 ↪Major Section, Loc Type]
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7941286 entries, 0 to 1456713
Data columns (total 21 columns):
Date                    object
Primary Type            object
Location Description    object
Arrest                  object
District                float64
Community Area          float64
Year                    float64
date_id1                object
date_id2                object
day                     object
month                   object
year                    object
Seasons                 object
hour                    object
minute                  object
second                  object
dayperiod               object
TimePeriod              object
Major Section           object
Primary Crime Type      object
Loc Type                object
dtypes: float64(3), object(18)
memory usage: 1.3+ GB
```

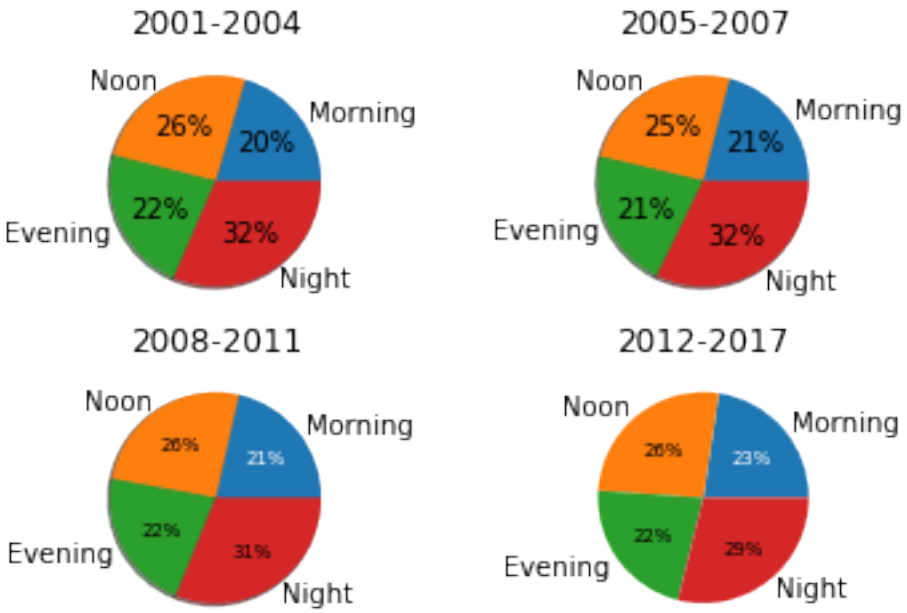
[42]:
```
#Number of crimes committed in 2001 - 2017
#For Final Dataset

labels = ['Morning', 'Noon', 'Evening', 'Night']
sizes = [len(finaldatasets[finaldatasets['TimePeriod'] ==␣
 ↪'Morning']),len(finaldatasets[finaldatasets['TimePeriod'] ==␣
 ↪'Noon']),len(finaldatasets[finaldatasets['TimePeriod'] ==␣
 ↪'Evening']),len(finaldatasets[finaldatasets['TimePeriod'] == 'Night'])]
fig1, ax1 = plt.subplots()
patches, texts, autotexts = ax1.pie(sizes, labels=labels, autopct='%1.1f%%',␣
 ↪startangle=90)

for text in texts:
    text.set_color('grey')
for autotext in autotexts:
```

```
    autotext.set_color('white')
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
plt.show()
```



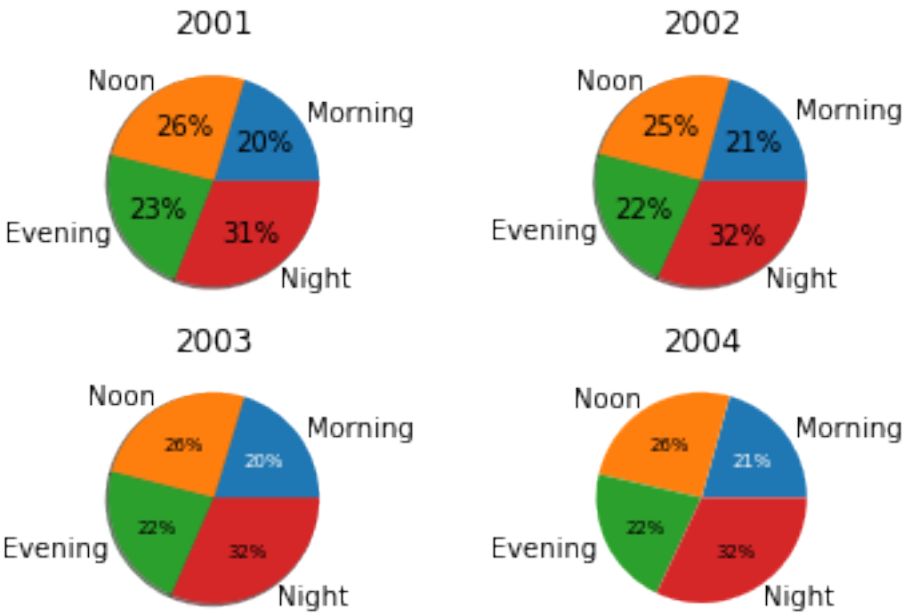
[43]:
```
#Number of crimes committed in 2001 - 2004, 05-07,08-11,12-17

labels = ['Morning', 'Noon', 'Evening', 'Night']
sizes1 = [len(two1[two1['TimePeriod'] ==␣
 ↪'Morning']),len(two1[two1['TimePeriod'] ==␣
 ↪'Noon']),len(two1[two1['TimePeriod'] ==␣
 ↪'Evening']),len(two1[two1['TimePeriod'] == 'Night'])]
sizes2 = [len(two5[two5['TimePeriod'] ==␣
 ↪'Morning']),len(two5[two5['TimePeriod'] ==␣
 ↪'Noon']),len(two5[two5['TimePeriod'] ==␣
 ↪'Evening']),len(two5[two5['TimePeriod'] == 'Night'])]
sizes3 = [len(two8[two8['TimePeriod'] ==␣
 ↪'Morning']),len(two8[two8['TimePeriod'] ==␣
 ↪'Noon']),len(two8[two8['TimePeriod'] ==␣
 ↪'Evening']),len(two8[two8['TimePeriod'] == 'Night'])]
```

```
sizes4 = [len(two12[two12['TimePeriod'] ==␣
 ↪'Morning']),len(two12[two12['TimePeriod'] ==␣
 ↪'Noon']),len(two12[two12['TimePeriod'] ==␣
 ↪'Evening']),len(two12[two12['TimePeriod'] == 'Night'])]
# Make figure and axes
fig, axs = plt.subplots(2, 2)

# A standard pie plot
axs[0, 0].pie(sizes1, labels=labels, autopct='%.0f%%', shadow=True)
axs[0,0].set_title("2001-2004")


# Shift the second slice using explode
axs[0, 1].pie(sizes2, labels=labels, autopct='%.0f%%', shadow=True,
              )
axs[0,1].set_title("2005-2007")

# Adapt radius and text size for a smaller pie
patches, texts, autotexts = axs[1, 0].pie(sizes3, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=True )
axs[1,0].set_title("2008-2011")
# Make percent texts even smaller
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

# Use a smaller explode and turn of the shadow for better visibility
patches, texts, autotexts = axs[1, 1].pie(sizes4, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=False
                                         )
axs[1,1].set_title("2012-2017")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()
```



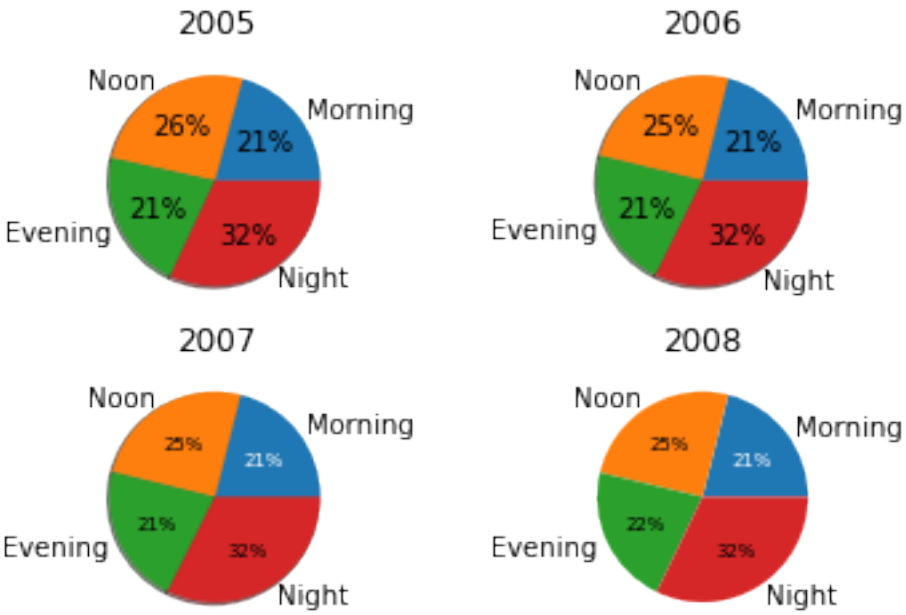
[44]:
```
#Number of crimes committed in 2001 - 2004

labels = ['Morning', 'Noon', 'Evening', 'Night']
one = finaldatasets[finaldatasets['year'] == '2001']
two = finaldatasets[finaldatasets['year'] == '2002']
three = finaldatasets[finaldatasets['year'] == '2003']
four = finaldatasets[finaldatasets['year'] == '2004']
sizes1 = [len(one[one['TimePeriod'] == 'Morning']),len(one[one['TimePeriod'] ==␣
 ↪'Noon']),len(one[one['TimePeriod'] == 'Evening']),len(one[one['TimePeriod']␣
 ↪== 'Night'])]
sizes2 = [len(two[two['TimePeriod'] == 'Morning']),len(two[two['TimePeriod'] ==␣
 ↪'Noon']),len(two[two['TimePeriod'] == 'Evening']),len(two[two['TimePeriod']␣
 ↪== 'Night'])]
sizes3 = [len(three[three['TimePeriod'] ==␣
 ↪'Morning']),len(three[three['TimePeriod'] ==␣
 ↪'Noon']),len(three[three['TimePeriod'] ==␣
 ↪'Evening']),len(three[three['TimePeriod'] == 'Night'])]
sizes4 = [len(four[four['TimePeriod'] ==␣
 ↪'Morning']),len(four[four['TimePeriod'] ==␣
 ↪'Noon']),len(four[four['TimePeriod'] ==␣
 ↪'Evening']),len(four[four['TimePeriod'] == 'Night'])]
# Make figure and axes
fig, axs = plt.subplots(2, 2)

