Denver Crime Predcition

Business Problem

Crime is a challenging social problem across major cities and metropolitan areas across the United States. When we look at the statistics leading up to 2021 the results are astounding. In Denver, business burglaries are up by 143 percent, carjackings are up by 140 percent and homicides are up 81 percent since 2019. As crime increases, it is important to try to find novel ways to implement crime prevention, and this project's goal is to use that through prediction, focused on neighborhoods and time of day.

This project will be helpful for various different populations. It will be helpful for those that wish to move to the Denver area, as they will be able to determine what area they would like to move to. Also, it will be helpful for locals and tourists alike in the sense that they will be able to determine what places should be avoided or at what time their risk of being a victim of a crime might be higher. Most importantly, I believe that this could help in the allocation of resources whether it would mean a more equal distribution of police resources or better allocation of city funds to put in place preventative measures to make more affected communities safer. The goal is to raise awareness to the increase in crime, and the neighborhoods in Denver that are requiring the most help reducing this issue.

Import Libraries

In this section we will import the libraries that we will be using during the development our data and models.

```
!pip install geopandas
```

```
import os
import glob
import csv
from google.colab import drive
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import graphviz
from numpy import savetxt
from numpy import asarray
import numbers
import folium
from folium.plugins import HeatMap
import datetime
import holidays
from scipy import stats
import descartes
import geopandas as gpd
```

```
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
    Mounted at /content/drive
```

```
print(os.getcwd())

# Get contents of the directory
print(os.listdir())
```

```
    /content
    ['.config', 'combined.csv', '__MACOSX', 'crime_cleaned.csv', 'crime.csv.zip', 'drive', 'crime.csv', 'community_data2.csv', 'samp
```

Data Description

Below are the potential datasets and shx information that we will be working with. The first is crime data from Denver, including its 78 neighborhoods including crime from 2016 to 2021. The next dataset outlines the offense codes used to correlate with the type of crime committed. The next file is a shapefile that will potentially be used to outline Denver's neighborhoods. The final piece of data is the American Community Survey from 2015 to 2019 which highlights the population, dempographics and housing data for Denver's neighborhoods.

```
drive_path = '/content/drive/MyDrive/crime.csv.zip'
drive_2_path = '/content/drive/MyDrive/offense_codes.csv'
drive_3_path = '/content/drive/MyDrive/statistical_neighborhoods.shx'
```

```
drive_4_path = '/content/drive/MyDrive/american_community_survey_nbrhd_2015_2019.csv'
drive_5_path = '/content/drive/MyDrive/police_stations.csv'
drive_6_path = '/content/drive/MyDrive/statistical_neighborhoods.csv'
drive_7_path = '/content/drive/MyDrive/denver_statistical_neighborhoods.geojson'
local_path = '/content'


!cp '{drive_path}' .



os.chdir(local_path)
!unzip -q 'crime.csv.zip'

     replace crime.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: N
```

Data Cleaning

Cleaning the data provided through Denver's Crime Database

```
crime=pd.read_csv('crime.csv')


crime.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 506614 entries, 0 to 506613
     Data columns (total 19 columns):
      #   Column                  Non-Null Count   Dtype
     ---  ------                  --------------   -----
      0   INCIDENT_ID             506614 non-null  int64
      1   OFFENSE_ID              506614 non-null  int64
      2   OFFENSE_CODE            506614 non-null  int64
      3   OFFENSE_CODE_EXTENSION  506614 non-null  int64
      4   OFFENSE_TYPE_ID         506614 non-null  object
      5   OFFENSE_CATEGORY_ID     506614 non-null  object
      6   FIRST_OCCURRENCE_DATE   506614 non-null  object
      7   LAST_OCCURRENCE_DATE    177181 non-null  object
      8   REPORTED_DATE           506614 non-null  object
      9   INCIDENT_ADDRESS        463293 non-null  object
      10  GEO_X                   502276 non-null  float64
      11  GEO_Y                   502276 non-null  float64
      12  GEO_LON                 502275 non-null  float64
      13  GEO_LAT                 502275 non-null  float64
      14  DISTRICT_ID             506613 non-null  float64
      15  PRECINCT_ID             506613 non-null  float64
      16  NEIGHBORHOOD_ID         506613 non-null  object
      17  IS_CRIME                506614 non-null  int64
      18  IS_TRAFFIC              506614 non-null  int64
     dtypes: float64(6), int64(6), object(7)
     memory usage: 73.4+ MB
```

Basic view of the first five rows of the data we are working with.

```
crime.head()
```

	INCIDENT_ID	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DA
0	2021224206	20212242062202 00	2202	0	burglary-residence-by-force	burglary	4/18/2021 10:30:00 F
1	2021225308	20212253082404 00	2404	0	theft-of-motor-vehicle	auto-theft	4/21/2021 11:25:00 F
2	20216009452	20216009452239900	2399	0	theft-other	larceny	3/22/2021 12:51:00 F
3	20216009439	20216009439230500	2305	0	theft-items-from-vehicle	theft-from-motor-vehicle	4/21/2021 12:00:00 F
4	20218017976	20218017976240400	2404	0	theft-of-motor-vehicle	auto-theft	3/9/2021 12:01:00 F

Checking the values that we have for each, there are a number of rows such as traffic accidents that don't play a pertinent role in our project so we will be removing them. Since we are focusing on crime, we will initally remove data points that are marked as "traffic-accident".

```
crime['OFFENSE_CATEGORY_ID'].value_counts()
```

```
traffic-accident               120591
all-other-crimes                84682
larceny                         52837
public-disorder                 52810
theft-from-motor-vehicle        48504
auto-theft                      36836
drug-alcohol                    27120
burglary                        26036
other-crimes-against-persons    24307
aggravated-assault              13977
white-collar-crime               6842
robbery                          6664
sexual-assault                   4338
arson                             691
murder                            379
Name: OFFENSE_CATEGORY_ID, dtype: int64
```

```
crime['OFFENSE_CODE'].value_counts()
```

```
5441    81452
2404    36030
5401    35973
2999    35302
2305    32524
        ...
5207        1
2309        1
3804        1
3580        1
5704        1
Name: OFFENSE_CODE, Length: 155, dtype: int64
```

```
crime['OFFENSE_TYPE_ID'].value_counts()
```

```
traffic-accident                81452
theft-of-motor-vehicle          36030
traffic-accident-hit-and-run    35973
theft-items-from-vehicle        32524
traf-other                      25820
                                ...
theft-from-yards                    1
explosives-posses                   1
riot                                1
bigamy                              1
homicide-negligent                  1
Name: OFFENSE_TYPE_ID, Length: 200, dtype: int64
```

```
crime['NEIGHBORHOOD_ID'].value_counts()
```

```
five-points     27441
central-park    21072
capitol-hill    18208
cbd             17128
montbello       17057
                ...
skyland          1897
rosedale         1596
country-club     1201
indian-creek      675
wellshire         577
Name: NEIGHBORHOOD_ID, Length: 79, dtype: int64
```

Viewing and removing data points that are marked as being a traffic accident. The dataset uses a binary system between "IS_TRAFFIC" and "IS_CRIME" to determine whether the incident was an accident or a crime using 0 (no) and 1 (yes). This is helpful in the sense that we do not have to go through other columns to remove this data.

```
crime['IS_TRAFFIC'].value_counts()
```

```
0    385738
1    120876
Name: IS_TRAFFIC, dtype: int64
```

```
crime = crime.loc[(crime['IS_TRAFFIC'] != 1)]
```

```
crime['IS_TRAFFIC'].value_counts()
```

```
0    385738
Name: IS_TRAFFIC, dtype: int64
```

```
crime.info()
```

```
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 385738 entries, 0 to 506613
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   INCIDENT_ID             385738 non-null  int64  
 1   OFFENSE_ID              385738 non-null  int64  
 2   OFFENSE_CODE            385738 non-null  int64  
 3   OFFENSE_CODE_EXTENSION  385738 non-null  int64  
 4   OFFENSE_TYPE_ID         385738 non-null  object 
 5   OFFENSE_CATEGORY_ID     385738 non-null  object 
 6   FIRST_OCCURRENCE_DATE   385738 non-null  object 
 7   LAST_OCCURRENCE_DATE    176565 non-null  object 
 8   REPORTED_DATE           385738 non-null  object 
 9   INCIDENT_ADDRESS        356259 non-null  object 
 10  GEO_X                   381400 non-null  float64
 11  GEO_Y                   381400 non-null  float64
 12  GEO_LON                 381400 non-null  float64
 13  GEO_LAT                 381400 non-null  float64
 14  DISTRICT_ID             385738 non-null  float64
 15  PRECINCT_ID             385738 non-null  float64
 16  NEIGHBORHOOD_ID         385738 non-null  object 
 17  IS_CRIME                385738 non-null  int64  
 18  IS_TRAFFIC              385738 non-null  int64  
dtypes: float64(6), int64(6), object(7)
memory usage: 58.9+ MB
```

```
crime = crime.loc[(crime['OFFENSE_CATEGORY_ID'] != 'all-other-crimes')]
```

```
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301317 entries, 0 to 506612
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   INCIDENT_ID             301317 non-null  int64  
 1   OFFENSE_ID              301317 non-null  int64  
 2   OFFENSE_CODE            301317 non-null  int64  
 3   OFFENSE_CODE_EXTENSION  301317 non-null  int64  
 4   OFFENSE_TYPE_ID         301317 non-null  object 
 5   OFFENSE_CATEGORY_ID     301317 non-null  object 
 6   FIRST_OCCURRENCE_DATE   301317 non-null  object 
 7   LAST_OCCURRENCE_DATE    170972 non-null  object 
 8   REPORTED_DATE           301317 non-null  object 
 9   INCIDENT_ADDRESS        277167 non-null  object 
 10  GEO_X                   296979 non-null  float64
 11  GEO_Y                   296979 non-null  float64
 12  GEO_LON                 296979 non-null  float64
 13  GEO_LAT                 296979 non-null  float64
 14  DISTRICT_ID             301317 non-null  float64
 15  PRECINCT_ID             301317 non-null  float64
 16  NEIGHBORHOOD_ID         301317 non-null  object 
 17  IS_CRIME                301317 non-null  int64  
 18  IS_TRAFFIC              301317 non-null  int64  
dtypes: float64(6), int64(6), object(7)
memory usage: 46.0+ MB
```

There are three options that we can use in terms of identofying the temporal component of crime in this project. They are under "FIRST_OCCURANCE_DATE", "LAST_OCCURRENCE_DATE", and "REPORTED_DATE". I decided to go with "FIRST_OCCURANCE_DATE" since there was differences between these dates and the dates that the crime was reported. Since we are trying to predict crime before it might happen, it seems to follow our goal for the project to look more towards the actual date that the crime was committed. Reported date might be helpful in other cases, such as identifying how long it takes from a the time that a crime occurs/noticed to its reporting.

Date modifcation to make it easier to use in our modeling. We will employ military time as well (24:00) for ease of use.

```
crime['FIRST_OCCURRENCE_DATE'] = pd.to_datetime(crime['FIRST_OCCURRENCE_DATE'])
```

```
crime.head()
```

	INCIDENT_ID	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DA
0	2021224206	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:
1	2021225308	2021225308240400	2404	0	theft-of-motor-vehicle	auto-theft	2021-04-21 23:25:
2	20216009452	20216009452239900	2399	0	theft-other	larceny	2021-03-22 12:51:

Looking at data for our null results before cleaning more rows and columns.

vehicle

```
crime.isnull().sum()
```

```
INCIDENT_ID                  0
OFFENSE_ID                   0
OFFENSE_CODE                 0
OFFENSE_CODE_EXTENSION       0
OFFENSE_TYPE_ID              0
OFFENSE_CATEGORY_ID          0
FIRST_OCCURRENCE_DATE        0
LAST_OCCURRENCE_DATE    130345
REPORTED_DATE                0
INCIDENT_ADDRESS         24150
GEO_X                     4338
GEO_Y                     4338
GEO_LON                   4338
GEO_LAT                   4338
DISTRICT_ID                  0
PRECINCT_ID                  0
NEIGHBORHOOD_ID              0
IS_CRIME                     0
IS_TRAFFIC                   0
dtype: int64
```

```
crime[crime['LAST_OCCURRENCE_DATE'].isnull()]
```

	INCIDENT_ID	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENC
1	2021225308	2021225308240400	2404	0	theft-of-motor-vehicle	auto-theft	2021-04-21
10	2021224625	2021224625230800	2308	0	theft-from-bldg	larceny	2021-04-21
11	2021224877	2021224877131501	1315	1	aggravated-assault-dv	aggravated-assault	2021-04-21
13	2021225155	2021225155531200	5312	0	disturbing-the-peace	public-disorder	2021-04-21
23	2021224336	2021224336352000	3520	0	drug-opium-or-deriv-sell	drug-alcohol	2021-04-21
...	...	...	...	...	...	...	
506593	20218030332	20218030332260201	2602	1	theft-fail-return-rent-veh	white-collar-crime	2021-05-20
506595	2021388043	2021388043110200	1102	0	sex-aslt-rape	sexual-assault	2021-07-08
506600	2021388020	2021388020131300	1313	0	assault-simple	other-crimes-against-persons	2021-07-08
506609	2021387732	2021387732357100	3571	0	drug-methampetamine-sell	drug-alcohol	2021-07-04
506611	2021387817	2021387817131600	1316	0	threats-to-injure	public-disorder	2021-07-07

130345 rows × 19 columns

```
crime[crime['INCIDENT_ADDRESS'].isnull()]
```

	INCIDENT_ID	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENC
91	20215002133	20215002133110200	1102	0	sex-aslt-rape	sexual-assault	2021-03-01
148	2021182331	2021182331220200	2202	0	burglary-residence-by-force	burglary	2021-03-31
151	2021183342	2021183342530901	5309	1	harassment-dv	public-disorder	2021-03-31
213	2021183015	2021183015360500	3605	0	indecent-exposure	other-crimes-against-persons	2021-03-31
216	2021182377	2021182377530903	5309	3	harassment-sexual-in-nature	public-disorder	2021-03-28
...	...	...	...	...	...	...	
506564	2021387912	2021387912239900	2399	0	theft-other	larceny	2021-07-08
506595	2021388043	2021388043110200	1102	0	sex-aslt-rape	sexual-assault	2021-07-08
506596	2021387154	2021387154110200	1102	0	sex-aslt-rape	sexual-assault	2020-10-10
506603	20216014146	20216014146239900	2399	0	theft-other	larceny	2021-07-02

Missing Spatial Information

The geographical values missing are interesting since as far as we can see they are correlated with sexual assualt crimes. Since we want to include these in our dataset we will need to find a way to incoporate them since the NaN values can cause issues in our models.

```
crime[crime['GEO_X'].isnull()]
```

	INCIDENT_ID	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENC
91	20215002133	20215002133110200	1102	0	sex-aslt-rape	sexual-assault	2021-03-01
347	2021183357	2021183357110200	1102	0	sex-aslt-rape	sexual-assault	2021-03-31
601	2016118614	2016118614110200	1102	0	sex-aslt-rape	sexual-assault	2016-02-23
611	20215001826	20215001826110200	1102	0	sex-aslt-rape	sexual-assault	2021-04-08
696	2021174758	2021174758110200	1102	0	sex-aslt-rape	sexual-assault	2020-12-13
...	...	...	...	...	...	...	
505985	2021402348	2021402348360101	3601	1	sex-aslt-fondle-adult-victim	sexual-assault	2021-07-14
506479	2021385480	2021385480110200	1102	0	sex-aslt-rape	sexual-assault	2021-05-01
506541	2021378678	2021378678360101	3601	1	sex-aslt-fondle-adult-victim	sexual-assault	2021-07-03
506595	2021388043	2021388043110200	1102	0	sex-aslt-rape	sexual-assault	2021-07-08
506596	2021387154	2021387154110200	1102	0	sex-aslt-rape	sexual-assault	2020-10-10

4338 rows × 19 columns

```
p_stations = pd.read_csv('/content/drive/MyDrive/police_stations.csv')
```

```
p_stations.head(8)
```

	POLICE_STATION_ID	STATION	STATION_NAME	ADDRESS_LINE1	CITY	DISTRICT_NUMBER	LAT	LONG
0	0	D1	DISTRICT 1	1311 W 46th Ave	Denver	1	39.780670	-105.002970
1	3	D2	DISTRICT 2	3921 N Holly St	Denver	2	39.770620	-104.923430
2	11	D3	DISTRICT 3	1625 S Universitv Blvd	Denver	3	39.686840	-104.960570

```
geo_loc = zip(list(p_stations.LAT), list(p_stations.LONG))

# convert district number field to integer
p_stations.DISTRICT_NUMBER = p_stations.DISTRICT_NUMBER.astype(int)

station_dict = dict(zip(list(p_stations.DISTRICT_NUMBER), geo_loc))
station_dict
```

```
{1: (39.78067, -105.00296999999999),
 2: (39.77062, -104.92343000000001),
 3: (39.686840000000004, -104.96056999999999),
 4: (39.67768, -105.01983999999999),
 5: (39.718361, -105.02736100000001),
 6: (39.741087, -104.97863000000001),
 7: (39.849563, -104.673847)}
```

```
# get list of index values for rows where "geo_lat" is null
ind = list(crime.loc[pd.isna(crime["GEO_LAT"]), :].index)


# create empty list for missing lat/lon district ids
# using the index numbers in ind, lookup the district id from the indexed row.
dist_of_missing_geo_lat = []
for i in ind:
    dist_of_missing_geo_lat.append(crime.loc[i]['DISTRICT_ID'])


geo_loc_of_dist_stations = []
for dist in dist_of_missing_geo_lat:
    geo_loc_of_dist_stations.append(station_dict.get(dist))



lats = []
lons = []
for lat, lon in geo_loc_of_dist_stations:
    lats.append(lat)
    lons.append(lon)


# replace the missing lat/lon with the values in lats and lons by index label
l = 0
for i in ind:
    crime.GEO_LAT.loc[i] = lats[l]
    crime.GEO_LON.loc[i] = lons[l]
    l = l + 1
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
  iloc._setitem_with_indexer(indexer, value)
```

```
# check number of nulls in the lat column
len(crime[crime["GEO_LAT"].isnull()])
```

```
0
```

```
len(crime[crime["GEO_LON"].isnull()])
```

```
0
```

```
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301317 entries, 0 to 506612
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   INCIDENT_ID             301317 non-null  int64
 1   OFFENSE_ID              301317 non-null  int64
 2   OFFENSE_CODE            301317 non-null  int64
 3   OFFENSE_CODE_EXTENSION  301317 non-null  int64
```

```
 4   OFFENSE_TYPE_ID         301317 non-null  object
 5   OFFENSE_CATEGORY_ID     301317 non-null  object
 6   FIRST_OCCURRENCE_DATE   301317 non-null  datetime64[ns]
 7   LAST_OCCURRENCE_DATE    170972 non-null  object
 8   REPORTED_DATE           301317 non-null  object
 9   INCIDENT_ADDRESS        277167 non-null  object
 10  GEO_X                   296979 non-null  float64
 11  GEO_Y                   296979 non-null  float64
 12  GEO_LON                 301317 non-null  float64
 13  GEO_LAT                 301317 non-null  float64
 14  DISTRICT_ID             301317 non-null  float64
 15  PRECINCT_ID             301317 non-null  float64
 16  NEIGHBORHOOD_ID         301317 non-null  object
 17  IS_CRIME                301317 non-null  int64
 18  IS_TRAFFIC              301317 non-null  int64
dtypes: datetime64[ns](1), float64(6), int64(6), object(6)
memory usage: 56.0+ MB
```

Dropping Unnecessary Columns

In this section we drop columns that are not pertinent to our work. Since we want to look at the probability of being a victim of a crime based on neighborhood, there are some columns that do not correlate with our goal.

```
drop = ['INCIDENT_ID','LAST_OCCURRENCE_DATE', 'INCIDENT_ADDRESS', 'IS_TRAFFIC', 'IS_CRIME', 'REPORTED_DATE']
crime = crime.drop(columns= drop, axis=1)
```

```
# clean up NEIGHBORHOOD_ID
crime.rename(columns = {'NEIGHBORHOOD_ID':'NEIGHBORHOOD'}, inplace = True)
crime['NEIGHBORHOOD'] = crime['NEIGHBORHOOD'].str.replace('-', ' ')
```

```
crime.head()
```

	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE	GEO_X
0	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:00	3142828.0
1	2021225308240400	2404	0	theft-of-motor-vehicle	auto-theft	2021-04-21 23:25:00	3124936.0
2	20216009452239900	2399	0	theft-other	larceny	2021-03-22 12:51:00	3149191.0
3	20216009439230500	2305	0	theft-items-from-vehicle	theft-from-motor-vehicle	2021-04-21 12:00:00	3146781.0

Save our cleaned crime dataframe

```
crime.to_csv('crime_cleaned.csv')
crime2 = pd.read_csv('crime_cleaned.csv')
```

```
crime2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301317 entries, 0 to 301316
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Unnamed: 0              301317 non-null  int64
 1   OFFENSE_ID              301317 non-null  int64
 2   OFFENSE_CODE            301317 non-null  int64
 3   OFFENSE_CODE_EXTENSION  301317 non-null  int64
 4   OFFENSE_TYPE_ID         301317 non-null  object
 5   OFFENSE_CATEGORY_ID     301317 non-null  object
 6   FIRST_OCCURRENCE_DATE   301317 non-null  object
 7   GEO_X                   296979 non-null  float64
 8   GEO_Y                   296979 non-null  float64
 9   GEO_LON                 301317 non-null  float64
 10  GEO_LAT                 301317 non-null  float64
 11  DISTRICT_ID             301317 non-null  float64
 12  PRECINCT_ID             301317 non-null  float64
 13  NEIGHBORHOOD            301317 non-null  object
dtypes: float64(6), int64(4), object(4)
memory usage: 32.2+ MB
```

```
crime2.head()
```

	Unnamed: 0	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE
0	0	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:00
1	1	2021225308240400	2404	0	theft-of-motor-vehicle	auto-theft	2021-04-21 23:25:00
2	2	20216009452239900	2399	0	theft-other	larceny	2021-03-22 12:51:00

```
drop_2 = ['Unnamed: 0']


crime2 = crime2.drop(columns=drop_2, axis=1)


crime2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301317 entries, 0 to 301316
Data columns (total 13 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              301317 non-null  int64  
 1   OFFENSE_CODE            301317 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  301317 non-null  int64  
 3   OFFENSE_TYPE_ID         301317 non-null  object 
 4   OFFENSE_CATEGORY_ID     301317 non-null  object 
 5   FIRST_OCCURRENCE_DATE   301317 non-null  object 
 6   GEO_X                   296979 non-null  float64
 7   GEO_Y                   296979 non-null  float64
 8   GEO_LON                 301317 non-null  float64
 9   GEO_LAT                 301317 non-null  float64
 10  DISTRICT_ID             301317 non-null  float64
 11  PRECINCT_ID             301317 non-null  float64
 12  NEIGHBORHOOD            301317 non-null  object 
dtypes: float64(6), int64(3), object(4)
memory usage: 29.9+ MB
```

COMMUNITY DATA REVIEW

```
community_data = pd.read_csv("/content/drive/MyDrive/american community survey nbrhd 2015 2019.csv")
community_data.head()
```

	NBHD_NAME	TTL_POPULATION_ALL	HISPANIC_OR_LATINO	WHITE	BLACK	NATIVE_AMERICAN	ASIAN	HAWAIIAN_PI	OTHER_RACE	TWO_OR_MORE
0	Chaffee Park	3820.0	1899.0	1750.0	53.0	9.0	8.0	21.0	0.0	80.0
1	Sunnyside	10091.0	4424.0	4763.0	458.0	57.0	181.0	11.0	60.0	137.0
2	Highland	10549.0	2619.0	7290.0	101.0	211.0	201.0	0.0	0.0	127.0
3	Globeville	4377.0	2551.0	1453.0	182.0	10.0	53.0	0.0	18.0	110.0
4	Jefferson Park	3490.0	964.0	2335.0	13.0	0.0	160.0	0.0	0.0	18.0

5 rows × 147 columns

```
community_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Columns: 147 entries, NBHD_NAME to PCT_FAM_POVERTY
dtypes: float64(56), int64(90), object(1)
memory usage: 89.7+ KB
```

```
drop1 = ['FOREIGN_BORN_FB', 'EUROPEAN_FB','NORTHERN_EUROPE_FB','WESTERN_EUROPE_FB',
         'SOUTHERN_EUROPE_FB','EASTERN_EUROPE_FB','ASIA_FB','EASTERN_ASIA_FB',
         'SOUTH_CENTRAL_ASIA_FB','SOUTH_EASTERN_ASIA_FB','WESTERN_ASIA_FB',
         'AFRICA_FB','EASTERN_AFRICA_FB','MIDDLE_AFRICA_FB','SOUTHERN_AFRICA_FB',
         'WESTERN_AFRICA_FB','OCEANIA_FB','AMERICAS_FB','LATIN_AMERICA_FB',
         'CARRIBEAN_FB','CENTRAL_AMERICA_FB','SOUTH_AMERICA_FB','NORTH_AMERICA_FB',
         'BUILT_1939_OR_EARLIER','BUILT_1940_1949','BUILT_1950_1959','BUILT_1960_1969',
         'BUILT_1970_1979','BUILT_1980_1989','BUILT_1990_1999','BUILT_2000_2009',
         'BUILT_2010_2013','BUILT_2014_OR_LATER','MED_YR_STRUCTURE_BUILT',
          'PCT_HISPANIC','PCT_WHITE','PCT_BLACK','PCT_NATIVEAM','PCT_ASIAN',
          'PCT_HAWAIIANPI','PCT_OTHERRACE','PCT_TWOORMORE_RACES','AGELESS18',
          'AGE65PLUS','AGE65PLUS','PCT_AGE65PLUS','MEDIAN_AGE_ALL','MEDIAN_AGE_MALE',
```

```
        'MEDIAN_AGE_FEMALE','MALE_HHLDR_NO_WIFE_PRESENT','FEMALE_HHLDR_NO_HSBND_PRESENT',
        'NONFAMILY_HOUSEHOLD','HOUSEHOLDER_ALONE','HOUSEHOLDER_NOT_ALONE','PCT_AGELESS18',
        'TTL_AGE_3_PLUS_ENRSTATUS','ENROLLED_IN_SCHOOL','NURSERY_OR_PRESCHOOL',
        'KINDERGARTEN','GRADES_1_TO_4','GRADES_5_TO_8','GRADES_9_TO_12','COLLEGE_UNDERGRADUATE',
        'GRADUATE_SCHOOL','NOT_ENROLLED','TOTAL_COMMUTERS','COMMUTE_LESS_15',
        'COMMUTE_15_TO_30','COMMUTE_30_TO_45','RENTER_OCCUPIED_HU','OTHER_FAMILY',
        'MEDIAN_EARN_MALE','MEDIAN_EARN_FEMALE','MEDIAN_EARN_FEMALE','PER_CAPITA_INCOME',
        'MARRIED_COUPLE_FAMILY','MARRIED_COUPLE_FAMILY', 'COMMUTE_45_TO_60',
        'COMMUTE_60_TO_PLUS','TTLPOP_25PLUS_EDU','TTLPOP_5PLUS_LNG',
        'ONLY_ENGLISH_LNG','SPANISH_LNG','TTL_HOUSING_UNITS','TTL_HOUSEHOLDS',
        'FAMILY_HOUSEHOLDS','HH_INC_LESS_10000','HH_INC_10000_14999','HH_INC_15000_19999',
        'HH_INC_20000_24999','HH_INC_25000_29999','HH_INC_30000_34999','HH_INC_35000_39999',
        'HH_INC_40000_44999','HH_INC_45000_49999','HH_INC_50000_59999','HH_INC_60000_74999',
        'HH_INC_75000_99999','HH_INC_100000_124999','HH_INC_125000_149999',
        'HH_INC_150000_199999','HH_INC_OVER_200000','AGE_LESS_5','AGE_5_TO_9',
        'AGE_10_TO_14','AGE_15_TO_17','AGE_0_TO_9','AGE_10_TO_19','AGE_20_TO_29',
        'AGE_30_TO_39','AGE_40_TO_49','AGE_50_TO_59','AGE_60_TO_69','AGE_70_TO_79',
        'AGE_80_PLUS','LESS_THAN_HS_DIPLOMA_EDU','HSGRAD_OR_EQUIV_EDU','SOMECOLLEGE_OR_AA_EDU',
        'BACHELORS_OR_HIGHER_EDU','NATIVE','MEDEARN_LESSHS','MEDEARN_HIGHSCHOOL',
        'MEDEARN_SOMECOLLEGE','MEDEARN_BACHELORS','MEDEARN_GRAD_PROFESSIONAL',
        'HISPANIC_OR_LATINO', 'WHITE', 'BLACK', 'NATIVE_AMERICAN', 'ASIAN',
        'HAWAIIAN_PI', 'OTHER_RACE', 'TWO_OR_MORE']
community_data2 = community_data.drop(columns= drop1, axis=1)
community_data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   NBHD_NAME           78 non-null     object 
 1   TTL_POPULATION_ALL  78 non-null     float64
 2   MALE                78 non-null     float64
 3   FEMALE              78 non-null     float64
 4   OCCUPIED_HU         78 non-null     int64  
 5   VACANT_HU           78 non-null     int64  
 6   OWNER_OCCUPIED_HU   78 non-null     int64  
 7   MED_HH_INCOME       78 non-null     int64  
 8   MED_FAMILY_INCOME   78 non-null     int64  
 9   AVG_HH_INCOME       78 non-null     int64  
 10  AVG_FAM_INCOME      78 non-null     int64  
 11  MEDIAN_EARNINGS     78 non-null     int64  
 12  MED_CONTRACT_RENT   78 non-null     int64  
 13  MED_GROSS_RENT      78 non-null     int64  
 14  MEDIAN_HOME_VALUE   78 non-null     int64  
 15  PCT_POVERTY         78 non-null     float64
 16  PCT_FAM_POVERTY     78 non-null     float64
dtypes: float64(5), int64(11), object(1)
memory usage: 10.5+ KB
```

```
community_data2.head(15)
```

	NBHD_NAME	TTL_POPULATION_ALL	MALE	FEMALE	OCCUPIED_HU	VACANT_HU	OWNER_OCCUPIED_HU	MED_HH_INCOME	MED_FAMILY_INCOME	AVG
0	Chaffee Park	3820.0	1779.0	2041.0	1663	100	981	62639	75185	

```
community_data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   NBHD_NAME           78 non-null     object
 1   TTL_POPULATION_ALL  78 non-null     float64
 2   MALE                78 non-null     float64
 3   FEMALE              78 non-null     float64
 4   OCCUPIED_HU         78 non-null     int64
 5   VACANT_HU           78 non-null     int64
 6   OWNER_OCCUPIED_HU   78 non-null     int64
 7   MED_HH_INCOME       78 non-null     int64
 8   MED_FAMILY_INCOME   78 non-null     int64
 9   AVG_HH_INCOME       78 non-null     int64
 10  AVG_FAM_INCOME      78 non-null     int64
 11  MEDIAN_EARNINGS     78 non-null     int64
 12  MED_CONTRACT_RENT   78 non-null     int64
 13  MED_GROSS_RENT      78 non-null     int64
 14  MEDIAN_HOME_VALUE   78 non-null     int64
 15  PCT_POVERTY         78 non-null     float64
 16  PCT_FAM_POVERTY     78 non-null     float64
dtypes: float64(5), int64(11), object(1)
memory usage: 10.5+ KB
```

```
crime2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301317 entries, 0 to 301316
Data columns (total 13 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   OFFENSE_ID              301317 non-null  int64
 1   OFFENSE_CODE            301317 non-null  int64
 2   OFFENSE_CODE_EXTENSION  301317 non-null  int64
 3   OFFENSE_TYPE_ID         301317 non-null  object
 4   OFFENSE_CATEGORY_ID     301317 non-null  object
 5   FIRST_OCCURRENCE_DATE   301317 non-null  object
 6   GEO_X                   296979 non-null  float64
 7   GEO_Y                   296979 non-null  float64
 8   GEO_LON                 301317 non-null  float64
 9   GEO_LAT                 301317 non-null  float64
 10  DISTRICT_ID             301317 non-null  float64
 11  PRECINCT_ID             301317 non-null  float64
 12  NEIGHBORHOOD            301317 non-null  object
dtypes: float64(6), int64(3), object(4)
memory usage: 29.9+ MB
```

```
community_data2.rename(columns = {'NBHD_NAME':'NEIGHBORHOOD'}, inplace = True)
```

The names in the neighborhood column will be used to merge the correct neighborhood data with reports in the crime data. To do this, the names in the column must be formatted correctly in order to match. I'll examine the census neighborhood names and compare them to the crime data names.