# A standard pie plot
axs[0, 0].pie(sizes1, labels=labels, autopct='%.0f%%', shadow=True)
```

```
axs[0,0].set_title("2001")


# Shift the second slice using explode
axs[0, 1].pie(sizes2, labels=labels, autopct='%.0f%%', shadow=True,
              )
axs[0,1].set_title("2002")

# Adapt radius and text size for a smaller pie
patches, texts, autotexts = axs[1, 0].pie(sizes3, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=True )
axs[1,0].set_title("2003")
# Make percent texts even smaller
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

# Use a smaller explode and turn of the shadow for better visibility
patches, texts, autotexts = axs[1, 1].pie(sizes4, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=False
                                         )
axs[1,1].set_title("2004")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()
```



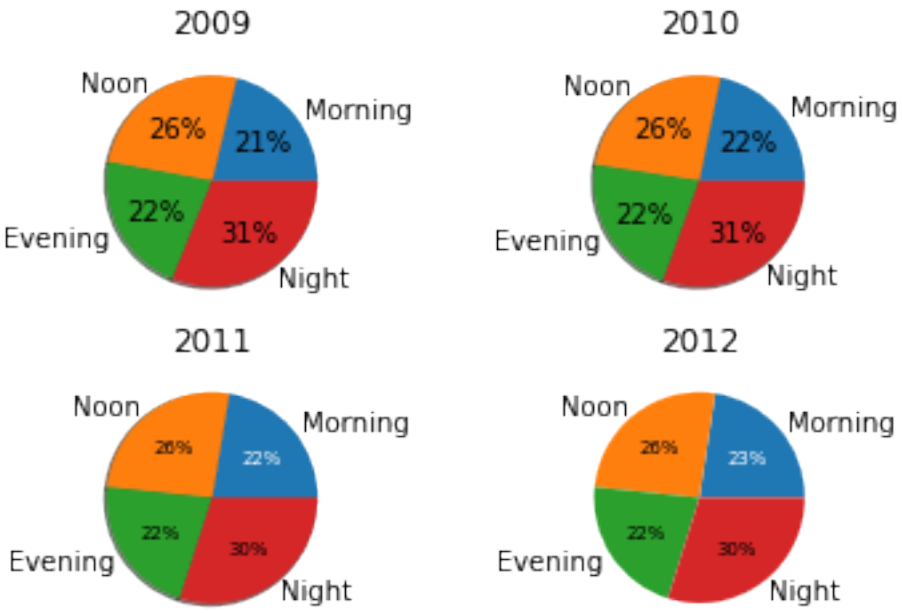
[45]:
```
#Number of crimes committed in 2005 - 2008

labels = ['Morning', 'Noon', 'Evening', 'Night']
one = finaldatasets[finaldatasets['year'] == '2005']
two = finaldatasets[finaldatasets['year'] == '2006']
three = finaldatasets[finaldatasets['year'] == '2007']
four = finaldatasets[finaldatasets['year'] == '2008']
sizes1 = [len(one[one['TimePeriod'] == 'Morning']),len(one[one['TimePeriod'] ==␣
 ↪'Noon']),len(one[one['TimePeriod'] == 'Evening']),len(one[one['TimePeriod']␣
 ↪== 'Night'])]
sizes2 = [len(two[two['TimePeriod'] == 'Morning']),len(two[two['TimePeriod'] ==␣
 ↪'Noon']),len(two[two['TimePeriod'] == 'Evening']),len(two[two['TimePeriod']␣
 ↪== 'Night'])]
sizes3 = [len(three[three['TimePeriod'] ==␣
 ↪'Morning']),len(three[three['TimePeriod'] ==␣
 ↪'Noon']),len(three[three['TimePeriod'] ==␣
 ↪'Evening']),len(three[three['TimePeriod'] == 'Night'])]
sizes4 = [len(four[four['TimePeriod'] ==␣
 ↪'Morning']),len(four[four['TimePeriod'] ==␣
 ↪'Noon']),len(four[four['TimePeriod'] ==␣
 ↪'Evening']),len(four[four['TimePeriod'] == 'Night'])]
# Make figure and axes
fig, axs = plt.subplots(2, 2)

# A standard pie plot
axs[0, 0].pie(sizes1, labels=labels, autopct='%.0f%%', shadow=True)
axs[0,0].set_title("2005")


# Shift the second slice using explode
axs[0, 1].pie(sizes2, labels=labels, autopct='%.0f%%', shadow=True,
              )
axs[0,1].set_title("2006")

# Adapt radius and text size for a smaller pie
patches, texts, autotexts = axs[1, 0].pie(sizes3, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=True )
axs[1,0].set_title("2007")
# Make percent texts even smaller
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

# Use a smaller explode and turn of the shadow for better visibility
patches, texts, autotexts = axs[1, 1].pie(sizes4, labels=labels,
                                          autopct='%.0f%%',
```

```
                                        shadow=False
                                    )
axs[1,1].set_title("2008")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()
```



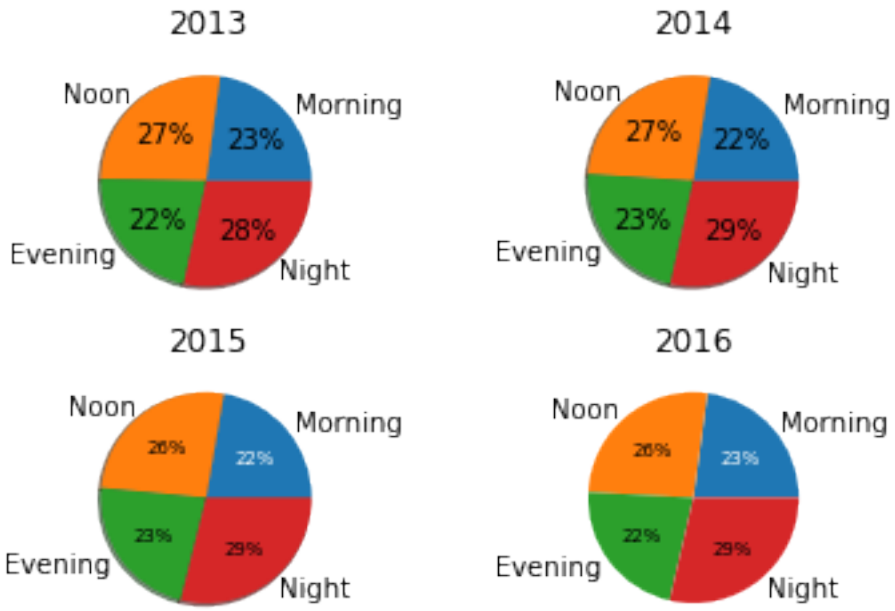
[46]:
```
#Number of crimes committed in 2009 - 2012

labels = ['Morning', 'Noon', 'Evening', 'Night']
one = finaldatasets[finaldatasets['year'] == '2009']
two = finaldatasets[finaldatasets['year'] == '2010']
three = finaldatasets[finaldatasets['year'] == '2011']
four = finaldatasets[finaldatasets['year'] == '2012']
sizes1 = [len(one[one['TimePeriod'] == 'Morning']),len(one[one['TimePeriod'] ==␣
 ↪'Noon']),len(one[one['TimePeriod'] == 'Evening']),len(one[one['TimePeriod']␣
 ↪== 'Night'])]
sizes2 = [len(two[two['TimePeriod'] == 'Morning']),len(two[two['TimePeriod'] ==␣
 ↪'Noon']),len(two[two['TimePeriod'] == 'Evening']),len(two[two['TimePeriod']␣
 ↪== 'Night'])]
sizes3 = [len(three[three['TimePeriod'] ==␣
 ↪'Morning']),len(three[three['TimePeriod'] ==␣
 ↪'Noon']),len(three[three['TimePeriod'] ==␣
 ↪'Evening']),len(three[three['TimePeriod'] == 'Night'])]
```

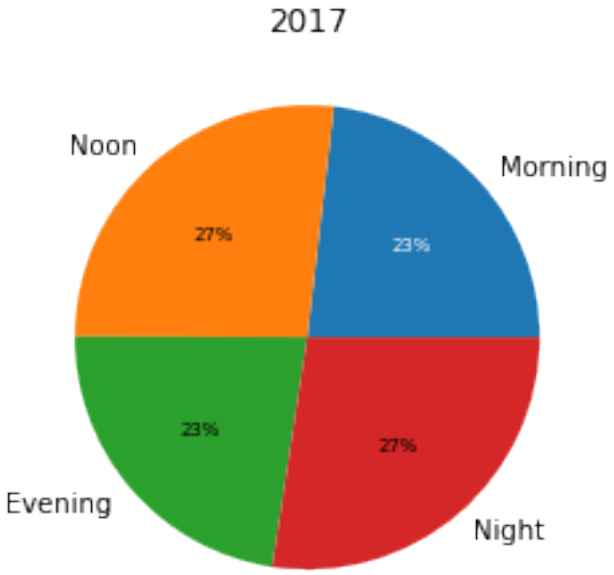
```
sizes4 = [len(four[four['TimePeriod'] ==␣
 ↪'Morning']),len(four[four['TimePeriod'] ==␣
 ↪'Noon']),len(four[four['TimePeriod'] ==␣
 ↪'Evening']),len(four[four['TimePeriod'] == 'Night'])]
# Make figure and axes
fig, axs = plt.subplots(2, 2)

# A standard pie plot
axs[0, 0].pie(sizes1, labels=labels, autopct='%.0f%%', shadow=True)
axs[0,0].set_title("2009")


# Shift the second slice using explode
axs[0, 1].pie(sizes2, labels=labels, autopct='%.0f%%', shadow=True,
              )
axs[0,1].set_title("2010")

# Adapt radius and text size for a smaller pie
patches, texts, autotexts = axs[1, 0].pie(sizes3, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=True )
axs[1,0].set_title("2011")
# Make percent texts even smaller
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

# Use a smaller explode and turn of the shadow for better visibility
patches, texts, autotexts = axs[1, 1].pie(sizes4, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=False
                                         )
axs[1,1].set_title("2012")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()
```



[47]:
```
#Number of crimes committed in 2013 - 2017

labels = ['Morning', 'Noon', 'Evening', 'Night']
one = finaldatasets[finaldatasets['year'] == '2013']
two = finaldatasets[finaldatasets['year'] == '2014']
three = finaldatasets[finaldatasets['year'] == '2015']
four = finaldatasets[finaldatasets['year'] == '2016']
five = finaldatasets[finaldatasets['year'] == '2017']
sizes1 = [len(one[one['TimePeriod'] == 'Morning']),len(one[one['TimePeriod'] ==␣
 ↪'Noon']),len(one[one['TimePeriod'] == 'Evening']),len(one[one['TimePeriod']␣
 ↪== 'Night'])]
sizes2 = [len(two[two['TimePeriod'] == 'Morning']),len(two[two['TimePeriod'] ==␣
 ↪'Noon']),len(two[two['TimePeriod'] == 'Evening']),len(two[two['TimePeriod']␣
 ↪== 'Night'])]
sizes3 = [len(three[three['TimePeriod'] ==␣
 ↪'Morning']),len(three[three['TimePeriod'] ==␣
 ↪'Noon']),len(three[three['TimePeriod'] ==␣
 ↪'Evening']),len(three[three['TimePeriod'] == 'Night'])]
sizes4 = [len(four[four['TimePeriod'] ==␣
 ↪'Morning']),len(four[four['TimePeriod'] ==␣
 ↪'Noon']),len(four[four['TimePeriod'] ==␣
 ↪'Evening']),len(four[four['TimePeriod'] == 'Night'])]
sizes5 = [len(five[five['TimePeriod'] ==␣
 ↪'Morning']),len(five[five['TimePeriod'] ==␣
 ↪'Noon']),len(five[five['TimePeriod'] ==␣
 ↪'Evening']),len(five[five['TimePeriod'] == 'Night'])]
```

```
# Make figure and axes
fig, axs = plt.subplots(2, 2)

# A standard pie plot
axs[0, 0].pie(sizes1, labels=labels, autopct='%.0f%%', shadow=True)
axs[0,0].set_title("2013")


# Shift the second slice using explode
axs[0, 1].pie(sizes2, labels=labels, autopct='%.0f%%', shadow=True,
              )
axs[0,1].set_title("2014")

# Adapt radius and text size for a smaller pie
patches, texts, autotexts = axs[1, 0].pie(sizes3, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=True )
axs[1,0].set_title("2015")
# Make percent texts even smaller
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

# Use a smaller explode and turn of the shadow for better visibility
patches, texts, autotexts = axs[1, 1].pie(sizes4, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=False
                                         )
axs[1,1].set_title("2016")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()

fig1, axs = plt.subplots()

# A standard pie plot
patches, texts, autotexts = axs.pie(sizes5, labels=labels,
                                          autopct='%.0f%%',
                                          shadow=False
                                         )
axs.set_title("2017")
plt.setp(autotexts, size='x-small')
autotexts[0].set_color('white')

plt.show()
```





[48]:
```
#Grouped Bar Graphs
finaldatasets = finaldatasets[finaldatasets['Major Section'] != 0.0]
finaldatasets = finaldatasets[finaldatasets['Community Area'].isna() != True]
```

[49]:
```
finaldatasets['Major Section'].unique()
```

[49]:
```
array(['Far Southeast side', 'Southwest side', 'North side',
       'Northwest side', 'West side', 'Far Southwest side', 'Central',
       'Far North side', 'South side'], dtype=object)
```

[50]:
```
finaldatasets['Primary Crime Type'].unique()
```

[50]:
```
array(['THEFT', 'OFFENSE', 'ASSAULT', 'OTHER', 'VIOLATION',
       'WHITE COLLAR CRIME', 'DAMAGES', 'MURDER', 'KIDNAPPING'],
      dtype=object)
```

[51]:
```
# finaldatasets = finaldatasets.sample(n=200)
```

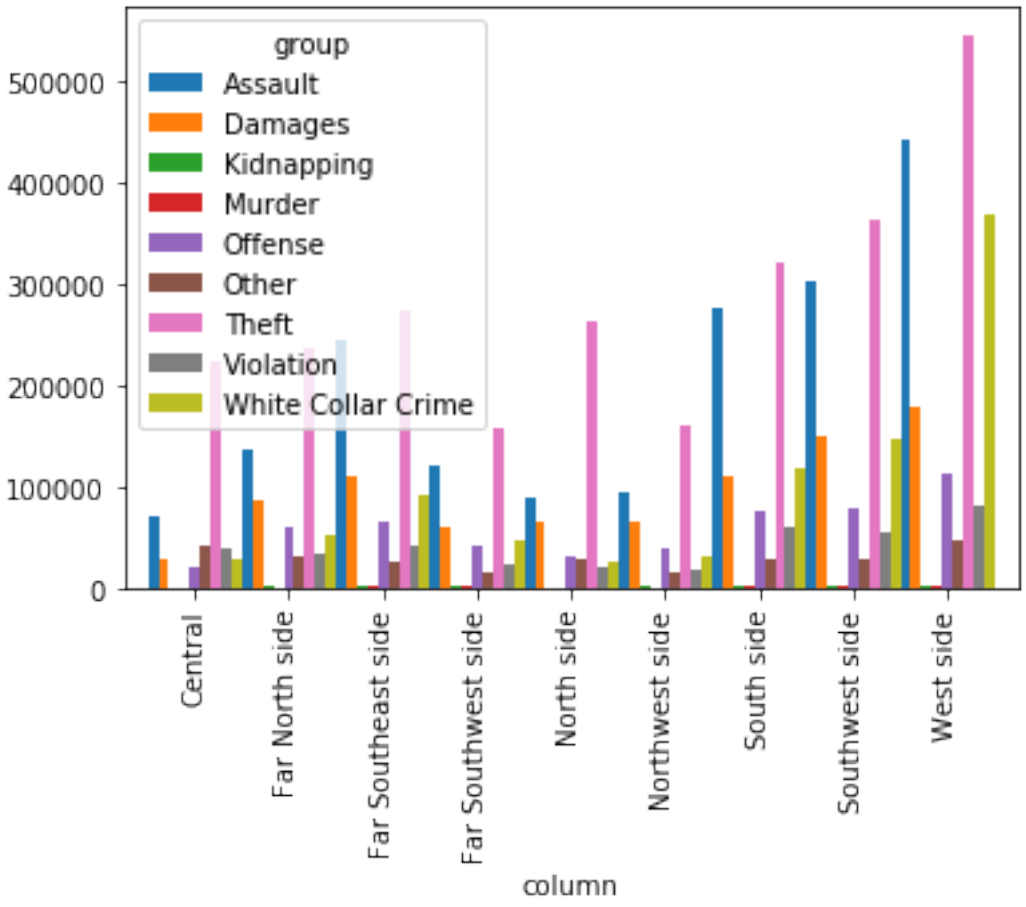
[52]:
```
fss = finaldatasets[finaldatasets['Major Section'] == 'Far Southeast side']
sws = finaldatasets[finaldatasets['Major Section'] == 'Southwest side']
ns = finaldatasets[finaldatasets['Major Section'] == 'North side']
nws = finaldatasets[finaldatasets['Major Section'] == 'Northwest side']
ws = finaldatasets[finaldatasets['Major Section'] == 'West side']
fsws = finaldatasets[finaldatasets['Major Section'] == 'Far Southwest side']
central = finaldatasets[finaldatasets['Major Section'] == 'Central']
fns = finaldatasets[finaldatasets['Major Section'] == 'Far North side']
ss = finaldatasets[finaldatasets['Major Section'] == 'South side']
```

[53]:
```
fss.reset_index()
sws.reset_index()
nws.reset_index()
df = pd.DataFrame(
    [['Theft','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'THEFT'])],
     ['Offense','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'OFFENSE'])],
     ['Assault','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'ASSAULT'])],
     ['Other','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'OTHER'])],
     ['Violation','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'VIOLATION'])],
     ['White Collar Crime','Far Southeast side',len(fss[fss['Primary Crime␣
↪Type'] == 'WHITE COLLAR CRIME'])],
     ['Damages','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'DAMAGES'])],
     ['Murder','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'MURDER'])],
     ['Kidnapping','Far Southeast side',len(fss[fss['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','Southwest side',len(sws[sws['Primary Crime Type'] == 'THEFT'])],
     ['Offense','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'OFFENSE'])],
     ['Assault','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'ASSAULT'])],
```