In addition, the names will be saved to a csv file for future operations.

```
nbhd = community_data2.NEIGHBORHOOD.unique()
np.savetxt('community_data2.csv', nbhd, delimiter=',', fmt='%s')
nbhd
```

```
array(['Chaffee Park', 'Sunnyside', 'Highland', 'Globeville',
       'Jefferson Park', 'Sun Valley', 'Valverde', 'Athmar Park',
       'Virginia Village', 'DIA', 'University Hills', 'Harvey Park',
       'Mar Lee', 'Westwood', 'East Colfax', 'Auraria', 'University Park',
       'Platt Park', 'College View - South Platte', 'Overland',
       'Ruby Hill', 'Kennedy', 'Hampden', 'Baker', 'Fort Logan',
       'Bear Valley', 'Harvey Park South', 'Southmoor Park',
       'Hampden South', 'Goldsmith', 'Cory - Merrill', 'Windsor',
       'Belcaro', 'Washington Park', 'Washington Park West',
       'Elyria Swansea', 'Wellshire', 'University', 'Rosedale',
       'Cheesman Park', 'Hilltop', 'Montclair', 'Hale', 'North Park Hill',
       'South Park Hill', 'Central Park', 'Montbello', 'Lowry Field',
       'West Colfax', 'West Highland', 'Sloan Lake', 'Berkeley', 'Regis',
       'Lincoln Park', 'City Park West', 'Whittier', 'Skyland', 'Cole',
```

```
            'Marston', 'Washington Virginia Vale', 'Barnum', 'Barnum West',
            'Villa Park', 'Clayton', 'Gateway - Green Valley Ranch',
            'Indian Creek', 'Five Points', 'Northeast Park Hill', 'Speer',
            'Cherry Creek', 'Country Club', 'Congress Park', 'City Park',
            'Capitol Hill', 'North Capitol Hill', 'Civic Center', 'CBD',
            'Union Station'], dtype=object)
```

```
# remove - and extra spaces
community_data2['NEIGHBORHOOD'] = community_data2['NEIGHBORHOOD'].str.lower()
community_data2.NEIGHBORHOOD.replace(to_replace='gateway - green valley ranch', value='gateway green valley ranch', inplace=True )
community_data2.NEIGHBORHOOD.replace(to_replace='college view - south platte', value='college view south platte', inplace=True )
community_data2.NEIGHBORHOOD.replace(to_replace='cory - merrill', value='cory merrill', inplace=True )
community_data2.NEIGHBORHOOD.unique()
```

```
    array(['chaffee park', 'sunnyside', 'highland', 'globeville',
           'jefferson park', 'sun valley', 'valverde', 'athmar park',
           'virginia village', 'dia', 'university hills', 'harvey park',
           'mar lee', 'westwood', 'east colfax', 'auraria', 'university park',
           'platt park', 'college view south platte', 'overland', 'ruby hill',
           'kennedy', 'hampden', 'baker', 'fort logan', 'bear valley',
           'harvey park south', 'southmoor park', 'hampden south',
           'goldsmith', 'cory merrill', 'windsor', 'belcaro',
           'washington park', 'washington park west', 'elyria swansea',
           'wellshire', 'university', 'rosedale', 'cheesman park', 'hilltop',
           'montclair', 'hale', 'north park hill', 'south park hill',
           'central park', 'montbello', 'lowry field', 'west colfax',
           'west highland', 'sloan lake', 'berkeley', 'regis', 'lincoln park',
           'city park west', 'whittier', 'skyland', 'cole', 'marston',
           'washington virginia vale', 'barnum', 'barnum west', 'villa park',
           'clayton', 'gateway green valley ranch', 'indian creek',
           'five points', 'northeast park hill', 'speer', 'cherry creek',
           'country club', 'congress park', 'city park', 'capitol hill',
           'north capitol hill', 'civic center', 'cbd', 'union station'],
          dtype=object)
```

```
crime2.NEIGHBORHOOD.unique()
```

```
    array(['civic center', 'bear valley', 'cole', 'platt park', 'dia',
           'washington park west', 'northeast park hill', 'speer',
           'harvey park', 'elyria swansea', 'union station', 'west colfax',
           'university park', 'ruby hill', 'berkeley', 'five points',
           'stapleton', 'fort logan', 'hampden south', 'central park',
           'auraria', 'college view south platte', 'sunnyside',
           'cheesman park', 'montbello', 'highland', 'montclair',
           'harvey park south', 'goldsmith', 'hale', 'east colfax',
           'gateway green valley ranch', 'washington virginia vale',
           'sloan lake', 'globeville', 'athmar park', 'cbd', 'windsor',
           'clayton', 'cory merrill', 'sun valley', 'regis', 'lincoln park',
           'barnum', 'villa park', 'baker', 'marston', 'cherry creek',
           'southmoor park', 'capitol hill', 'university', 'hilltop',
           'lowry field', 'westwood', 'whittier', 'north capitol hill',
           'city park west', 'kennedy', 'virginia village', 'west highland',
           'belcaro', 'north park hill', 'skyland', 'jefferson park',
           'south park hill', 'congress park', 'overland', 'hampden',
           'university hills', 'mar lee', 'valverde', 'washington park',
           'country club', 'barnum west', 'indian creek', 'rosedale',
           'city park', 'chaffee park', 'wellshire'], dtype=object)
```

```
combined = pd.merge(crime2, community_data2, on= 'NEIGHBORHOOD')
```

```
combined.head(25)
```

	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE	GEO_X
0	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:00	3142828.0
1	2021224336352000	3520	0	drug-opium-or-deriv-sell	drug-alcohol	2021-04-21 13:15:00	3144130.0
2	2021224464359900	3599	0	drug-pcs-other-drug	drug-alcohol	2021-04-21 13:25:00	3144130.0
3	2021224359351000	3510	0	drug-heroin-sell	drug-alcohol	2021-04-21 13:20:00	3144130.0
4	2021224359220600	2206	0	burglary-poss-of-tools	burglary	2021-04-21 13:20:00	3144130.0
5	20216009462230400	2304	0	theft-parts-from-vehicle	theft-from-motor-vehicle	2021-04-21 08:00:00	3142380.0
6	20215002135359901	3599	1	drug-make-sell-other-drug	drug-alcohol	2021-05-12 22:39:00	3142777.0
7	20215002133110200	1102	0	sex-aslt-rape	sexual-assault	2021-03-01 00:00:00	NaN
8	20216008038230400	2304	0	theft-parts-from-vehicle	theft-from-motor-vehicle	2021-03-28 11:00:00	3142832.0
9	2021224336351000	3510	0	drug-heroin-sell	drug-alcohol	2021-04-21 13:15:00	3144130.0
10	2016335725131300	1313	0	assault-simple	other-crimes-against-persons	2016-05-29 13:34:00	3144156.0
11	2021224464353200	3532	0	drug-cocaine-possess	drug-alcohol	2021-04-21 13:25:00	3144130.0
12	20216009423239900	2399	0	theft-other	larceny	2021-04-20 16:25:00	3143230.0
13	20215001827299900	2999	0	criminal-mischief-other	public-disorder	2021-04-20 08:30:00	3142777.0
14	2021224572353200	3532	0	drug-cocaine-possess	drug-alcohol	2021-04-21 15:12:00	3144130.0
15	2021224553357100	3571	0	drug-methampetamine-sell	drug-alcohol	2021-04-21 14:40:00	3144130.0
16	2016245068356200	3562	0	drug-marijuana-possess	drug-alcohol	2016-04-20 14:20:00	3144130.0
17	20216007742299900	2999	0	criminal-mischief-other	public-disorder	2021-03-26 19:00:00	3143021.0
					other-crimes-against-		

Checking the sizes of our dataframes, original crime, community data, and the combined dataset.

19	2021174995131600	1316	0	threats-to-injure	public-disorder	2021-03-27 16:00:00	3143838.0

```
print(crime2.shape)
print(community_data2.shape)
print(combined.shape)
```

```
(301317, 13)
(78, 17)
(300437, 29)
```

22	2021177374299900	2399	0	other	public-disorder	2021-03-28 22:09:00	3142202.0

```
combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 300437 entries, 0 to 300436
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              300437 non-null  int64  
 1   OFFENSE_CODE            300437 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  300437 non-null  int64  
 3   OFFENSE_TYPE_ID         300437 non-null  object 
 4   OFFENSE_CATEGORY_ID     300437 non-null  object 
 5   FIRST_OCCURRENCE_DATE   300437 non-null  object 
 6   GEO_X                   296099 non-null  float64
 7   GEO_Y                   296099 non-null  float64
 8   GEO_LON                 300437 non-null  float64
 9   GEO_LAT                 300437 non-null  float64
 10  DISTRICT_ID             300437 non-null  float64
 11  PRECINCT_ID             300437 non-null  float64
 12  NEIGHBORHOOD            300437 non-null  object 
 13  TTL_POPULATION_ALL      300437 non-null  float64
 14  MALE                    300437 non-null  float64
 15  FEMALE                  300437 non-null  float64
 16  OCCUPIED_HU             300437 non-null  int64  
 17  VACANT_HU               300437 non-null  int64
```

```
 18  OWNER_OCCUPIED_HU       300437 non-null  int64
 19  MED_HH_INCOME           300437 non-null  int64
 20  MED_FAMILY_INCOME       300437 non-null  int64
 21  AVG_HH_INCOME           300437 non-null  int64
 22  AVG_FAM_INCOME          300437 non-null  int64
 23  MEDIAN_EARNINGS         300437 non-null  int64
 24  MED_CONTRACT_RENT       300437 non-null  int64
 25  MED_GROSS_RENT          300437 non-null  int64
 26  MEDIAN_HOME_VALUE       300437 non-null  int64
 27  PCT_POVERTY             300437 non-null  float64
 28  PCT_FAM_POVERTY         300437 non-null  float64
dtypes: float64(11), int64(14), object(4)
memory usage: 68.8+ MB
```

```
print(crime.loc[0, 'NEIGHBORHOOD'])
print(combined.loc[0, 'NEIGHBORHOOD'])
```

```
civic center
civic center
```

Verify merged data by checking a random data point.

```
crime.loc[101]
```

```
OFFENSE_ID                            2021264355131303
OFFENSE_CODE                                      1313
OFFENSE_CODE_EXTENSION                               3
OFFENSE_TYPE_ID                   assault-police-simple
OFFENSE_CATEGORY_ID       other-crimes-against-persons
FIRST_OCCURRENCE_DATE              2021-05-10 22:10:00
GEO_X                                     3.14329e+06
GEO_Y                                     1.69622e+06
GEO_LON                                       -104.99
GEO_LAT                                       39.7438
DISTRICT_ID                                         6
PRECINCT_ID                                       611
NEIGHBORHOOD                                      cbd
Name: 101, dtype: object
```

```
combined.loc[combined['OFFENSE_ID'] == 2021264355131303]
```

	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE	GEO
167537	2021264355131303	1313	3	assault-police-simple	other-crimes-against-persons	2021-05-10 22:10:00	314329

```
print(combined.loc[167537]['NEIGHBORHOOD'])
```

```
cbd
```

Saving our merged dataframe

```
combined.to_csv('combined.csv')
```

```
crime2.isnull().sum()
```

```
OFFENSE_ID                  0
OFFENSE_CODE                0
OFFENSE_CODE_EXTENSION      0
OFFENSE_TYPE_ID             0
OFFENSE_CATEGORY_ID         0
FIRST_OCCURRENCE_DATE       0
GEO_X                    4338
GEO_Y                    4338
GEO_LON                     0
GEO_LAT                     0
DISTRICT_ID                 0
PRECINCT_ID                 0
NEIGHBORHOOD                0
dtype: int64
```

```
crime.to_csv('crime_cleaned.csv', index=False)
```

```
combined.head()
```

	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE	GEO_X
0	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:00	3142828.0
1	2021224336352000	3520	0	drug-opium-or-deriv-sell	drug-alcohol	2021-04-21 13:15:00	3144130.0
2	2021224464359900	3599	0	drug-pcs-other-drug	drug-alcohol	2021-04-21 13:25:00	3144130.0
3	2021224359351000	3510	0	drug-heroin-sell	drug-alcohol	2021-04-21 13:20:00	3144130.0
4	2021224359220600	2206	0	burglary-poss-of-tools	burglary	2021-04-21 13:20:00	3144130.0

```
crime2.info()
```

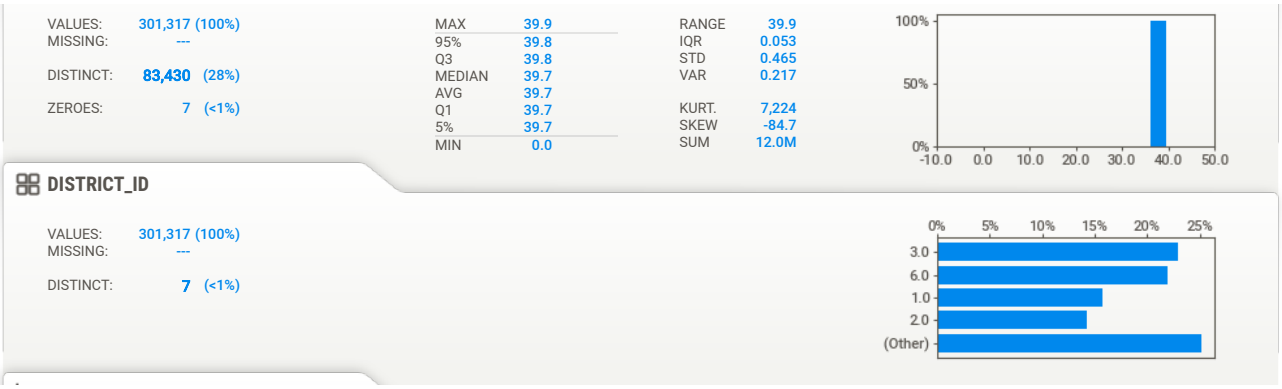
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301317 entries, 0 to 301316
Data columns (total 13 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              301317 non-null  int64  
 1   OFFENSE_CODE            301317 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  301317 non-null  int64  
 3   OFFENSE_TYPE_ID         301317 non-null  object 
 4   OFFENSE_CATEGORY_ID     301317 non-null  object 
 5   FIRST_OCCURRENCE_DATE   301317 non-null  object 
 6   GEO_X                   296979 non-null  float64
 7   GEO_Y                   296979 non-null  float64
 8   GEO_LON                 301317 non-null  float64
 9   GEO_LAT                 301317 non-null  float64
 10  DISTRICT_ID             301317 non-null  float64
 11  PRECINCT_ID             301317 non-null  float64
 12  NEIGHBORHOOD            301317 non-null  object 
dtypes: float64(6), int64(3), object(4)
memory usage: 29.9+ MB
```

Exploratory Data Analysis

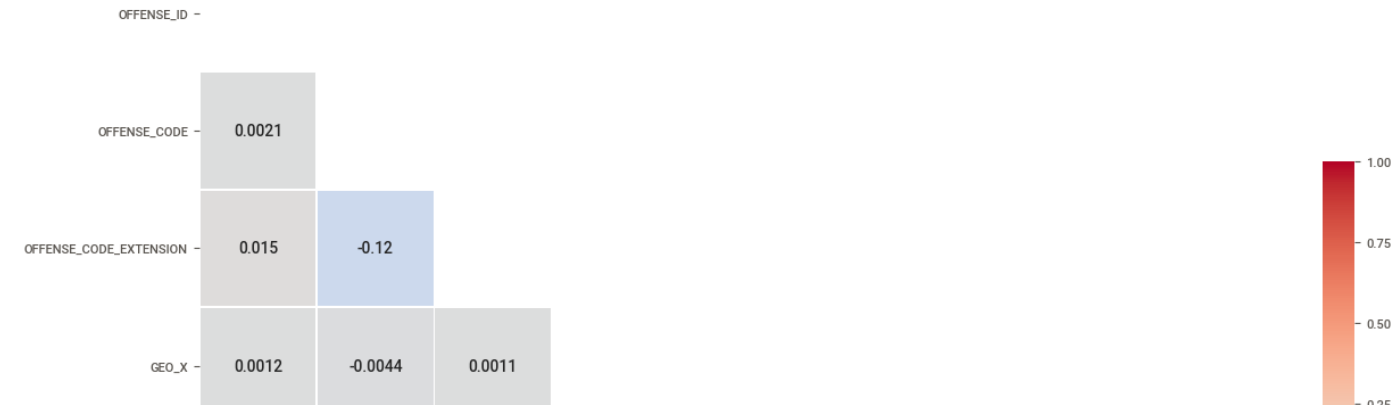
```
!pip install sweetviz

# importing sweetviz
import sweetviz as sv
# analyzing the dataset
crime_report = sv.analyze(crime2)
#display the report
crime_report.show_notebook(w=None,
                h=None,
                scale=None,
                layout='widescreen',
                filepath=None)
```