```
    ['Other','Southwest side',len(sws[sws['Primary Crime Type'] == 'OTHER'])],
    ['Violation','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','Southwest side',len(sws[sws['Primary Crime Type']␣
↪== 'WHITE COLLAR CRIME'])],
    ['Damages','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'DAMAGES'])],
    ['Murder','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'MURDER'])],
    ['Kidnapping','Southwest side',len(sws[sws['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','Northwest side',len(nws[nws['Primary Crime Type'] == 'THEFT'])],
    ['Offense','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'OFFENSE'])],
    ['Assault','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'ASSAULT'])],
    ['Other','Northwest side',len(nws[nws['Primary Crime Type'] == 'OTHER'])],
    ['Violation','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','Northwest side',len(nws[nws['Primary Crime Type']␣
↪== 'WHITE COLLAR CRIME'])],
    ['Damages','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'DAMAGES'])],
    ['Murder','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'MURDER'])],
    ['Kidnapping','Northwest side',len(nws[nws['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','North side',len(ns[ns['Primary Crime Type'] == 'THEFT'])],
    ['Offense','North side',len(ns[ns['Primary Crime Type'] == 'OFFENSE'])],
    ['Assault','North side',len(ns[ns['Primary Crime Type'] == 'ASSAULT'])],
    ['Other','North side',len(ns[ns['Primary Crime Type'] == 'OTHER'])],
    ['Violation','North side',len(ns[ns['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','North side',len(ns[ns['Primary Crime Type'] ==␣
↪'WHITE COLLAR CRIME'])],
    ['Damages','North side',len(ns[ns['Primary Crime Type'] == 'DAMAGES'])],
    ['Murder','North side',len(ns[ns['Primary Crime Type'] == 'MURDER'])],
    ['Kidnapping','North side',len(ns[ns['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','West side',len(ws[ws['Primary Crime Type'] == 'THEFT'])],
    ['Offense','West side',len(ws[ws['Primary Crime Type'] == 'OFFENSE'])],
    ['Assault','West side',len(ws[ws['Primary Crime Type'] == 'ASSAULT'])],
    ['Other','West side',len(ws[ws['Primary Crime Type'] == 'OTHER'])],
    ['Violation','West side',len(ws[ws['Primary Crime Type'] == 'VIOLATION'])],
    ['White Collar Crime','West side',len(ws[ws['Primary Crime Type'] ==␣
↪'WHITE COLLAR CRIME'])],
```

```
    ['Damages','West side',len(ws[ws['Primary Crime Type'] == 'DAMAGES'])],
    ['Murder','West side',len(ws[ws['Primary Crime Type'] == 'MURDER'])],
    ['Kidnapping','West side',len(ws[ws['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'THEFT'])],
    ['Offense','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'OFFENSE'])],
    ['Assault','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'ASSAULT'])],
    ['Other','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'OTHER'])],
    ['Violation','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','Far Southwest side',len(fsws[fsws['Primary Crime␣
↪Type'] == 'WHITE COLLAR CRIME'])],
    ['Damages','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'DAMAGES'])],
    ['Murder','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'MURDER'])],
    ['Kidnapping','Far Southwest side',len(fsws[fsws['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','Central',len(central[central['Primary Crime Type'] == 'THEFT'])],
    ['Offense','Central',len(central[central['Primary Crime Type'] ==␣
↪'OFFENSE'])],
    ['Assault','Central',len(central[central['Primary Crime Type'] ==␣
↪'ASSAULT'])],
    ['Other','Central',len(central[central['Primary Crime Type'] == 'OTHER'])],
    ['Violation','Central',len(central[central['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','Central',len(central[central['Primary Crime Type']␣
↪== 'WHITE COLLAR CRIME'])],
    ['Damages','Central',len(central[central['Primary Crime Type'] ==␣
↪'DAMAGES'])],
    ['Murder','Central',len(central[central['Primary Crime Type'] ==␣
↪'MURDER'])],
    ['Kidnapping','Central',len(central[central['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','Far North side',len(fns[fns['Primary Crime Type'] == 'THEFT'])],
    ['Offense','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'OFFENSE'])],
    ['Assault','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'ASSAULT'])],
    ['Other','Far North side',len(fns[fns['Primary Crime Type'] == 'OTHER'])],
    ['Violation','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'VIOLATION'])],
```

```
    ['White Collar Crime','Far North side',len(fns[fns['Primary Crime Type']␣
↪== 'WHITE COLLAR CRIME'])],
    ['Damages','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'DAMAGES'])],
    ['Murder','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'MURDER'])],
    ['Kidnapping','Far North side',len(fns[fns['Primary Crime Type'] ==␣
↪'KIDNAPPING'])],
    ['Theft','South side',len(ss[ss['Primary Crime Type'] == 'THEFT'])],
    ['Offense','South side',len(ss[ss['Primary Crime Type'] == 'OFFENSE'])],
    ['Assault','South side',len(ss[ss['Primary Crime Type'] == 'ASSAULT'])],
    ['Other','South side',len(ss[ss['Primary Crime Type'] == 'OTHER'])],
    ['Violation','South side',len(ss[ss['Primary Crime Type'] ==␣
↪'VIOLATION'])],
    ['White Collar Crime','South side',len(ss[ss['Primary Crime Type'] ==␣
↪'WHITE COLLAR CRIME'])],
    ['Damages','South side',len(ss[ss['Primary Crime Type'] == 'DAMAGES'])],
    ['Murder','South side',len(ss[ss['Primary Crime Type'] == 'MURDER'])],
    ['Kidnapping','South side',len(ss[ss['Primary Crime Type'] ==␣
↪'KIDNAPPING'])]
   ],

   columns=['group','column','val'])

df.reset_index().pivot("column", "group", "val").plot(kind='bar', width=1.0)
# plt.rcParams['figure.figsize'] = (50,50)
plt.figure(figsize=(20, 3))
plt.show()
```




```
<Figure size 1440x216 with 0 Axes>
```

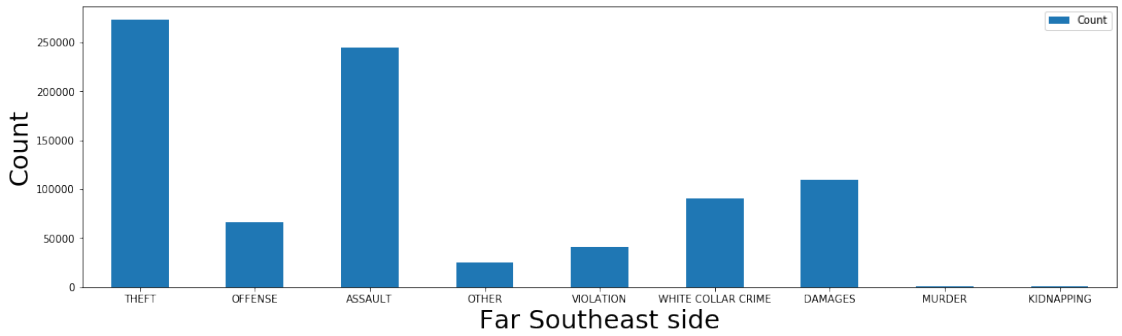
[54]:
```
uniq_crimes = finaldatasets['Primary Crime Type'].unique()
list(uniq_crimes)
```

[54]:
```
['THEFT',
 'OFFENSE',
 'ASSAULT',
 'OTHER',
 'VIOLATION',
 'WHITE COLLAR CRIME',
 'DAMAGES',
 'MURDER',
 'KIDNAPPING']
```

[55]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(fss[fss['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
```

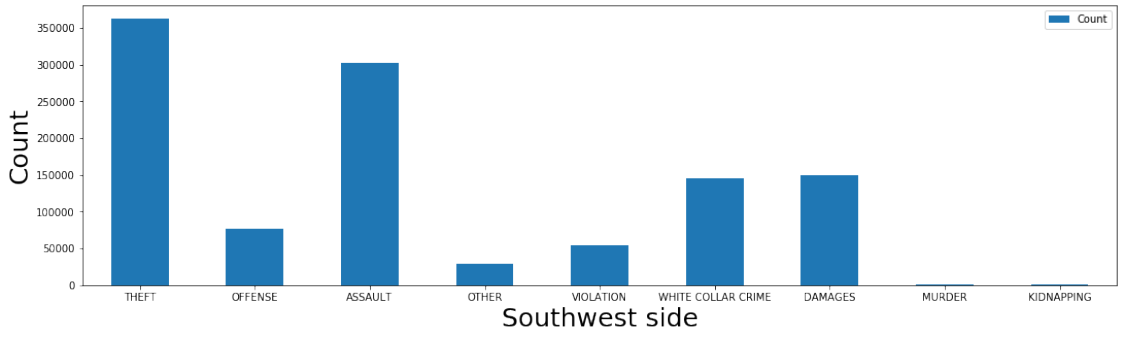
```
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Far Southeast side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[55]: `Text(0, 0.5, 'Count')`



[56]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(sws[sws['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Southwest side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

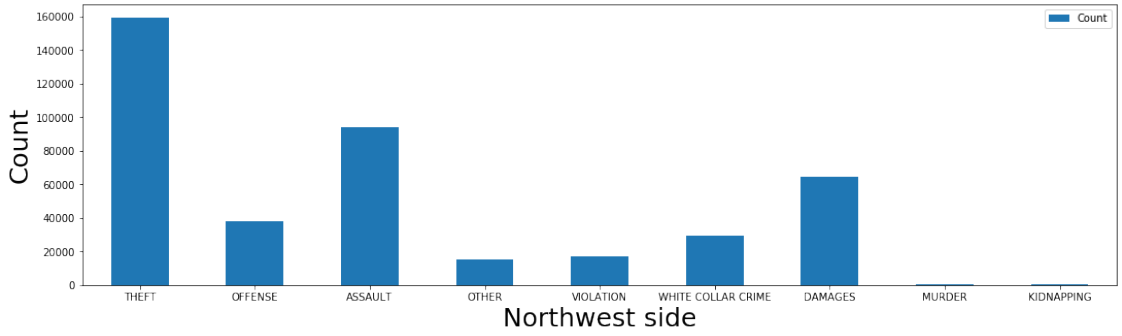
[56]: `Text(0, 0.5, 'Count')`



[57]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(nws[nws['Primary Crime Type'] == crime]))
```