```
Requirement already satisfied: sweetviz in /usr/local/lib/python3.7/dist-packages (2.1.3)
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3 in /usr/local/lib/python3.7/dist-packages (from sweetviz)
Requirement already satisfied: jinja2>=2.11.1 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (2.11.3)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (1.19.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (1.4.1)
Requirement already satisfied: tqdm>=4.43.0 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (4.62.0)
Requirement already satisfied: matplotlib>=3.1.3 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (3.2.2)
Requirement already satisfied: importlib-resources>=1.2.0 in /usr/local/lib/python3.7/dist-packages (from sweetviz) (5.2.2)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-packages (from importlib-resources>=1.2.0->sweetviz)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2>=2.11.1->sweetviz) (2.0.1
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.1.3->sweetviz)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotl
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.1.3->sweetviz) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.1.3->sweetviz) (0.10.0
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.1.3->sweetviz) (1
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25
```

Done! Use 'show' commands to display/save. [100%] 00:00 -> (00:00 left)



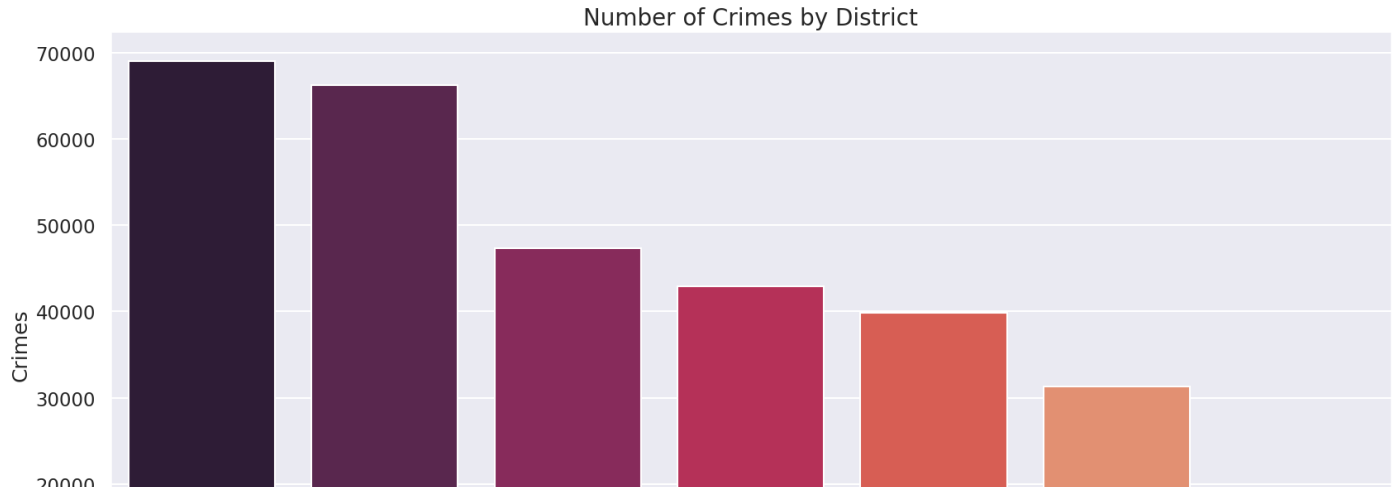

```
corr = crime2.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corr, mask=mask, cmap='coolwarm', vmax=1, center=0,
square=True, linewidths=.5,annot=True, cbar_kws={"shrink": .5});
```




```
crime2.nunique()
```

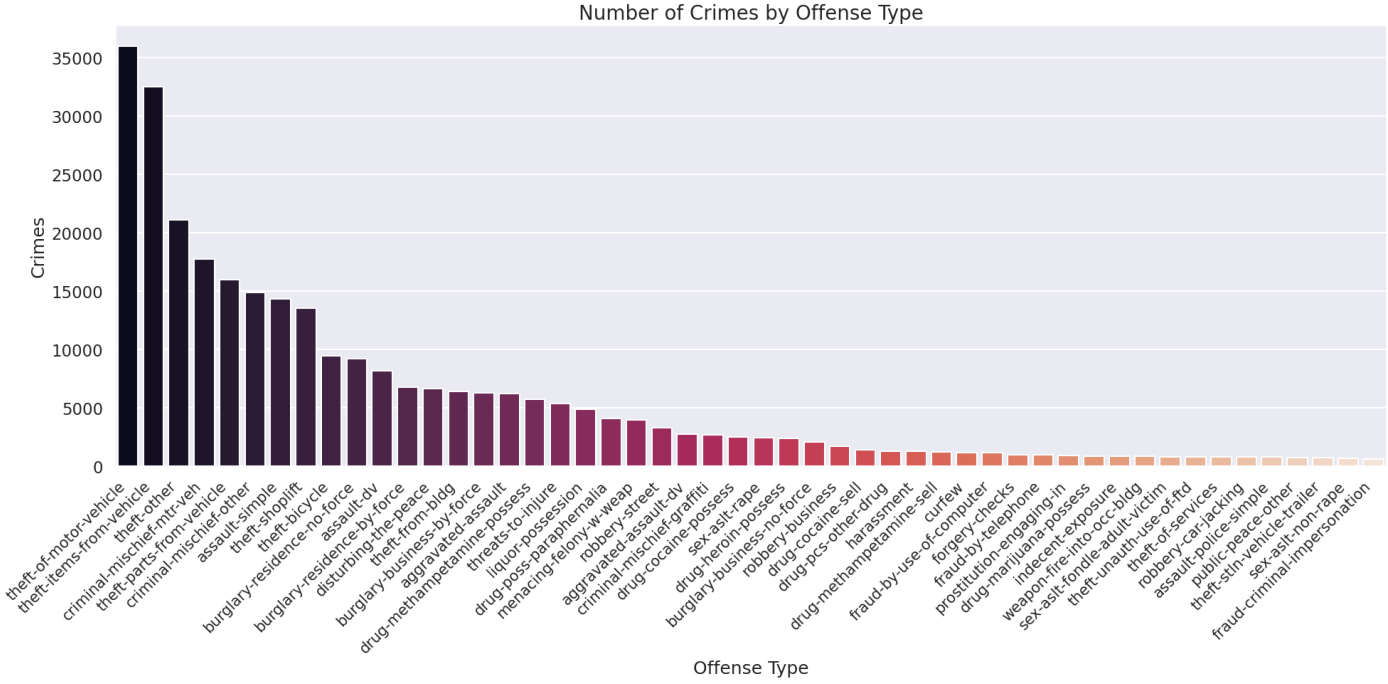
```
OFFENSE_ID                301317
OFFENSE_CODE                  98
OFFENSE_CODE_EXTENSION         6
OFFENSE_TYPE_ID              128
OFFENSE_CATEGORY_ID           13
FIRST_OCCURRENCE_DATE     195703
GEO_X                      43227
GEO_Y                      42138
GEO_LON                    83644
GEO_LAT                    83430
DISTRICT_ID                    7
PRECINCT_ID                   36
NEIGHBORHOOD                  79
dtype: int64
```

```
sns.set_context("talk")
sns.set_style("darkgrid")
plt.figure(figsize =(20,10))
plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontsize='small'
)

ax = sns.countplot(x='DISTRICT_ID', data=combined,
                   order = crime['DISTRICT_ID'].value_counts().head(7).index, palette = 'rocket')
plt.xlabel('District')
plt.ylabel('Crimes')
plt.title('Number of Crimes by District', fontsize=20)
plt.show()
```

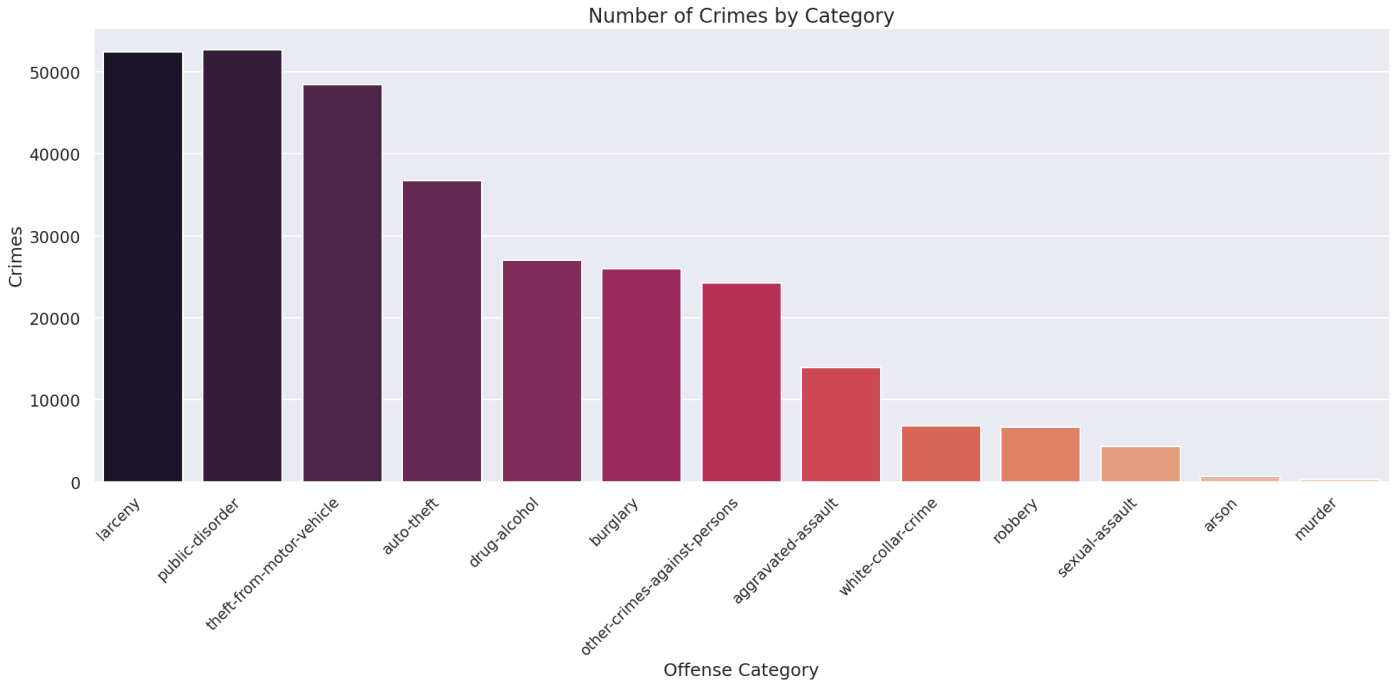



```
sns.set_context("talk")
sns.set_style("darkgrid")
plt.figure(figsize =(20,10))
plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontsize='small'
)

ax = sns.countplot(x='NEIGHBORHOOD', data=combined,
                   order = crime['NEIGHBORHOOD'].value_counts().head(78).index, palette = 'rocket')
plt.xlabel('Neighboorhood')
plt.ylabel('Crimes')
plt.title('Number of Crimes by Neighborhood', fontsize=20)
plt.show()
```


```
sns.set_context("talk")
sns.set_style("darkgrid")
plt.figure(figsize =(20,10))
plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontsize='small'
)

ax = sns.countplot(x="OFFENSE_TYPE_ID", data=combined,
                   order = crime['OFFENSE_TYPE_ID'].value_counts().head(50).index, palette = 'rocket')
plt.xlabel('Offense Type')
plt.ylabel('Crimes')
plt.title('Number of Crimes by Offense Type', fontsize=20)
plt.tight_layout()
plt.show()
```



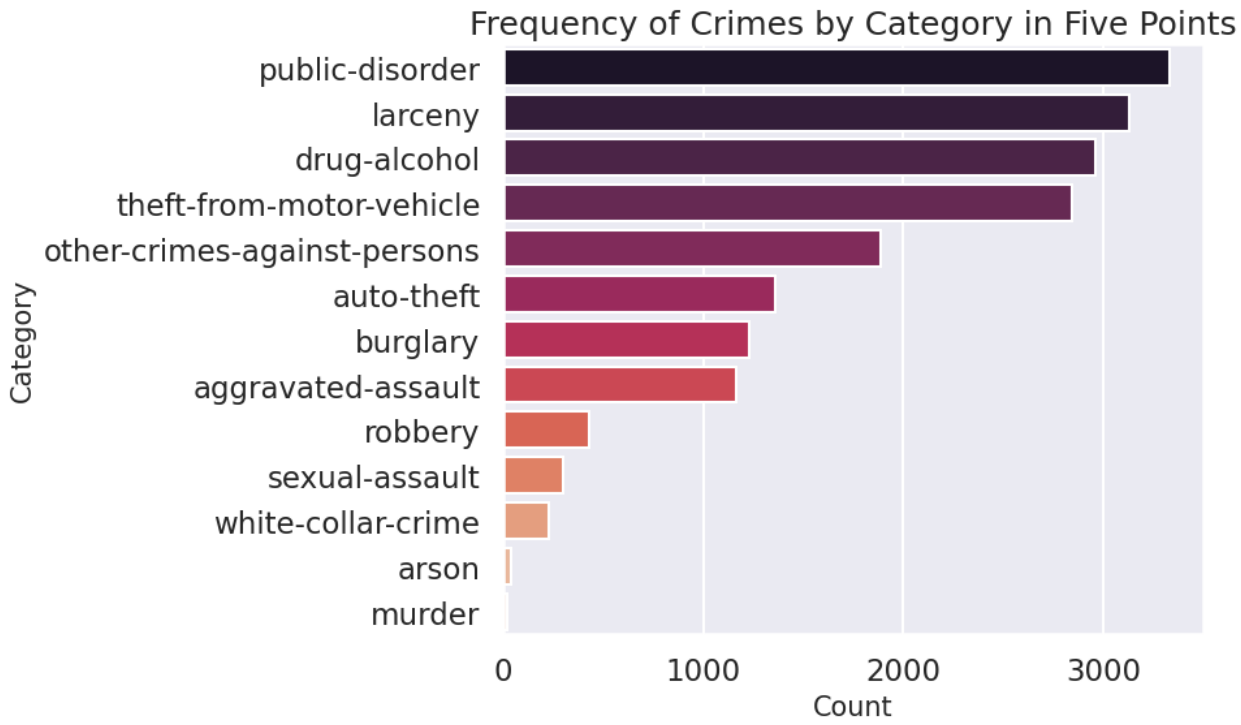
```
sns.set_context("talk")
sns.set_style("darkgrid")
plt.figure(figsize =(20,10))
plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontsize='small'
)

ax = sns.countplot(x="OFFENSE_CATEGORY_ID", data=combined,
                   order = crime['OFFENSE_CATEGORY_ID'].value_counts().head(50).index, palette = 'rocket')
plt.xlabel('Offense Category')
plt.ylabel('Crimes')
plt.title('Number of Crimes by Category', fontsize=20)
plt.tight_layout()
```




```
five_points = combined[combined['NEIGHBORHOOD'] == 'five points']
five_points = pd.DataFrame(five_points.OFFENSE_CATEGORY_ID.value_counts())
five_points.rename(columns={'OFFENSE_CATEGORY_ID': 'Count'}, inplace=True)
five_points['Crime_Category'] = five_points.index
five_points = five_points.reset_index(drop=True)
five_points.sort_values(by=['Count'], ascending=True, inplace=True)
five_points
```

	Count	Crime_Category
12	24	murder
11	40	arson
10	229	white-collar-crime
9	299	sexual-assault
8	432	robbery
7	1165	aggravated-assault
6	1233	burglary
5	1362	auto-theft
4	1887	other-crimes-against-persons
3	2349	theft-from-motor-vehicle

```
plt.figure(figsize=(10,6))
sns.barplot(x='Count',
            y='Crime_Category',
            data=five_points,
            palette = 'rocket',
            order=five_points.sort_values('Count', ascending=False).Crime_Category,
           )
plt.xlabel("Count", size=15)
plt.ylabel("Category", size=15)
plt.title("Frequency of Crimes by Category in Five Points", size=18)
plt.tight_layout()
```



Time Series Analysis

```
# copy dataframe setting reported date as the index
crime_time = combined.copy(deep=True)

crime_time['DATE'] = pd.to_datetime(crime_time.FIRST_OCCURRENCE_DATE).dt.date

crime_time = crime_time.set_index('FIRST_OCCURRENCE_DATE')

crime_time.head()
```

FIRST_OCCURRENCE_DATE	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	GEO_X	
2021-04-18 22:30:00	2021224206220200	2202	0	burglary-residence-by-force	burglary	3142828.0	169
2021-04-21 13:15:00	2021224336352000	3520	0	drug-opium-or-deriv-sell	drug-alcohol	3144130.0	169
2021-04-21 13:25:00	2021224464359900	3599	0	drug-pcs-other-drug	drug-alcohol	3144130.0	169
2021-04-21 13:20:00	2021224359351000	3510	0	drug-heroin-sell	drug-alcohol	3144130.0	169