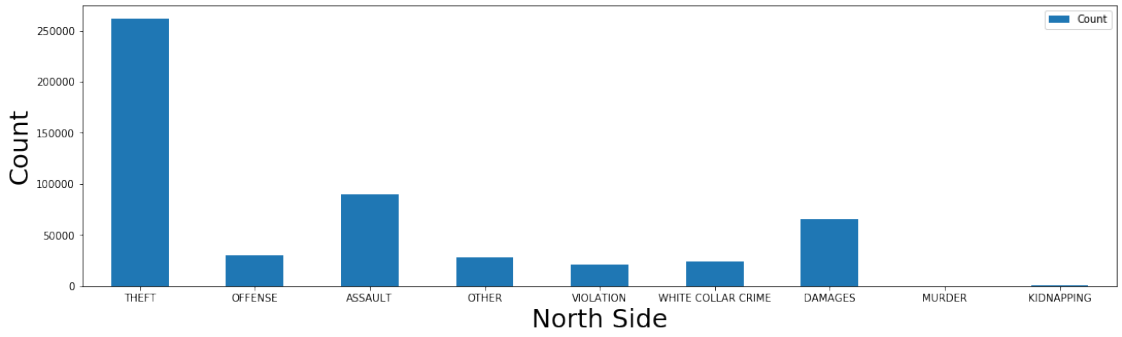
```
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Northwest side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[57]: `Text(0, 0.5, 'Count')`



[58]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(ns[ns['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('North Side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

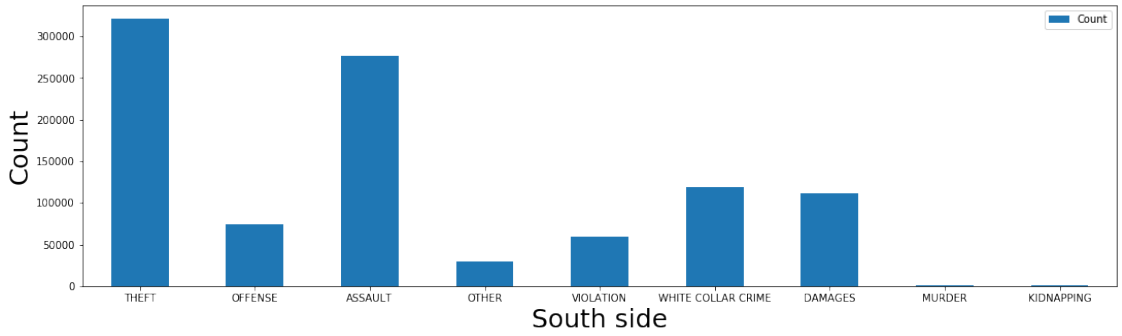
[58]: `Text(0, 0.5, 'Count')`



[59]:
```
vals = []
for crime in list(uniq_crimes):
```

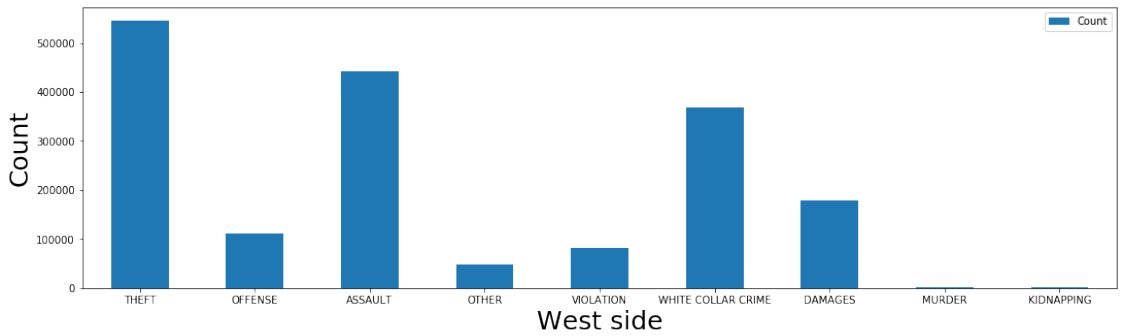
```
    vals.append(len(ss[ss['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('South side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[59]: `Text(0, 0.5, 'Count')`



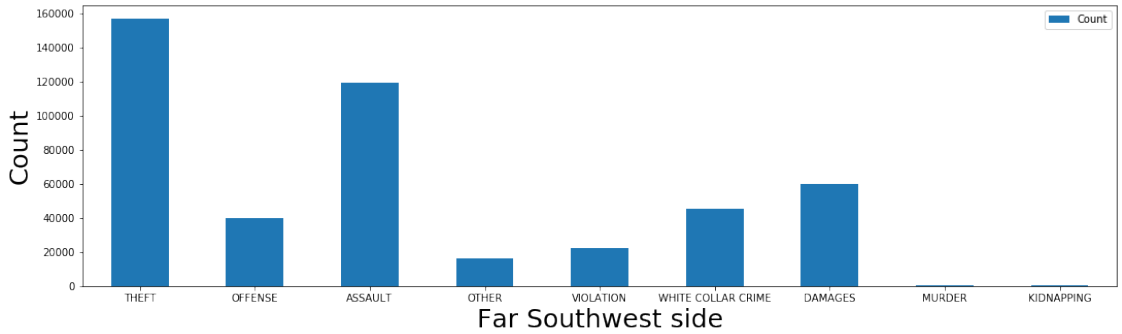
[60]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(ws[ws['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('West side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[60]: `Text(0, 0.5, 'Count')`



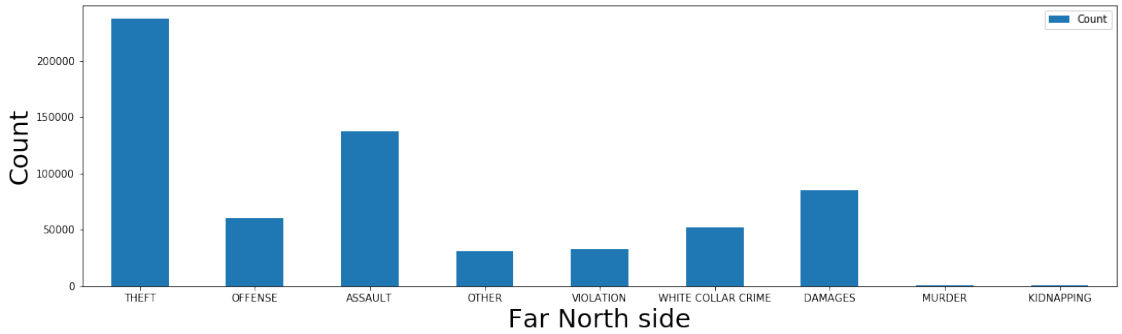
[61]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(fsws[fsws['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Far Southwest side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[61]:
```
Text(0, 0.5, 'Count')
```



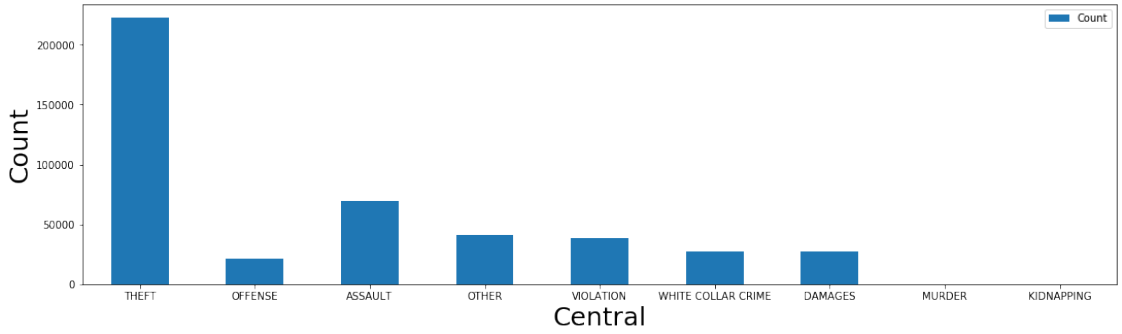
[62]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(fns[fns['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Far North side', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[62]:
```
Text(0, 0.5, 'Count')
```



[63]:
```
vals = []
for crime in list(uniq_crimes):
    vals.append(len(central[central['Primary Crime Type'] == crime]))
df = pd.DataFrame({'lab':list(uniq_crimes), 'Count':vals})
ax = df.plot.bar(x='lab', y='Count', rot=0, figsize = (18,5))
# plt.subplots(figsize=(18,5))
fig.suptitle('test title', fontsize=200)
plt.xlabel('Central', fontsize=25)
plt.ylabel('Count', fontsize=25)
```

[63]:
```
Text(0, 0.5, 'Count')
```



[64]:
```
def toString(x):
    return str(int(x))

df_offense = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'OFFENSE']
df_offense.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_offense = df_offense[keep_cols].reset_index()

df_offense_allyears = df_offense.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_offense_allyears['Community Area'] = df_offense['Community Area'].
 ↪apply(toString)
# ____________________________________________________ #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
```

```
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_offense_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Offense per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map1.html")
IFrame('map1.html', width=900, height=700)
```

```
C:\Users\ibray\Anaconda3\lib\site-packages\folium\folium.py:415: FutureWarning:

The choropleth  method has been deprecated. Instead use the new Choropleth
class, which has the same arguments. See the example notebook
'GeoJSON_and_choropleth' for how to do this.
```

[64]:
```
<IPython.lib.display.IFrame at 0x23288393128>
```

[65]:
```
df_theft = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'THEFT']
df_theft.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_theft = df_theft[keep_cols].reset_index()

df_theft_allyears = df_theft.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_theft_allyears['Community Area'] = df_theft['Community Area'].apply(toString)
# ------------------------------------------------------ #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
```

```
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_theft_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Theft per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map2.html")
IFrame('map2.html', width=900, height=700)
```

[65]: <IPython.lib.display.IFrame at 0x233141012b0>

[66]:
```
df_assault = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'ASSAULT']
df_assault.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_assault = df_assault[keep_cols].reset_index()

df_assault_allyears = df_assault.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_assault_allyears['Community Area'] = df_assault['Community Area'].
 ↪apply(toString)
# ------------------------------------------------------ #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)
```

```
m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_assault_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Assault per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map3.html")
IFrame('map3.html', width=900, height=700)
```

[66]: <IPython.lib.display.IFrame at 0x23314119160>

[67]:
```
df_violation = finaldatasets.loc[finaldatasets['Primary Crime Type'] ==␣
 ↪'VIOLATION']
df_violation.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_violation = df_violation[keep_cols].reset_index()

df_violation_allyears = df_violation.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_violation_allyears['Community Area'] = df_violation['Community Area'].
 ↪apply(toString)
# ------------------------------------------------------- #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
```

```
    data=df_violation_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Violation per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map4.html")
IFrame('map4.html', width=900, height=700)
```

[67]:
```
<IPython.lib.display.IFrame at 0x232883c4cf8>
```

[68]:
```
df_white_collar_crime = finaldatasets.loc[finaldatasets['Primary Crime Type']␣
 ↪== 'WHITE COLLAR CRIME']
df_white_collar_crime.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_white_collar_crime = df_white_collar_crime[keep_cols].reset_index()

df_white_collar_crime_allyears = df_white_collar_crime.groupby(['Community␣
 ↪Area']).count().Arrest.reset_index()
df_white_collar_crime_allyears['Community Area'] =␣
 ↪df_white_collar_crime['Community Area'].apply(toString)
# ------------------------------------------------------- #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_white_collar_crime_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
```

```
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of White Collar Crime per Community Area:␣
 ↪2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map5.html")
IFrame('map5.html', width=900, height=700)
```

[68]:
```
<IPython.lib.display.IFrame at 0x2331056eda0>
```

[69]:
```
df_damages = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'DAMAGES']
df_damages.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_damages = df_damages[keep_cols].reset_index()

df_damages_allyears = df_damages.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_damages_allyears['Community Area'] = df_damages['Community Area'].
 ↪apply(toString)
# ------------------------------------------------------- #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_damages_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
```

```
    legend_name='Choropleth of Damages per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map6.html")
IFrame('map6.html', width=900, height=700)
```

[69]: <IPython.lib.display.IFrame at 0x232884d8f98>

[70]:
```
df_murder = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'MURDER']
df_murder.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_murder = df_murder[keep_cols].reset_index()

df_murder_allyears = df_murder.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_murder_allyears['Community Area'] = df_murder['Community Area'].
 ↪apply(toString)
# ------------------------------------------------------- #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_murder_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Murder per Community Area: 2001-2017',
    highlight=True
)
```

```
folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map7.html")
IFrame('map7.html', width=900, height=700)
```

[70]: <IPython.lib.display.IFrame at 0x23310763080>

[71]:
```
df_kidnapping = finaldatasets.loc[finaldatasets['Primary Crime Type'] ==␣
 ↪'KIDNAPPING']
df_kidnapping.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_kidnapping = df_kidnapping[keep_cols].reset_index()

df_kidnapping_allyears = df_kidnapping.groupby(['Community Area']).count().
 ↪Arrest.reset_index()
df_kidnapping_allyears['Community Area'] = df_kidnapping['Community Area'].
 ↪apply(toString)
# ------------------------------------------------------ #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_kidnapping_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Kidnapping per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
```

```
folium.LayerControl().add_to(m)
m.save("map8.html")
IFrame('map8.html', width=900, height=700)
```

[71]: <IPython.lib.display.IFrame at 0x2331056ed68>

[72]:
```
df_other = finaldatasets.loc[finaldatasets['Primary Crime Type'] == 'OTHER']
df_other.dropna()

keep_cols = ['Date','Primary Type','Location␣
 ↪Description','Arrest','District','Community Area','Year']
df_other = df_other[keep_cols].reset_index()

df_other_allyears = df_other.groupby(['Community Area']).count().Arrest.
 ↪reset_index()
df_other_allyears['Community Area'] = df_other['Community Area'].apply(toString)
# ---------------------------------------------------------- #
chicago = location=[41.85, -87.68]

m = folium.Map(chicago, zoom_start=10)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(m)

m.choropleth(
    geo_data='community_areas.geojson',
    name='choropleth',
    data=df_other_allyears,
    columns=['Community Area', 'Arrest'],
    key_on='feature.properties.area_numbe',
    fill_color='YlOrRd',
    fill_opacity=0.4,
    line_opacity=0.85,
    legend_name='Choropleth of Other per Community Area: 2001-2017',
    highlight=True
)

folium.TileLayer('openstreetmap').add_to(m)
folium.TileLayer('cartodbpositron').add_to(m)
#folium.TileLayer('major_section_1').add_to(m)
folium.LayerControl().add_to(m)
m.save("map9.html")
IFrame('map9.html', width=900, height=700)
```

[72]: <IPython.lib.display.IFrame at 0x23314119438>

[73]:
```
#Classification
```

[74]:
```
#Finding Correlation

corr_matrix = finaldatasets.corr().abs()
corr_matrix.head()
# x_train, x_test, y_train, y_test = train_test_split(finaldatasets[['Total␣
 ↪Population','Percent White, not Hispanic or Latino','Percent Black, not␣
 ↪Hispanic or Latino','Percent Hispanic or Latino','Percent Foreign␣
 ↪Born','Percent Female','Percent Age 29 and Under','Percent Age 65 and␣
 ↪Older','Median Household Income','Percent Unemployed','Percent Less than␣
 ↪High School Degree','Percent Rural','Percent Less than Bachelor\'s␣
 ↪Degree']], df['Party'], random_state = 0, test_size = 0.25)
```

[74]:

	District	Community Area	Year
District	1.00000	0.498560	0.001870
Community Area	0.49856	1.000000	0.000188
Year	0.00187	0.000188	1.000000

[75]:
```
#[Date, Year, TimePeriod, Primary Crime Type, Arrest, Community Area, District,␣
 ↪Major Section, Loc Type]
finaldatasets = finaldatasets[finaldatasets['Loc Type'] != '']
```

[76]:
```
finaldatasets['Loc Type N'] = finaldatasets['Loc Type'].replace({'RESIDENTIAL':
 ↪1,'OTHER':2,'COMMERCIAL':3,'PUBLIC':4,'HEALTHCARE FACILITY':5,'PLACE OF␣
 ↪WORSHIP':6,'EDUCATION':7,'PRIVATE':8,'GOVERNMENT':9})
```

[77]:
```
finaldatasets['Major Section N'] = finaldatasets['Major Section'].replace({'Far␣
 ↪Southeast side': 1,
 'Southwest side': 2,
 'North side': 3,
 'Northwest side': 4,
 'West side': 5,
 'Far Southwest side': 6,
 'Central': 7,
 'Far North side': 8,
 'South side': 9})
```

[78]:
```
finaldatasets['Primary Crime Type N'] = finaldatasets['Primary Crime Type'].
 ↪replace({'THEFT': 1,
 'OFFENSE': 2,
 'ASSAULT': 3,
 'OTHER': 4,
 'VIOLATION': 5,
 'WHITE COLLAR CRIME': 6,
 'DAMAGES': 7,
 'MURDER': 8,
 'KIDNAPPING': 9})
```

[79]:
```
finaldatasets['Arrest N'] = np.where(finaldatasets['Arrest']=='True', 1, 0)
```

[80]:
```
finaldatasets['TimePeriod N'] = finaldatasets['TimePeriod'].replace({'Night':
↪1, 'Morning': 2, 'Evening': 3, 'Noon': 4})
```

[81]:
```
finaldatasets.columns
```

[81]:
```
Index(['Date', 'Primary Type', 'Location Description', 'Arrest', 'District',
       'Community Area', 'Year', 'date_id1', 'date_id2', 'day', 'month',
       'year', 'Seasons', 'hour', 'minute', 'second', 'dayperiod',
       'TimePeriod', 'Major Section', 'Primary Crime Type', 'Loc Type',
       'Loc Type N', 'Major Section N', 'Primary Crime Type N', 'Arrest N',
       'TimePeriod N'],
      dtype='object')
```