```
crime_time.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 300437 entries, 2021-04-18 22:30:00 to 2021-06-10 20:30:00
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              300437 non-null  int64  
 1   OFFENSE_CODE            300437 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  300437 non-null  int64  
 3   OFFENSE_TYPE_ID         300437 non-null  object 
 4   OFFENSE_CATEGORY_ID     300437 non-null  object 
 5   GEO_X                   296099 non-null  float64
 6   GEO_Y                   296099 non-null  float64
 7   GEO_LON                 300437 non-null  float64
 8   GEO_LAT                 300437 non-null  float64
 9   DISTRICT_ID             300437 non-null  float64
 10  PRECINCT_ID             300437 non-null  float64
 11  NEIGHBORHOOD            300437 non-null  object 
 12  TTL_POPULATION_ALL      300437 non-null  float64
 13  MALE                    300437 non-null  float64
 14  FEMALE                  300437 non-null  float64
 15  OCCUPIED_HU             300437 non-null  int64  
 16  VACANT_HU               300437 non-null  int64  
 17  OWNER_OCCUPIED_HU       300437 non-null  int64  
 18  MED_HH_INCOME           300437 non-null  int64  
 19  MED_FAMILY_INCOME       300437 non-null  int64  
 20  AVG_HH_INCOME           300437 non-null  int64  
 21  AVG_FAM_INCOME          300437 non-null  int64  
 22  MEDIAN_EARNINGS         300437 non-null  int64  
 23  MED_CONTRACT_RENT       300437 non-null  int64  
 24  MED_GROSS_RENT          300437 non-null  int64  
 25  MEDIAN_HOME_VALUE       300437 non-null  int64  
 26  PCT_POVERTY             300437 non-null  float64
 27  PCT_FAM_POVERTY         300437 non-null  float64
 28  DATE                    300437 non-null  object 
dtypes: float64(11), int64(14), object(4)
memory usage: 68.8+ MB
```

```
crime_time['DATE'] = pd.to_datetime(crime_time['DATE'])
```

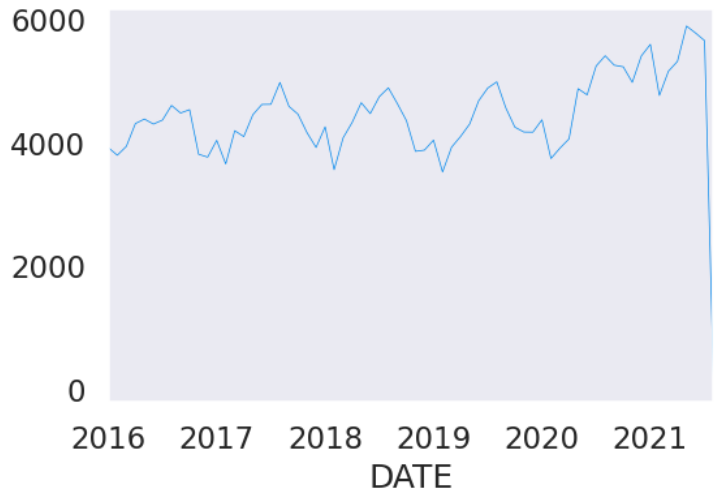
```
crime_time.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 300437 entries, 2021-04-18 22:30:00 to 2021-06-10 20:30:00
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              300437 non-null  int64  
 1   OFFENSE_CODE            300437 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  300437 non-null  int64  
 3   OFFENSE_TYPE_ID         300437 non-null  object 
 4   OFFENSE_CATEGORY_ID     300437 non-null  object 
 5   GEO_X                   296099 non-null  float64
 6   GEO_Y                   296099 non-null  float64
 7   GEO_LON                 300437 non-null  float64
 8   GEO_LAT                 300437 non-null  float64
 9   DISTRICT_ID             300437 non-null  float64
 10  PRECINCT_ID             300437 non-null  float64
 11  NEIGHBORHOOD            300437 non-null  object 
 12  TTL_POPULATION_ALL      300437 non-null  float64
 13  MALE                    300437 non-null  float64
 14  FEMALE                  300437 non-null  float64
 15  OCCUPIED_HU             300437 non-null  int64  
 16  VACANT_HU               300437 non-null  int64  
 17  OWNER_OCCUPIED_HU       300437 non-null  int64  
 18  MED_HH_INCOME           300437 non-null  int64  
 19  MED_FAMILY_INCOME       300437 non-null  int64  
 20  AVG_HH_INCOME           300437 non-null  int64  
 21  AVG_FAM_INCOME          300437 non-null  int64  
 22  MEDIAN_EARNINGS         300437 non-null  int64  
```

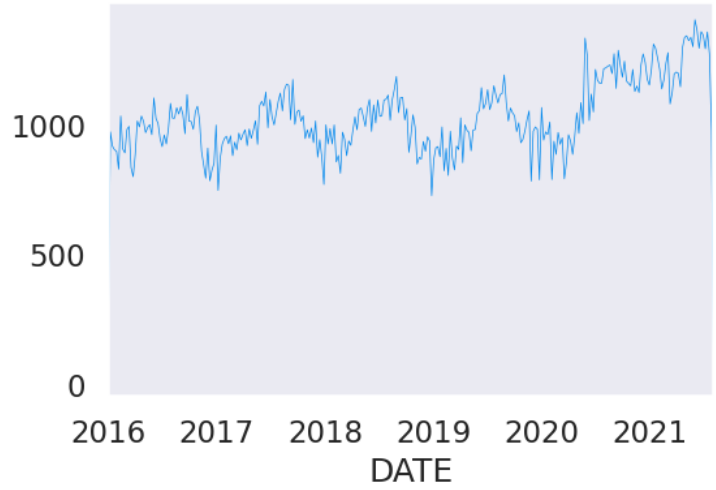
```
 23  MED_CONTRACT_RENT       300437 non-null  int64
 24  MED_GROSS_RENT          300437 non-null  int64
 25  MEDIAN_HOME_VALUE       300437 non-null  int64
 26  PCT_POVERTY             300437 non-null  float64
 27  PCT_FAM_POVERTY         300437 non-null  float64
 28  DATE                    300437 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(11), int64(14), object(3)
memory usage: 68.8+ MB
```

```
# get count of number of crimes by year and month
monthly = crime_time['DATE'].groupby(crime_time.DATE.dt.to_period("M")).count()
weekly = crime_time['DATE'].groupby(crime_time.DATE.dt.to_period("W")).count()
daily = crime_time['DATE'].groupby(crime_time.DATE.dt.to_period("D")).count()
```

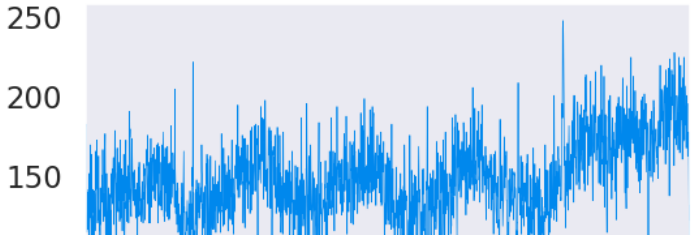
```
monthly.plot(linewidth=0.5)
plt.grid(b=None);
```




```
weekly.plot(linewidth=0.5)
plt.grid(b=None);
```




```
daily.plot(linewidth=0.5)
plt.grid(b=None);
```




```
car_theft = crime_time[crime_time.OFFENSE_TYPE_ID.str.contains('theft-of-motor-vehicle')]
car_theft.head()
```

FIRST_OCCURRENCE_DATE	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	GEO_X	
2021-03-26 19:00:00	2021176324240400	2404	0	theft-of-motor-vehicle	auto-theft	3142653.0	169
2021-01-25 08:00:00	2021219442240400	2404	0	theft-of-motor-vehicle	auto-theft	3142327.0	169
2021-04-10 22:00:00	2021204670240400	2404	0	theft-of-motor-vehicle	auto-theft	3143065.0	169
2021-07-17 21:00:00	2021408936240400	2404	0	theft-of-motor-vehicle	auto-theft	3143358.0	169
2016-08-05 08:00:00	2016504483240400	2404	0	theft-of-motor-vehicle	auto-theft	3141939.0	169

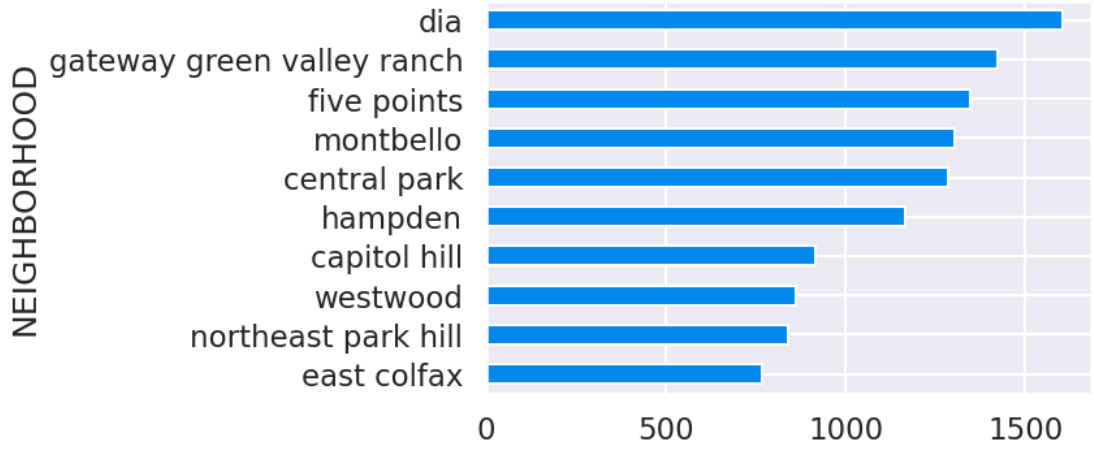
```
car_theft.groupby('NEIGHBORHOOD').size().sort_values(ascending = False)
```

```
NEIGHBORHOOD
dia                          1602
gateway green valley ranch   1423
five points                  1347
montbello                    1305
central park                 1285
                             ...
skyland                        88
indian creek                   88
auraria                        87
country club                   81
wellshire                      41
Length: 78, dtype: int64
```

```
top_10_car_theft_areas = car_theft.groupby('NEIGHBORHOOD').size().sort_values(ascending = True).tail(10)
```

```
top_10_car_theft_areas.plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3244cb7e10>
```



Geographical Visualization

```
nbhd = pd.read_csv('/content/drive/MyDrive/statistical_neighborhoods.csv')
```

```
nbhd.columns = ['NEIGHBORHOODS']
neighborhoods = nbhd.NEIGHBORHOODS.unique()
neighborhoods
```

```
array(['Auraria', 'Cory - Merrill', 'Belcaro', 'Washington Park',
       'Washington Park West', 'Speer', 'Cherry Creek', 'Country Club',
       'Congress Park', 'City Park', 'Marston', 'Fort Logan', 'Barnum',
       'Barnum West', 'West Colfax', 'West Highland', 'Sloan Lake',
       'Berkeley', 'Regis', 'Chaffee Park', 'Highland', 'Athmar Park',
       'Wellshire', 'University', 'Rosedale', 'Cheesman Park', 'Hilltop',
       'Montclair', 'Hale', 'North Park Hill', 'South Park Hill',
       'University Park', 'Platt Park', 'Overland', 'Ruby Hill',
       'Kennedy', 'Hampden', 'Southmoor Park', 'Hampden South',
       'Indian Creek', 'Goldsmith', 'University Hills', 'Harvey Park',
       'Mar Lee', 'East Colfax', 'Capitol Hill', 'North Capitol Hill',
       'Civic Center', 'CBD', 'Union Station', 'Central Park',
       'Montbello', 'Lowry Field', 'Gateway - Green Valley Ranch',
       'Harvey Park South', 'College View - South Platte',
       'City Park West', 'Sun Valley', 'Valverde', 'Villa Park',
       'Five Points', 'Globeville', 'Bear Valley', 'Virginia Village',
       'Windsor', 'Washington Virginia Vale', 'Jefferson Park',
       'Northeast Park Hill', 'Elyria Swansea', 'Baker', 'Clayton',
       'Skyland', 'Lincoln Park', 'Whittier', 'Cole', 'Westwood',
       'Sunnyside', 'DIA'], dtype=object)
```

```
crime_freq= combined.NEIGHBORHOOD.value_counts()

table = pd.DataFrame(data=crime_freq.values, index=crime_freq.index, columns=['Count'])
#table = table.reindex(neighborhoods)

table = table.reset_index()
table.rename({'index': 'NEIGHBORHOOD'}, axis='columns', inplace=True)

table
```

	NEIGHBORHOOD	Count
0	five points	18939
1	central park	12039
2	capitol hill	12038
3	cbd	11121
4	montbello	9126
...	...	...
73	rosedale	997
74	skyland	945
75	country club	746
76	indian creek	535
77	wellshire	379

78 rows × 2 columns

Change the capitalization of names to match data. The .title() method is used to capitalize all words in the name.

```
table['NEIGHBORHOOD'] = list( map(lambda x: x.title(), table['NEIGHBORHOOD']))
table.NEIGHBORHOOD.unique()
```

```
array(['Five Points', 'Central Park', 'Capitol Hill', 'Cbd', 'Montbello',
       'Union Station', 'Gateway Green Valley Ranch', 'East Colfax',
       'Lincoln Park', 'West Colfax', 'North Capitol Hill', 'Hampden',
       'Dia', 'Civic Center', 'Baker', 'Westwood', 'Highland',
       'Northeast Park Hill', 'Speer', 'Hampden South', 'Cheesman Park',
       'Washington Virginia Vale', 'City Park West',
       'College View South Platte', 'Mar Lee', 'Elyria Swansea',
       'Virginia Village', 'Villa Park', 'Sunnyside', 'Cherry Creek',
       'Athmar Park', 'Harvey Park', 'Ruby Hill', 'Congress Park',
       'Globeville', 'Windsor', 'West Highland', 'Berkeley',
       'Lowry Field', 'Overland', 'Goldsmith', 'Harvey Park South',
```

```
       'Sloan Lake', 'University Hills', 'Hale', 'Marston', 'Barnum',
       'Bear Valley', 'University Park', 'University', 'Montclair',
       'Jefferson Park', 'South Park Hill', 'Washington Park West',
       'Valverde', 'Cole', 'Platt Park', 'Clayton', 'North Park Hill',
       'Sun Valley', 'Auraria', 'Kennedy', 'Whittier', 'Fort Logan',
       'Southmoor Park', 'Hilltop', 'Cory Merrill', 'Washington Park',
       'Barnum West', 'City Park', 'Regis', 'Chaffee Park', 'Belcaro',
       'Rosedale', 'Skyland', 'Country Club', 'Indian Creek', 'Wellshire'],
      dtype=object)
```

Reformat problem names

Several of the names contain hyphens or are abbreviations with irregular capitalization .

```
table.NEIGHBORHOOD.replace(to_replace='Gateway Green Valley Ranch', value='Gateway - Green Valley Ranch', inplace=True )
table.NEIGHBORHOOD.replace(to_replace='College View South Platte', value='College View - South Platte', inplace=True )
table.NEIGHBORHOOD.replace(to_replace='Cory Merrill', value='Cory - Merrill', inplace=True )
table.NEIGHBORHOOD.replace(to_replace='Dia', value='DIA', inplace=True )
table.NEIGHBORHOOD.replace(to_replace='Cbd', value='CBD', inplace=True )
```

```
table.NEIGHBORHOOD.unique()
```

```
array(['Five Points', 'Central Park', 'Capitol Hill', 'CBD', 'Montbello',
       'Union Station', 'Gateway - Green Valley Ranch', 'East Colfax',
       'Lincoln Park', 'West Colfax', 'North Capitol Hill', 'Hampden',
       'DIA', 'Civic Center', 'Baker', 'Westwood', 'Highland',
       'Northeast Park Hill', 'Speer', 'Hampden South', 'Cheesman Park',
       'Washington Virginia Vale', 'City Park West',
       'College View - South Platte', 'Mar Lee', 'Elyria Swansea',
       'Virginia Village', 'Villa Park', 'Sunnyside', 'Cherry Creek',
       'Athmar Park', 'Harvey Park', 'Ruby Hill', 'Congress Park',
       'Globeville', 'Windsor', 'West Highland', 'Berkeley',
       'Lowry Field', 'Overland', 'Goldsmith', 'Harvey Park South',
       'Sloan Lake', 'University Hills', 'Hale', 'Marston', 'Barnum',
       'Bear Valley', 'University Park', 'University', 'Montclair',
       'Jefferson Park', 'South Park Hill', 'Washington Park West',
       'Valverde', 'Cole', 'Platt Park', 'Clayton', 'North Park Hill',
       'Sun Valley', 'Auraria', 'Kennedy', 'Whittier', 'Fort Logan',
       'Southmoor Park', 'Hilltop', 'Cory - Merrill', 'Washington Park',
       'Barnum West', 'City Park', 'Regis', 'Chaffee Park', 'Belcaro',
       'Rosedale', 'Skyland', 'Country Club', 'Indian Creek', 'Wellshire'],
      dtype=object)
```

```
table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   NEIGHBORHOOD  78 non-null     object
 1   Count         78 non-null     int64 
dtypes: int64(1), object(1)
memory usage: 1.3+ KB
```

Crime Maps

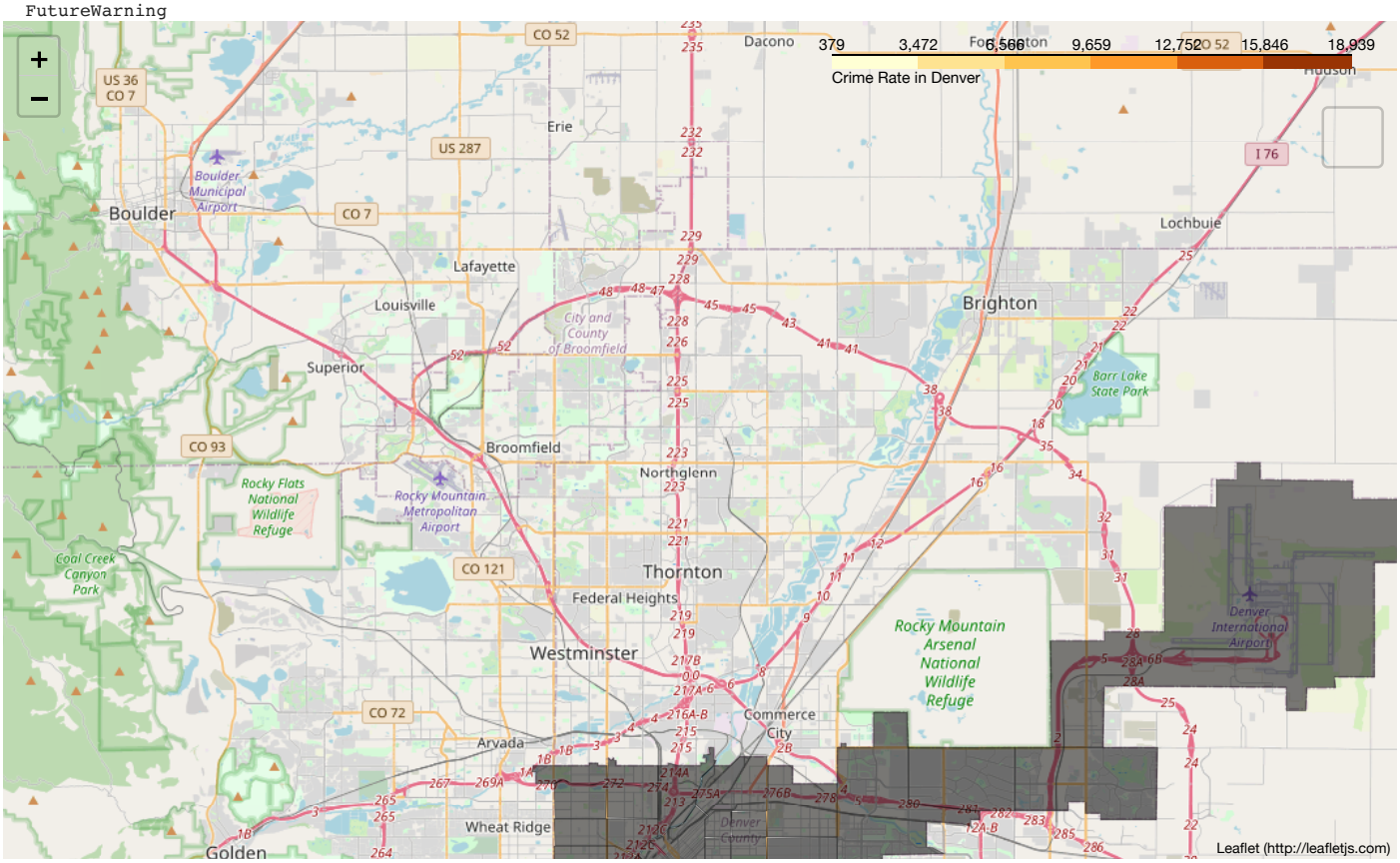
Maps of the crime reports will be created using the folium library.

```
from folium.plugins import MarkerCluster
den_lat_lon = [39.7348, -104.9653]
geojson = r'/content/drive/MyDrive/denver statistical neighborhoods.geojson'
den_map = folium.Map(location = den_lat_lon, zoom_start = 10)
```

```
#generate map 1
den_map.choropleth(
    geo_data=geojson,
    data=table,
    columns=['NEIGHBORHOOD', 'Count'],
    key_on='feature.properties.Count',
    fill_color='YlOrBr',
    fill_opacity=0.5,
    line_opacity=0.2,
    highlight=True,
    legend_name='Crime Rate in Denver',
    tiles='Mapbox Bright',
    reset=True,
```

```
    #style_function=style_function,
    tooltip=folium.GeoJsonTooltip(
        fields=['NEIGHBORHOOD', 'Count'],
        aliases=['NEIGHBORHOOD', 'Count'],
        localize=True)
)
folium.LayerControl().add_to(den_map)


den_map
```

```
/usr/local/lib/python3.7/dist-packages/folium/folium.py:426: FutureWarning: The choropleth  method has been deprecated. Instead
  FutureWarning
```



MODELING

```
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score, roc_curve, auc, r2_score
from sklearn.metrics import plot_confusion_matrix,confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn import preprocessing
from sklearn.utils import class_weight
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
import xgboost as xgb
from xgboost import XGBClassifier
import warnings
warnings.filterwarnings('always')
```

In this section we will work on preparing our dataset for modeling to see how effective it will be in predicting crime in Denver based on its location in Denver's neighborhood as well as certain demographic features that we will choose. It is important to note that while we have a number of good items in our crime and demographic data we may need to reduce some areas in order to reduce the computational complexity and time constraints. We will be using the combined dataframe we created earlier for our models.

```
model_data = combined
```

```
print(model_data.info())
print(model_data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 300437 entries, 0 to 300436
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   OFFENSE_ID              300437 non-null  int64  
 1   OFFENSE_CODE            300437 non-null  int64  
 2   OFFENSE_CODE_EXTENSION  300437 non-null  int64  
 3   OFFENSE_TYPE_ID         300437 non-null  object 
 4   OFFENSE_CATEGORY_ID     300437 non-null  object 
 5   FIRST_OCCURRENCE_DATE   300437 non-null  object 
 6   GEO_X                   296099 non-null  float64
 7   GEO_Y                   296099 non-null  float64
 8   GEO_LON                 300437 non-null  float64
 9   GEO_LAT                 300437 non-null  float64
 10  DISTRICT_ID             300437 non-null  float64
 11  PRECINCT_ID             300437 non-null  float64
 12  NEIGHBORHOOD            300437 non-null  object 
 13  TTL_POPULATION_ALL      300437 non-null  float64
 14  MALE                    300437 non-null  float64
 15  FEMALE                  300437 non-null  float64
 16  OCCUPIED_HU             300437 non-null  int64  
 17  VACANT_HU               300437 non-null  int64  
 18  OWNER_OCCUPIED_HU       300437 non-null  int64  
 19  MED_HH_INCOME           300437 non-null  int64  
 20  MED_FAMILY_INCOME       300437 non-null  int64  
 21  AVG_HH_INCOME           300437 non-null  int64  
 22  AVG_FAM_INCOME          300437 non-null  int64  
 23  MEDIAN_EARNINGS         300437 non-null  int64  
 24  MED_CONTRACT_RENT       300437 non-null  int64  
 25  MED_GROSS_RENT          300437 non-null  int64  
 26  MEDIAN_HOME_VALUE       300437 non-null  int64  
 27  PCT_POVERTY             300437 non-null  float64
 28  PCT_FAM_POVERTY         300437 non-null  float64
dtypes: float64(11), int64(14), object(4)
memory usage: 78.8+ MB
None
          OFFENSE_ID   OFFENSE_CODE  ...    PCT_POVERTY  PCT_FAM_POVERTY
count  3.004370e+05  300437.000000  ...  300437.000000    300437.000000
mean   5.692249e+15    2467.598132  ...      14.402714         9.935046
std    1.435704e+16     858.410837  ...       8.806224         8.728223
min    2.021323e+10     902.000000  ...       0.000000         0.000000
25%    2.017289e+15    2204.000000  ...       8.800000         4.516378
50%    2.019269e+15    2399.000000  ...      13.450000         7.201646
75%    2.021248e+15    2999.000000  ...      18.000000        13.178069
max    2.020890e+18    5399.000000  ...      76.800000        72.316384

[8 rows x 25 columns]
```

```
model_data['NEIGHBORHOOD'].value_counts()
```

```
five points     18939
central park    12039
capitol hill    12038
cbd             11121
```

```
montbello        9126
                ...
rosedale          997
skyland           945
country club      746
indian creek      535
wellshire         379
Name: NEIGHBORHOOD, Length: 78, dtype: int64
```

```
model_data.head()
```

	OFFENSE_ID	OFFENSE_CODE	OFFENSE_CODE_EXTENSION	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	FIRST_OCCURRENCE_DATE	GEO_X
0	2021224206220200	2202	0	burglary-residence-by-force	burglary	2021-04-18 22:30:00	3142828.0
1	2021224336352000	3520	0	drug-opium-or-deriv-sell	drug-alcohol	2021-04-21 13:15:00	3144130.0
2	2021224464359900	3599	0	drug-pcs-other-drug	drug-alcohol	2021-04-21 13:25:00	3144130.0
3	2021224359351000	3510	0	drug-heroin-sell	drug-alcohol	2021-04-21 13:20:00	3144130.0
4	2021224359220600	2206	0	burglary-poss-of-tools	burglary	2021-04-21 13:20:00	3144130.0

In this section we will be creating dictionaires that will combine some features in our data set. First we will work on the neighborhood data. The first course of action was to group the neighborhoods that had less than 3000 instances of crimes in them. The reasoning is that comparatively speaking, some neighborhoods have very small number of instances of crime compared to other nieghborhoods such as Five Points. We will join the neighborhoods with smaller number of instances into the group that will be named "other neighborhood" This will also hopefully reduce complexity and increase model performance since instead of having 78 columns of neighborhoods for modeling, we will slim it down to a workable number that would not overtax our system.

```
other_nbhd_dict = {'indian creek': 'other neighborhood', 'country club':'other neighborhood',
                   'stapleton':'other neighboorhood','skyland':'other neighborhood',
                   'rosedale':'other neighborhood','belcaro':'other neighborhood',
                   'chaffee park':'other neighborhood','regis':'other neighborhood',
                   'city park':'other neighborhood','barnum west':'other neighborhood',
                   'washington park':'other neighborhood','cory merrill':'other neighborhood',
                   'hilltop':'other neighborhood','southmore park':'other neighborhood',
                   'fort logan':'other neighborhood','whittier':'other neighborhood',
                   'kennedy':'other neighborhood','auraria':'other neighborhood',
                   'sun valley':'other neighborhood','north park hill':'other neighborhood',
                   'jefferson park':'other neighborhood','montclair':'other neighborhood',
                   'university':'other neighborhood','university park':'other neighborhood',
                   'bear valley':'other neighborhood','barnum':'other neighborhood',
                   'marston':'other neighborhood','hale':'other neighborhood',
                   'university hills':'other neighborhood','sloan lake':'other neighborhood',
                   'harvey park south':'other neighborhood','goldsmith':'other neighborhood',
                   'overland':'other neighborhood','lowry field':'other neighborhood',
                   'berkeley':'other neighborhood','west highland':'other neighborhood',
                   'windsor':'other neighborhood','globeville':'other neighborhood',
                   'south park hill':'other neighborhood','washington park west':'other neighborhood',
                   'valverde':'other neighborhood','cole':'other neighborhood',
                   'platt park':'other neighborhood','clayton':'other neighborhood',
                   'southmoor park':'other neighborhood','wellshire':'other neighborhood'}
```

```
model_data['NEIGHBORHOOD'] = model_data['NEIGHBORHOOD'].map(other_nbhd_dict).fillna(model_data['NEIGHBORHOOD'])
model_data['NEIGHBORHOOD'].value_counts()
```

```
other neighborhood            90529
five points                   18939
central park                  12039
capitol hill                  12038
cbd                           11121
montbello                      9126
union station                  8647
gateway green valley ranch     7905
east colfax                    7253
lincoln park                   7028
west colfax                    6456
north capitol hill             6334
hampden                        6277
dia                            6081
civic center                   5983
baker                          5608
westwood                       5579
highland                       5285
northeast park hill            5210
```

```
speer                          4960
hampden south                  4636
cheesman park                  4536
washington virginia vale       4514
city park west                 4248
college view south platte      4083
mar lee                        3910
elyria swansea                 3766
virginia village               3756
villa park                     3679
sunnyside                      3675
cherry creek                   3622
athmar park                    3517
harvey park                    3478
ruby hill                      3462
congress park                  3157
Name: NEIGHBORHOOD, dtype: int64
```

In this section we will follow the previous methodology that we used with neighborhoods and use with our Offsense Category ID. As we can see public disorder has the most instances, while arson, murder, sexual assault, robbery and white color crime all have comparatively smaller number of instances. We will be combining the aforementioned categories to one inclusive category called "other-crime" in order to balance our data. It is the hope that with this reassignment, it will make our datasets easier to work with as our predicted feature.

```
model_data['OFFENSE_CATEGORY_ID'].value_counts()
```

```
public-disorder                52721
larceny                        52470
theft-from-motor-vehicle       48448
auto-theft                     36774
drug-alcohol                   27016
burglary                       25958
other-crimes-against-persons   24249
aggravated-assault             13958
white-collar-crime              6810
robbery                         6649
sexual-assault                  4338
arson                            691
murder                           355
Name: OFFENSE_CATEGORY_ID, dtype: int64
```

```
other_crime_dict = {'white-collar-crime': 'other-crime', 'robbery':'other-crime',
                    'sexual-assault': 'other-crime', 'arson':'other-crime',
                    'murder':'other-crime'}
model_data['OFFENSE_CATEGORY_ID'] = model_data['OFFENSE_CATEGORY_ID'].map(other_crime_dict).fillna(model_data['OFFENSE_CATEGORY_ID'])
model_data['OFFENSE_CATEGORY_ID'].value_counts()
```

```
public-disorder                52721
larceny                        52470
theft-from-motor-vehicle       48448
auto-theft                     36774
drug-alcohol                   27016
burglary                       25958
other-crimes-against-persons   24249
other-crime                    18843
aggravated-assault             13958
Name: OFFENSE_CATEGORY_ID, dtype: int64
```

We will create a new version of our model data and drop some columns to try to make our model more efficient. We had to remove some of our demographic data in order to reduce model complexity. We will be keeping our nrighborhoods, occupied/vacant units, median earnings, home value and percent of poverty and family poverty in the neighborhoods.

```
model_data_2 = model_data.drop(columns=['OFFENSE_ID', 'OFFENSE_CODE', 'OFFENSE_CODE_EXTENSION',
                                        'OFFENSE_TYPE_ID','GEO_X','GEO_Y',
                                        'FIRST_OCCURRENCE_DATE', 'TTL_POPULATION_ALL','FEMALE','MALE',
                                        'MED_CONTRACT_RENT', 'MED_GROSS_RENT', 'MED_HH_INCOME',
                                        'MED_FAMILY_INCOME', 'AVG_HH_INCOME', 'AVG_FAM_INCOME'])
model_data_2.head()
```

	OFFENSE_CATEGORY_ID	GEO_LON	GEO_LAT	DISTRICT_ID	PRECINCT_ID	NEIGHBORHOOD	OCCUPIED_HU	VACANT_HU	OWNER_OCCUPIED_HU	MI
0	burglary	-104.992161	39.733543	6.0	611.0	civic center	1492	206	385	
1	drug-alcohol	-104.987485	39.739897	6.0	611.0	civic center	1492	206	385	
2	drug-alcohol	-104.987485	39.739897	6.0	611.0	civic center	1492	206	385	
3	drug-alcohol	-104.987485	39.739897	6.0	611.0	civic center	1492	206	385	
4	burglary	-104.987485	39.739897	6.0	611.0	civic center	1492	206	385	

```
model_data_2['NEIGHBORHOOD'].value_counts()
```

```
other neighborhood            90529
five points                   18939
central park                  12039
capitol hill                  12038
cbd                           11121
montbello                      9126
union station                  8647
gateway green valley ranch     7905
east colfax                    7253
lincoln park                   7028
west colfax                    6456
north capitol hill             6334
hampden                        6277
dia                            6081
civic center                   5983
baker                          5608
westwood                       5579
highland                       5285
northeast park hill            5210
speer                          4960
hampden south                  4636
cheesman park                  4536
washington virginia vale       4514
city park west                 4248
college view south platte      4083
mar lee                        3910
elyria swansea                 3766
virginia village               3756
villa park                     3679
sunnyside                      3675
cherry creek                   3622
athmar park                    3517
harvey park                    3478
ruby hill                      3462
congress park                  3157
Name: NEIGHBORHOOD, dtype: int64
```

We will be using the offense category ID as our dependent variable in our dataset while selecting all other values for our X values.