[82]:
```
corr_matrix = finaldatasets.corr().abs()
corr_matrix
```

[82]:

	District	Community Area	Year	Loc Type N	\
District	1.000000	0.498559	0.002015	0.018129	
Community Area	0.498559	1.000000	0.000415	0.037661	
Year	0.002015	0.000415	1.000000	0.016252	
Loc Type N	0.018129	0.037661	0.016252	1.000000	
Major Section N	0.054199	0.233335	0.004490	0.015153	
Primary Crime Type N	0.018806	0.030205	0.028458	0.066556	
Arrest N	0.001308	0.001390	0.001277	0.012249	
TimePeriod N	0.027981	0.008540	0.016699	0.037195	

	Major Section N	Primary Crime Type N	Arrest N	\
District	0.054199	0.018806	0.001308	
Community Area	0.233335	0.030205	0.001390	
Year	0.004490	0.028458	0.001277	
Loc Type N	0.015153	0.066556	0.012249	
Major Section N	1.000000	0.014653	0.000485	
Primary Crime Type N	0.014653	1.000000	0.023662	
Arrest N	0.000485	0.023662	1.000000	
TimePeriod N	0.013950	0.046746	0.001658	

	TimePeriod N
District	0.027981
Community Area	0.008540
Year	0.016699
Loc Type N	0.037195
Major Section N	0.013950
Primary Crime Type N	0.046746
Arrest N	0.001658
TimePeriod N	1.000000

[83]:
```
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
↪bool))
upper.head()
```

[83]:
```
                 District  Community Area      Year  Loc Type N  \
District              NaN        0.498559  0.002015    0.018129
Community Area        NaN             NaN  0.000415    0.037661
Year                  NaN             NaN       NaN    0.016252
Loc Type N            NaN             NaN       NaN         NaN
Major Section N       NaN             NaN       NaN         NaN

                 Major Section N  Primary Crime Type N  Arrest N  TimePeriod N
District                0.054199              0.018806  0.001308      0.027981
Community Area          0.233335              0.030205  0.001390      0.008540
Year                    0.004490              0.028458  0.001277      0.016699
Loc Type N              0.015153              0.066556  0.012249      0.037195
Major Section N              NaN              0.014653  0.000485      0.013950
```

[84]:
```
threshold = 0.9
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
```

[85]:
```
to_drop
```

[85]:
```
[]
```

[86]:
```
finaldatasets = finaldatasets.iloc[::4,:]
finaldatasets = finaldatasets.iloc[::4,:]
# finaldatasets = finaldatasets.iloc[::4,:]
```

[87]:
```
# finaldatasets = finaldatasets.iloc[::4,:]
finaldatasets.shape
```

[87]:
```
(452319, 26)
```

[88]:
```
#[Date, Year, TimePeriod, Primary Crime Type, Arrest, Community Area, District,␣
 ↪Major Section, Loc Type]
x_train, x_test, y_train, y_test = train_test_split(finaldatasets[['TimePeriod␣
 ↪N','Major Section N','Loc Type N','Arrest N']], finaldatasets['Primary Crime␣
 ↪Type N'], random_state = 0, test_size = 0.25)
x_test
```

[88]:
```
         TimePeriod N  Major Section N  Loc Type N  Arrest N
1154047             3                9           4         0
1216341             1                2           4         0
117905              1                2           1         0
25953               3                9           1         0
791167              3                2           2         0
2552373             3                5           4         0
1866074             1                5           4         0
1032398             1                5           4         0
1253970             2                8           1         0
558860              1                2           1         0
1415406             2                6           1         0
921521              3                4           4         0
354148              1                9           1         0
```

2222603	3	8	4	0
276290	3	2	1	0
759878	1	7	4	0
2445611	4	2	4	0
1134508	3	4	9	0
328988	4	2	1	0
1793273	4	5	3	0
1069211	4	5	1	0
1302825	4	5	1	0
2683710	1	4	1	0
371900	4	1	1	0
81869	1	9	8	0
1276034	1	5	1	0
1089700	3	1	3	0
2175970	3	3	4	0
1685625	3	4	1	0
1042445	4	1	7	0
...	...	...	...	...
1509950	1	9	4	0
391830	1	1	1	0
722179	1	5	1	0
776695	3	3	9	0
1203863	1	9	1	0
1437198	3	5	4	0
192843	1	5	4	0
822960	1	3	4	0
1373608	2	5	4	0
342160	2	9	4	0
585137	3	1	4	0
1194121	3	4	1	0
43470	4	4	3	0
71211	3	4	1	0
2663747	4	5	4	0
248678	2	1	4	0
285670	4	8	4	0
814188	4	8	4	0
2027478	3	2	4	0
1073422	4	9	2	0
1269154	4	5	8	0
396500	4	2	4	0
872847	3	5	4	0
700574	2	9	7	0
1488874	4	5	7	0
788703	4	1	3	0
1760948	4	9	1	0
451209	4	3	8	0
2548416	2	9	1	0

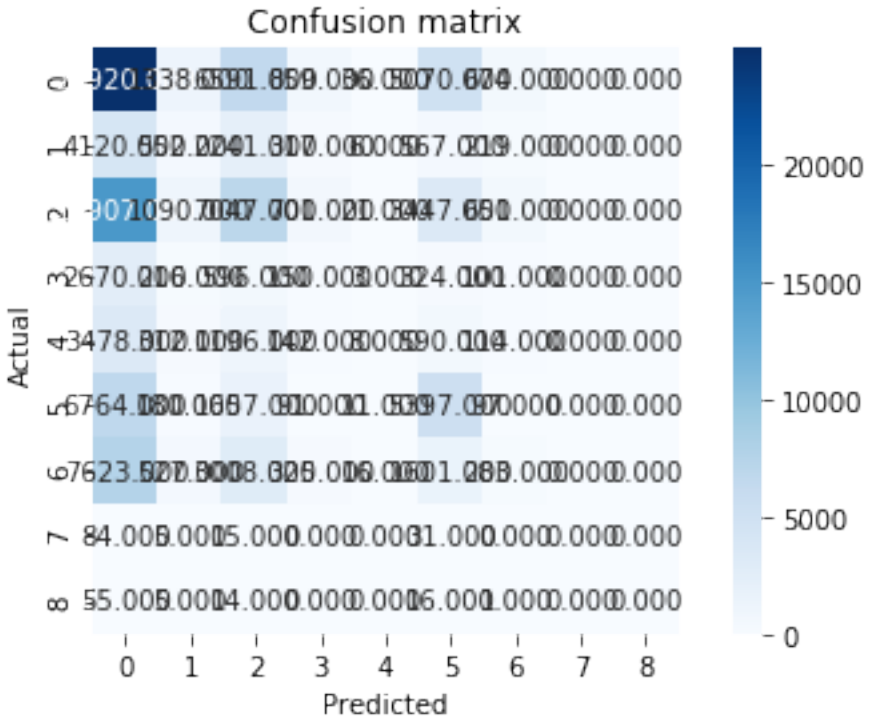
```
130132              4                1           3          0

[113080 rows x 4 columns]
```

[91]:
```
classifier = KNeighborsClassifier(n_neighbors = 3)
classifier.fit(x_train, y_train)
```

[91]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

[92]:
```
y_pred = classifier.predict(x_test)
```

[93]:
```
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
 ↪cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion matrix')
plt.rcParams['figure.figsize'] = 15, 15
plt.tight_layout()
```



[94]:
```
accuracy = metrics.accuracy_score(y_test, y_pred)
error = 1 - metrics.accuracy_score(y_test, y_pred)
precision = metrics.precision_score(y_test, y_pred, average = None)
recall = metrics.recall_score(y_test, y_pred, average = None)
```

```
F1_score = metrics.f1_score(y_test, y_pred, average = None)
print([accuracy, error, precision, recall, F1_score])
```