```
X = model_data_2.drop(['OFFENSE_CATEGORY_ID'],axis=1)
y = model_data_2['OFFENSE_CATEGORY_ID']
```

At this point it will be important to encode our remaining categorical variable which is neighborhoods. This will help in modeling since it will only work with numerical values.

```
X.select_dtypes(include=[object])
```

	NEIGHBORHOOD
0	civic center
1	civic center
2	civic center
3	civic center
4	civic center
...	...
300432	other neighborhood
300433	other neighborhood
300434	other neighborhood
300435	other neighborhood
300436	other neighborhood

300437 rows × 1 columns

```
model_data_3 = pd.get_dummies(X)
```

Below we can see that our neighborhoods have been encoded and we can see that we now have 34 columns for all our neighborhoods, which is a sizable reduction from 78 total neighborhoods. This will hopefully make modeling simpler and less computationally expensive. For example,

we currentl have 46 total columns that we are working with. If we would have not grouped some of nieghborhoods as we did earlier that would

```
model_data_3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 300437 entries, 0 to 300436
Data columns (total 46 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   GEO_LON                                 300437 non-null  float64
 1   GEO_LAT                                 300437 non-null  float64
 2   DISTRICT_ID                             300437 non-null  float64
 3   PRECINCT_ID                             300437 non-null  float64
 4   OCCUPIED_HU                             300437 non-null  int64  
 5   VACANT_HU                               300437 non-null  int64  
 6   OWNER_OCCUPIED_HU                       300437 non-null  int64  
 7   MEDIAN_EARNINGS                         300437 non-null  int64  
 8   MEDIAN_HOME_VALUE                       300437 non-null  int64  
 9   PCT_POVERTY                             300437 non-null  float64
 10  PCT_FAM_POVERTY                         300437 non-null  float64
 11  NEIGHBORHOOD_athmar park                300437 non-null  uint8  
 12  NEIGHBORHOOD_baker                      300437 non-null  uint8  
 13  NEIGHBORHOOD_capitol hill               300437 non-null  uint8  
 14  NEIGHBORHOOD_cbd                        300437 non-null  uint8  
 15  NEIGHBORHOOD_central park               300437 non-null  uint8  
 16  NEIGHBORHOOD_cheesman park              300437 non-null  uint8  
 17  NEIGHBORHOOD_cherry creek               300437 non-null  uint8  
 18  NEIGHBORHOOD_city park west             300437 non-null  uint8  
 19  NEIGHBORHOOD_civic center               300437 non-null  uint8  
 20  NEIGHBORHOOD_college view south platte  300437 non-null  uint8  
 21  NEIGHBORHOOD_congress park              300437 non-null  uint8  
 22  NEIGHBORHOOD_dia                        300437 non-null  uint8  
 23  NEIGHBORHOOD_east colfax                300437 non-null  uint8  
 24  NEIGHBORHOOD_elyria swansea             300437 non-null  uint8  
 25  NEIGHBORHOOD_five points                300437 non-null  uint8  
 26  NEIGHBORHOOD_gateway green valley ranch 300437 non-null  uint8  
 27  NEIGHBORHOOD_hampden                    300437 non-null  uint8  
 28  NEIGHBORHOOD_hampden south              300437 non-null  uint8  
 29  NEIGHBORHOOD_harvey park                300437 non-null  uint8  
 30  NEIGHBORHOOD_highland                   300437 non-null  uint8  
 31  NEIGHBORHOOD_lincoln park               300437 non-null  uint8  
 32  NEIGHBORHOOD_mar lee                    300437 non-null  uint8  
 33  NEIGHBORHOOD_montbello                  300437 non-null  uint8  
 34  NEIGHBORHOOD_north capitol hill         300437 non-null  uint8  
 35  NEIGHBORHOOD_northeast park hill        300437 non-null  uint8  
 36  NEIGHBORHOOD_other neighborhood         300437 non-null  uint8  
 37  NEIGHBORHOOD_ruby hill                  300437 non-null  uint8  
 38  NEIGHBORHOOD_speer                      300437 non-null  uint8  
 39  NEIGHBORHOOD_sunnyside                  300437 non-null  uint8  
 40  NEIGHBORHOOD_union station              300437 non-null  uint8  
 41  NEIGHBORHOOD_villa park                 300437 non-null  uint8  
 42  NEIGHBORHOOD_virginia village           300437 non-null  uint8  
 43  NEIGHBORHOOD_washington virginia vale   300437 non-null  uint8  
 44  NEIGHBORHOOD_west colfax                300437 non-null  uint8  
 45  NEIGHBORHOOD_westwood                   300437 non-null  uint8  
dtypes: float64(6), int64(5), uint8(35)
memory usage: 47.5 MB
```

```
X_train, X_test, y_train, y_test = train_test_split(model_data_3, y, random_state=23)
```

Decision Trees

```
classifier = DecisionTreeClassifier(random_state=23)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```

```
y_pred = classifier.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 336  491  184  494  418   89  494  698  326]
 [ 376 2641  679  505 1050  278  390 1216 1889]
 [ 226 1015 1499  218 1212  162  241  838 1189]
 [ 293  409  176 3581  820  111  372  658  230]
 [ 301 1100 1001  541 6648  212  508 1307 1553]
```

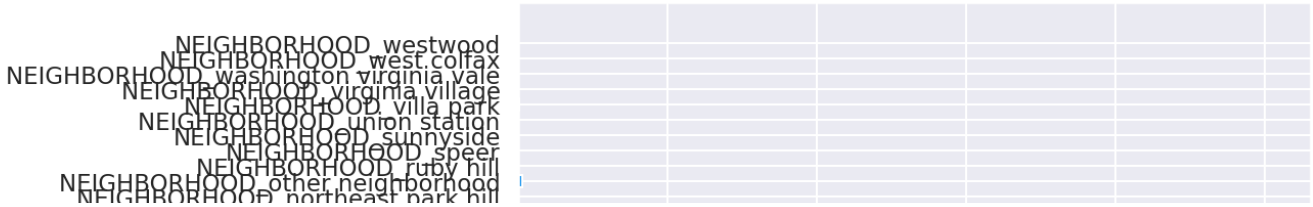
```
      [ 160  548  244  484  845 1407  227  501  361]
      [ 398  705  298  647  957  111 1104 1341  537]
      [ 605 1763  903 1083 2281  288 1215 3208 1789]
      [ 364 2483 1138  392 1887  265  406 1625 3565]]
```

	precision	recall	f1-score	support
aggravated-assault	0.11	0.10	0.10	3530
auto-theft	0.24	0.29	0.26	9024
burglary	0.24	0.23	0.24	6600
drug-alcohol	0.45	0.54	0.49	6650
larceny	0.41	0.50	0.45	13171
other-crime	0.48	0.29	0.37	4777
other-crimes-against-persons	0.22	0.18	0.20	6098
public-disorder	0.28	0.24	0.26	13135
theft-from-motor-vehicle	0.31	0.29	0.30	12125
accuracy			0.32	75110
macro avg	0.31	0.30	0.30	75110
weighted avg	0.32	0.32	0.31	75110

```
def metrics(labels, preds):
    print("Precision Score: {}".format(precision_score(labels, preds, average= 'weighted')))
    print("Recall Score: {}".format(recall_score(labels, preds,average='weighted')))
    print("Accuracy Score: {}".format(accuracy_score(labels, preds)))
    print("F1 Score: {}".format(f1_score(labels, preds,average= 'weighted')))

metrics(y_test,y_pred)
```

```
     Precision Score: 0.31560676338958277
     Recall Score: 0.31938490214352283
     Accuracy Score: 0.31938490214352283
     F1 Score: 0.3140496870993087
```

```
def plot_feature_importances(model):
  n_features = X_train.shape[1]
  plt.figure(figsize=(10,10))
  plt.barh(range(n_features), model.feature_importances_, align='center')
  plt.yticks(np.arange(n_features), X_train.columns.values)
  plt.xlabel('Feature Importance')
  plt.ylabel('Feature')

plot_feature_importances(classifier)
```

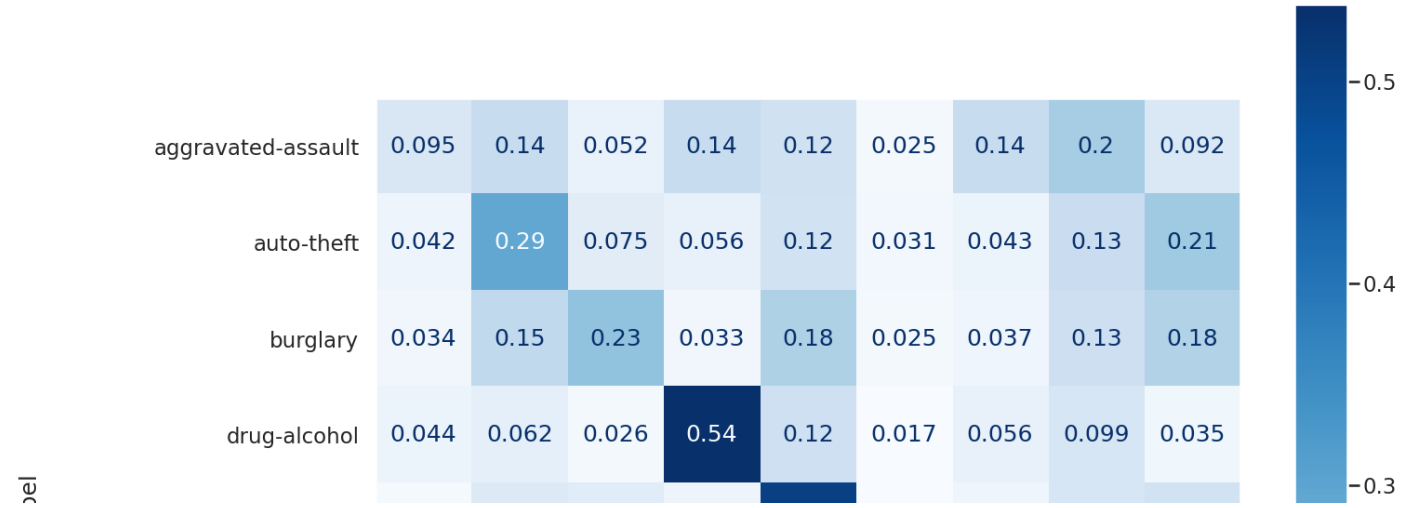


Model Performance

```
print('Testing Accuracy for Decision Tree Classifier: {:.4}%'.format(accuracy_score(y_test, y_pred)* 100))
```

```
Testing Accuracy for Decision Tree Classifier: 31.94%
```

```
fig, ax = plt.subplots(figsize=(15, 15))
plot_confusion_matrix(classifier, X_test, y_test, normalize='true', cmap=plt.cm.Blues,
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```



Random Forest

```
tree_clf = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=23)
tree_clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```

```
plot_feature_importances(tree_clf)
```

NEIGHBORHOOD_westwood

Metrics

NEIGHBORHOOD_sunnyside

```
pred = tree_clf.predict(X_test)
# Confusion matrix and classification report
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[   0    0    0  222  790   13   57 1930  518]
 [   0    0    1  212 1626   18    6 4895 2266]
 [   0    0    0  168 1525   19    4 2753 2131]
 [   0    0    0 1587 1718   20   86 2540  699]
 [   0    0    1  433 5498   23   27 4075 3114]
 [   0    0    0  225 1174  297   17 2243  821]
 [   0    0    0  363 1551   14  207 2959 1004]
 [   0    0    0  702 3106   25  114 6335 2853]
 [   0    0    3  212 2730   23    5 5254 3898]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
                            precision    recall  f1-score   support

          aggravated-assault       0.00      0.00      0.00      3530
                 auto-theft       0.00      0.00      0.00      9024
                    burglary       0.00      0.00      0.00      6600
               drug-alcohol       0.38      0.24      0.29      6650
                    larceny       0.28      0.42      0.33     13171
              other-crime       0.66      0.06      0.11      4777
other-crimes-against-persons       0.40      0.03      0.06      6098
            public-disorder       0.19      0.48      0.27     13135
    theft-from-motor-vehicle       0.23      0.32      0.26     12125

                    accuracy                           0.24     75110
                   macro avg       0.24      0.17      0.15     75110
                weighted avg       0.23      0.24      0.19     75110

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
```

Bagged Trees

```
bagged_tree = BaggingClassifier(DecisionTreeClassifier(criterion='gini', max_depth=5,
                                                       random_state=23), n_estimators=20)
```

```
bagged_tree.fit(X_train, y_train)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                        class_weight=None,
                                                        criterion='gini',
                                                        max_depth=5,
                                                        max_features=None,
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease=0.0,
                                                        min_impurity_split=None,
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_leaf=0.0,
                                                        presort='deprecated',
                                                        random_state=23,
                                                        splitter='best'),
                  bootstrap=True, bootstrap_features=False, max_features=1.0,
                  max_samples=1.0, n_estimators=20, n_jobs=None,
                  oob_score=False, random_state=None, verbose=0,
                  warm_start=False)
```

```
bagged_tree.score(X_train, y_train)
```

```
0.24460450811487305
```

```
bagged_tree.score(X_test, y_test)
```

```
0.24259086672879776
```

```
fig, ax = plt.subplots(figsize=(15, 15))
plot confusion matrix(bagged tree, X test, y test, normalize='true', cmap=plt.cm.Blues,
```

```
    _         _             _       _       _
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```

Instantiate and fit the RandomForestClassifier

```
forest = RandomForestClassifier(n_estimators=100, max_depth= 5, random_state = 42)
forest.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

```
forest.score(X_train, y_train)
```

```
0.24588708854242944
```

```
forest.score(X_test, y_test)
```

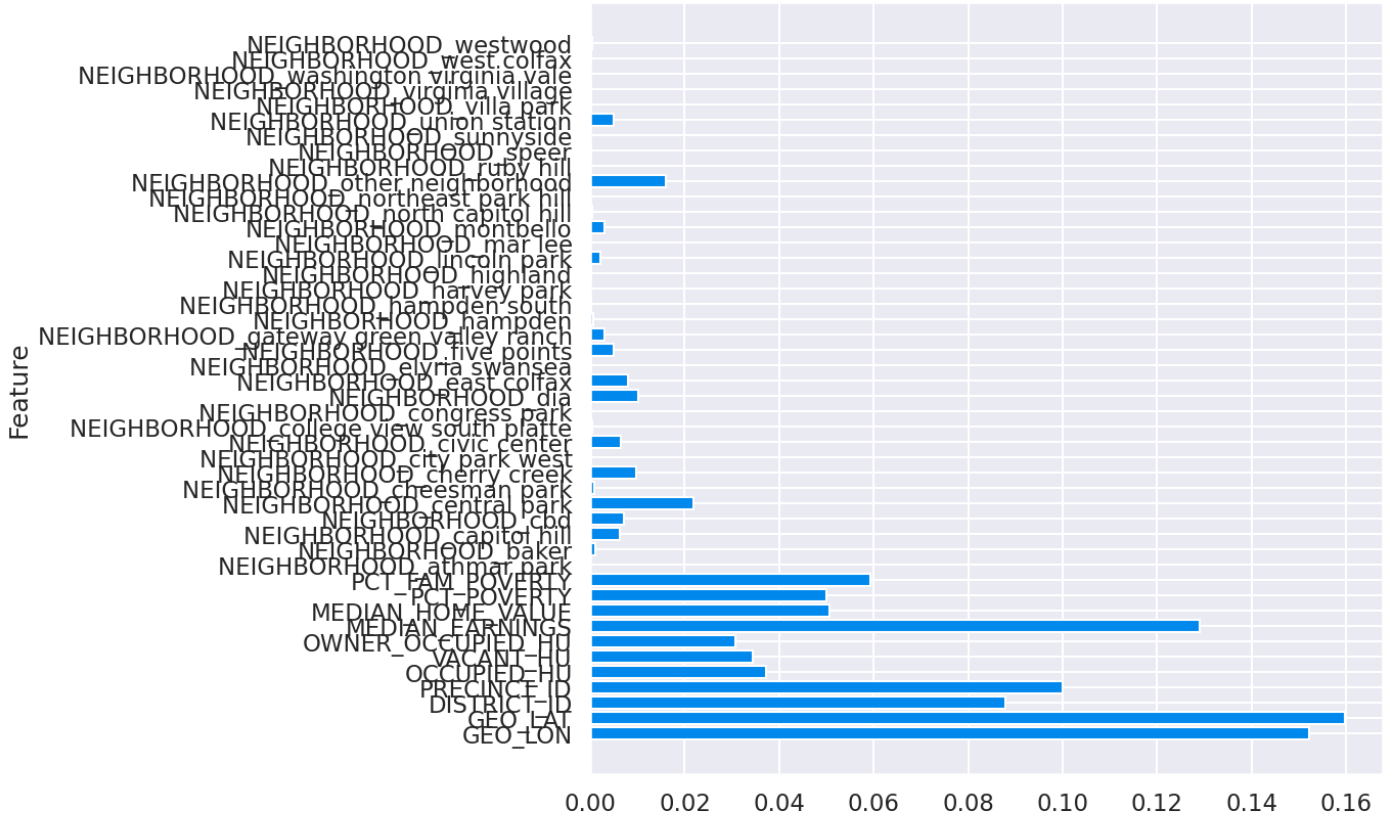
```
0.2452270003994142
```

```
pred_f = forest.predict(X_test)

print(confusion_matrix(y_test, pred_f))
print(classification_report(y_test, pred_f))
```

```
[[   0   13    0  311  514    0    0 1946  746]
 [   0  379    0  171 1106    0    0 4162 3206]
 [   0   32    0  143 1100    0    0 2668 2657]
 [   0   25    0 1633 1033    0    0 3098  861]
 [   0   63    0  556 4834    0    0 4197 3521]
 [   0  171    0  361  789  270    0 1991 1195]
 [   0   21    0  516 1126    0    0 2988 1447]
 [   0   98    0  705 2311    0    0 6369 3652]
 [   0  161    0  248 1961    0    0 4821 4934]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
                            precision    recall  f1-score   support

          aggravated-assault       0.00      0.00      0.00      3530
                  auto-theft       0.39      0.04      0.08      9024
                    burglary       0.00      0.00      0.00      6600
               drug-alcohol        0.35      0.25      0.29      6650
                    larceny        0.33      0.37      0.35     13171
                other-crime        1.00      0.06      0.11      4777
  other-crimes-against-persons       0.00      0.00      0.00      6098
             public-disorder       0.20      0.48      0.28     13135
      theft-from-motor-vehicle       0.22      0.41      0.29     12125

                   accuracy                           0.25     75110
                  macro avg       0.28      0.18      0.15     75110
               weighted avg       0.27      0.25      0.20     75110

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
```

```
fig, ax = plt.subplots(figsize=(15, 15))
plot_confusion_matrix(forest, X_test, y_test, normalize='true', cmap=plt.cm.Blues,
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```


```
plot_feature_importances(forest)
```



XGBoost

```
import time
start_time = time.time()
xgb_clf = XGBClassifier()

xgb_clf.fit(X_train, y_train)

training_preds = xgb_clf.predict(X_train)
test_preds = xgb_clf.predict(X_test)

training_accuracy = accuracy_score(y_train, training_preds)
test_accuracy = accuracy_score(y_test, test_preds)

print('Training Accuracy: {:.4}%'.format(training_accuracy * 100))
print('Validation Accuracy: {:.4}%'.format(test_accuracy * 100))
```

```
Training Accuracy: 27.59%
Validation Accuracy: 27.18%
```

```
xgb_clf.score(X_train, y_train)
```

```
0.27586574178860057
```

```
xgb_clf.score(X_test, y_test)
```

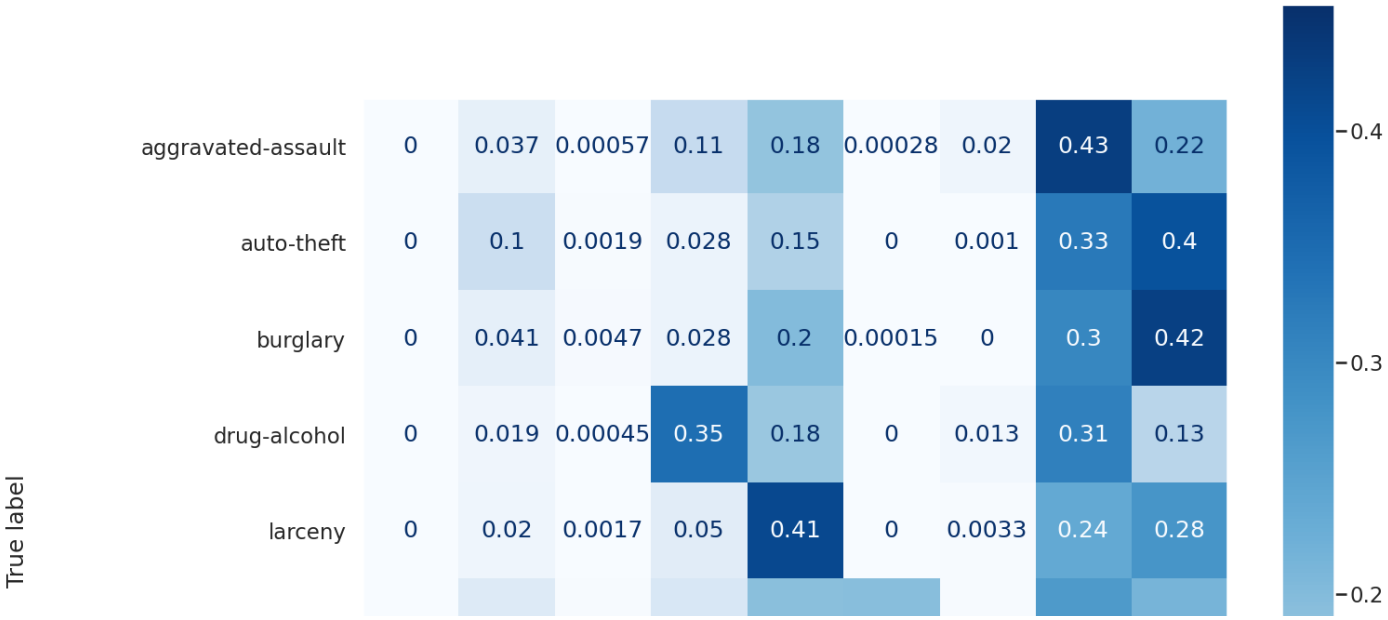
```
0.27181467181467184
```

```
pred_xgb = xgb_clf.predict(X_test)

print(confusion_matrix(y_test, pred_xgb))
print(classification_report(y_test, pred_xgb))
```

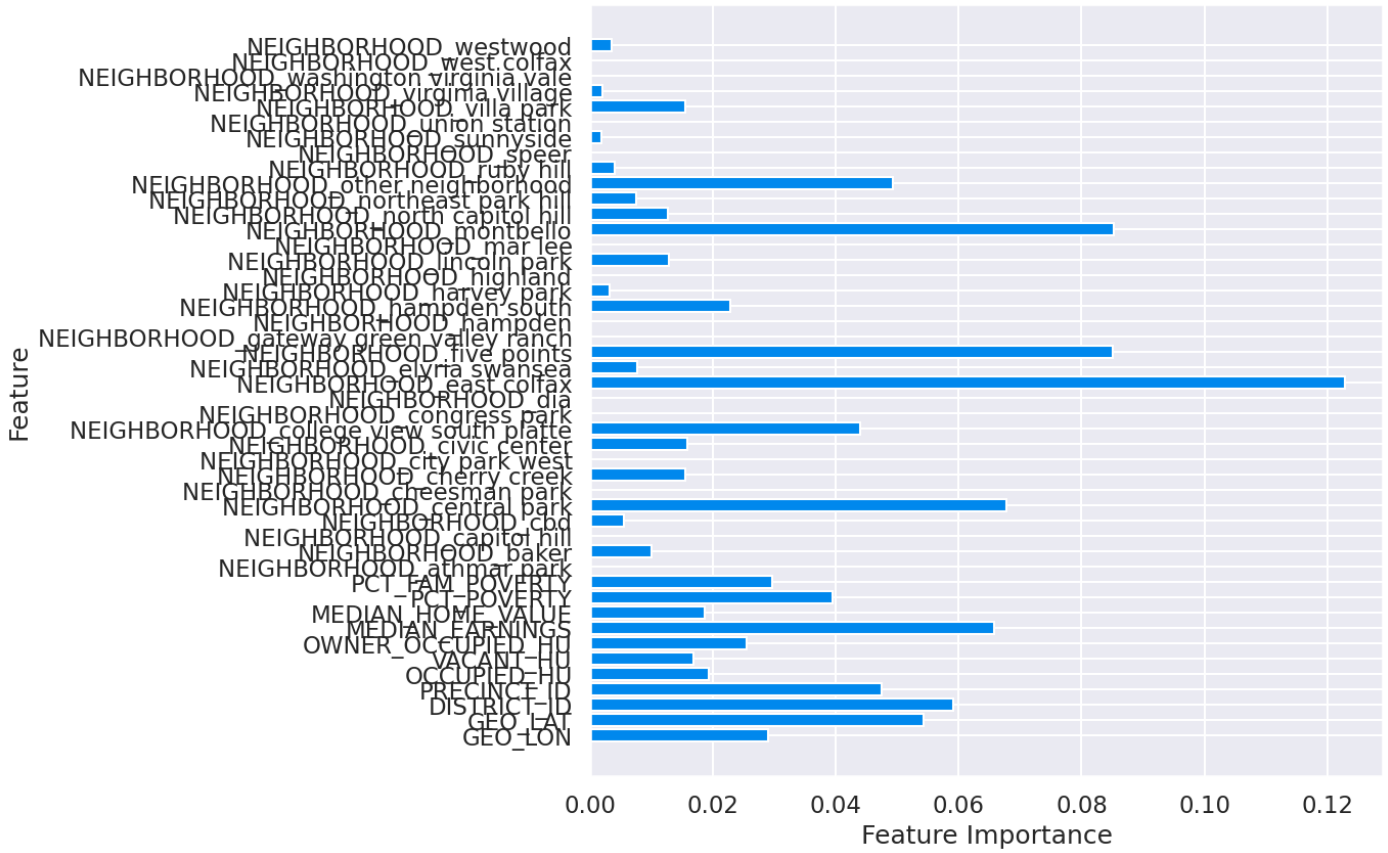
```
[[   0  132    2  395  641    1   70 1517  772]
 [   0  915   17  254 1314    0    9 2944 3571]
 [   0  269   31  185 1316    1    0 1995 2803]
 [   0  129    3 2296 1164    0   89 2084  885]
 [   0  270   22  663 5407    0   44 3124 3641]
 [   0  273    9  336  923  931    9 1277 1019]
 [   0  230    3  597 1298    4  253 2223 1490]
 [   0  485   26  960 2606    4  158 5070 3826]
 [   0  640   48  238 2183    2   16 3485 5513]]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
                              precision    recall  f1-score   support

          aggravated-assault       0.00      0.00      0.00      3530
                  auto-theft       0.27      0.10      0.15      9024
                    burglary       0.19      0.00      0.01      6600
                drug-alcohol       0.39      0.35      0.37      6650
                     larceny       0.32      0.41      0.36     13171
                other-crime       0.99      0.19      0.33      4777
  other-crimes-against-persons       0.39      0.04      0.08      6098
              public-disorder       0.21      0.39      0.28     13135
      theft-from-motor-vehicle       0.23      0.45      0.31     12125

                    accuracy                           0.27     75110
                   macro avg       0.33      0.22      0.21     75110
                weighted avg       0.31      0.27      0.24     75110

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score ar
  _warn_prf(average, modifier, msg_start, len(result))
```

```
fig, ax = plt.subplots(figsize=(15, 15))
plot_confusion_matrix(xgb_clf, X_test, y_test, normalize='true', cmap=plt.cm.Blues,
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```




```
plot_feature_importances(xgb_clf)
```



Gridsearch Model

```
import time
start time = time.time()
```

```
dt_clf = DecisionTreeClassifier(random_state = 23)
dt_cv_score = cross_val_score(dt_clf, X_train, y_train, cv=3)
mean_dt_cv_score = np.mean(dt_cv_score)
print(f"Mean Cross Validation Score: {mean_dt_cv_score :.2%}")
```

```
Mean Cross Validation Score: 29.09%
```

```
dt_param_grid = {'criterion': ['gini', 'entropy'],
                 'max_depth': [None, 2, 3, 4, 5, 6],
                 'min_samples_split': [2, 5, 10],
                 'min_samples_leaf': [1, 2, 3, 4, 5, 6]}


# Instantiate GridSearchCV
dt_grid_search = GridSearchCV(dt_clf, dt_param_grid, cv=3, return_train_score=True)
# Fit to the data
dt_grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=3, error_score=nan,
                 estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                  class_weight='balanced',
                                                  criterion='gini', max_depth=None,
                                                  max_features=None,
                                                  max_leaf_nodes=None,
                                                  min_impurity_decrease=0.0,
                                                  min_impurity_split=None,
                                                  min_samples_leaf=1,
                                                  min_samples_split=2,
                                                  min_weight_fraction_leaf=0.0,
                                                  presort='deprecated',
                                                  random_state=23,
                                                  splitter='best'),
                 iid='deprecated', n_jobs=None,
                 param_grid={'criterion': ['gini', 'entropy'],
                             'max_depth': [None, 2, 3, 4, 5, 6],
                             'min_samples_leaf': [1, 2, 3, 4, 5, 6],
                             'min_samples_split': [2, 5, 10]},
                 pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
                 scoring=None, verbose=0)
```

```
dt_gs_training_score = np.mean(dt_grid_search.cv_results_['mean_train_score'])

# Mean test score
dt_gs_testing_score = dt_grid_search.score(X_test, y_test)

print(f"Mean Training Score: {dt_gs_training_score :.2%}")
print(f"Mean Test Score: {dt_gs_testing_score :.2%}")
print("Best Parameter Combination Found During Grid Search:")
dt_grid_search.best_params_
```

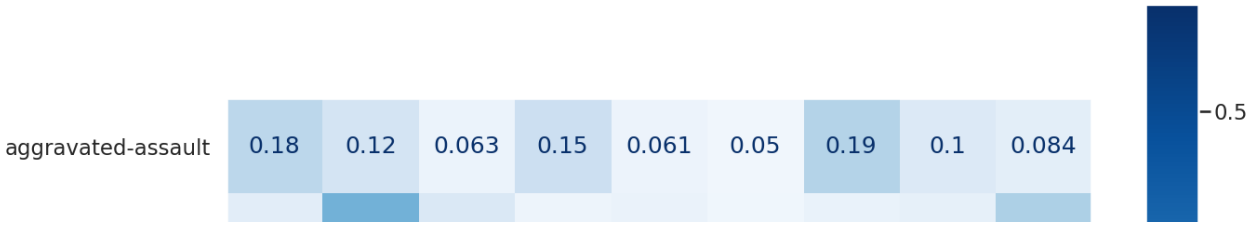
```
Mean Training Score: 22.50%
Mean Test Score: 29.78%
Best Parameter Combination Found During Grid Search:
{'criterion': 'entropy',
 'max_depth': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

```
rf_clf = RandomForestClassifier(random_state = 23)
mean_rf_cv_score = np.mean(cross_val_score(rf_clf, X_train, y_train, cv=3))

print(f"Mean Cross Validation Score for Random Forest Classifier: {mean_rf_cv_score :.2%}")
```

```
Mean Cross Validation Score for Random Forest Classifier: 30.06%
```

```
rf_param_grid = {'n_estimators': [10, 30, 100],
                 'criterion': ['gini', 'entropy'],
                 'max_depth': [None, 2, 6, 10],
                 'min_samples_split': [5, 10],
                 'min_samples_leaf': [3, 6]
}


rf_grid_search = GridSearchCV(rf_clf, rf_param_grid, cv=3)
rf_grid_search.fit(X_train, y_train)

print(f"Training Accuracy: {rf_grid_search.best_score_ :.2%}")
print("")
print(f"Optimal Parameters: {rf grid search.best params }")
```

```
                              _    _           _      _

     Training Accuracy: 30.03%

     Optimal Parameters: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 1


fig, ax = plt.subplots(figsize=(15, 15))
plot_confusion_matrix(rf_grid_search, X_