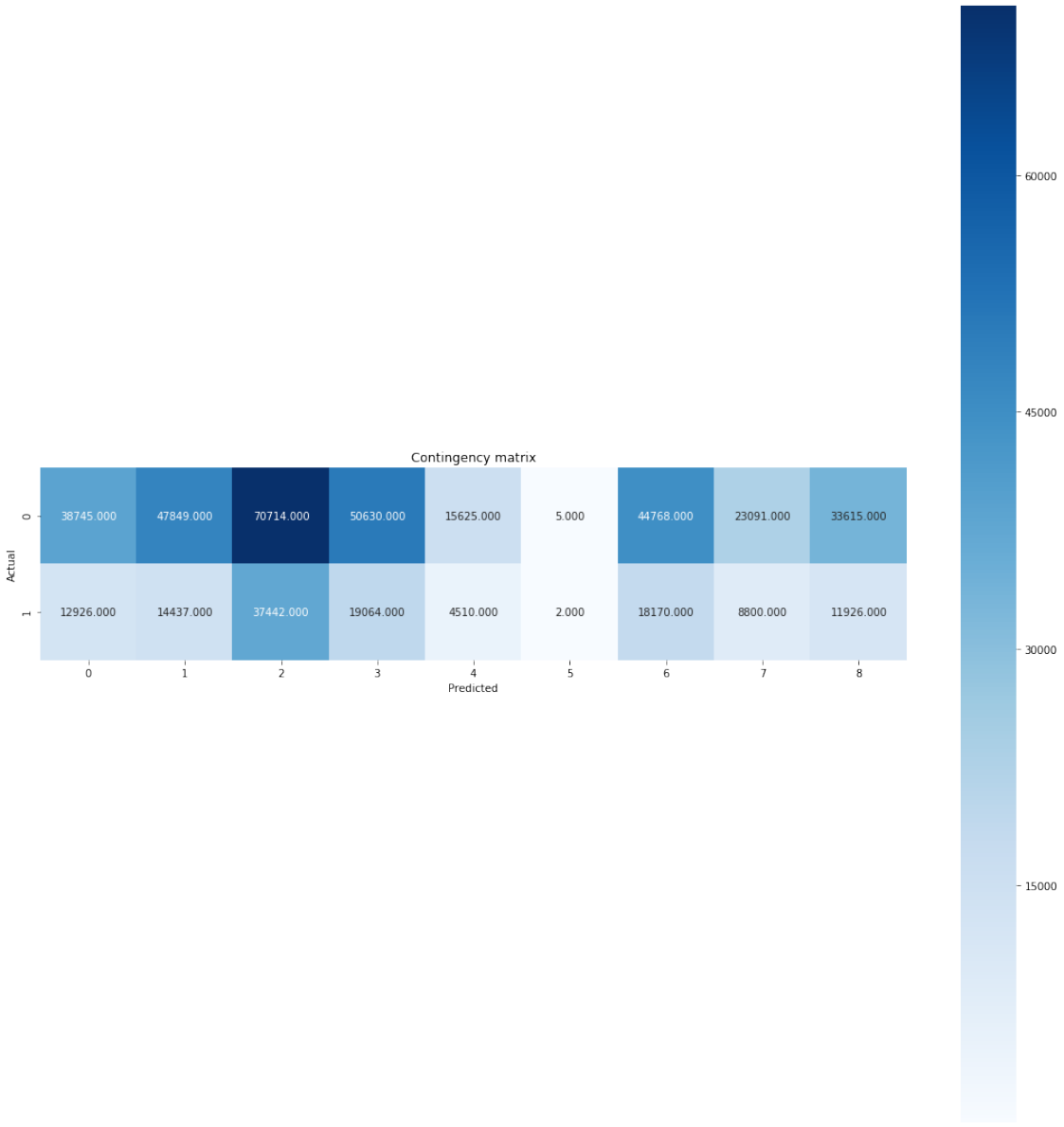
```
[0.3392023346303502, 0.6607976653696498, array([0.38563315, 0.13065089,
0.31509054, 0.05802708, 0.07920792,
       0.3166696 , 0.13224299, 0.        , 0.        ]), array([0.6310778 ,
0.06881077, 0.25290698, 0.03694581, 0.00136986,
       0.38015074, 0.02114623, 0.        , 0.        ]), array([0.47872902,
0.09014453, 0.28059488, 0.04514673, 0.00269315,
       0.34551857, 0.03646202, 0.        , 0.        ])]

C:\Users\ibray\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning:

Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.

C:\Users\ibray\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning:

F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
```

[95]:
```
# clustering begin
```

[96]:
```
finaldatasets['Arrest_N'] = finaldatasets['Arrest'].replace({'True': 1, 'False':
 ↪ 0})
#x_train, x_test, y_train, y_test = train_test_split(finaldatasets[['Year',␣
 ↪'TimePeriod N', 'Primary Crime Type N', 'Community Area', 'District', 'Major␣
 ↪Section N', 'Loc Type N']], finaldatasets['Arrest N'], random_state = 0,␣
 ↪test_size = 0.25, train_size=0.75)
```

[97]:
```
clustering_df = finaldatasets[['Year', 'TimePeriod N', 'Arrest_N','Primary␣
 ↪Crime Type N', 'Community Area', 'District', 'Major Section N', 'Loc Type␣
 ↪N']]
clustering_df.replace([np.inf, -np.inf], np.nan, inplace=True)
clustering_df.fillna(999, inplace=True)

X = clustering_df.drop('Arrest_N', axis=1)
Y = clustering_df['Arrest_N']

scaler = StandardScaler()
scaler.fit(X)
clustering_df_scaled = scaler.transform(X)

clustering = KMeans(n_clusters = 9, init='k-means++', n_init = 10,␣
 ↪random_state=0)
clustering.fit(X) #X
```

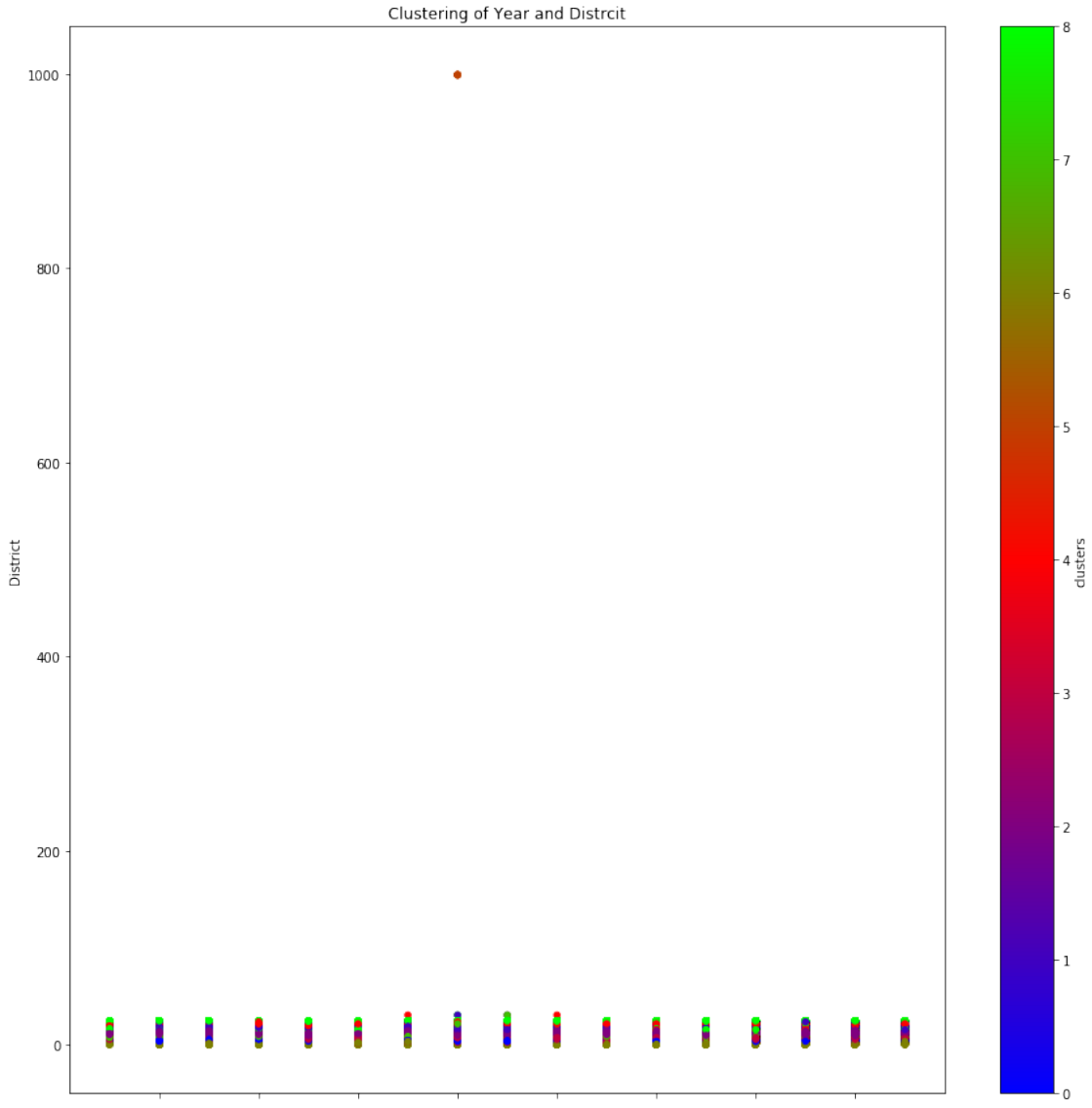
```
clusters = clustering.labels_
#print(clusters.shape)
cont_matrix = metrics.cluster.contingency_matrix(Y, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
 ↪cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()

# X, y_true = make_blobs(n_samples=200, n_features=8, centers=9, cluster_std=0.
 ↪60, random_state=0)

# # Create DataFrame
# df = pd.DataFrame(X, columns=['Year', 'TimePeriod N', 'Primary Crime Type N',␣
 ↪'Arrest N', 'Community Area', 'District', 'Major Section N', 'Loc Type N'])

# # Make k-means clusterer
# clusterer = KMeans(9, random_state=1)

# # Fit clusterer
# clusterer.fit(X)

# # Predict values
# df['group'] = clusterer.predict(X)

# # First few observations
# df         , 'TimePeriod N', 'Primary Crime Type N', 'Community Area',␣
 ↪'District', 'Major Section N', 'Loc Type N']], finaldatasets['Arrest N'
```



[100]:
```
# adjusted_rand_index = metrics.adjusted_rand_score(Y, clusters)
# silhouette_coefficient = metrics.silhouette_score(X, clusters,␣
 ↪metric="euclidean")
# print([adjusted_rand_index, silhouette_coefficient])
```

[99]:
```
clustering_df['clusters'] = clusters
ax = clustering_df.plot(kind = 'scatter', x = 'Year', y = 'District', c =␣
 ↪'clusters', colormap = plt.cm.brg)
ax.set(title = 'Clustering of Year and Distrcit', xlabel = 'Year', ylabel =␣
 ↪'District')
```

[99]:
```
[Text(0, 0.5, 'District'),
 Text(0.5, 0, 'Year'),
 Text(0.5, 1.0, 'Clustering of Year and Distrcit')]
```



[ ]:

[ ]:

[ ]: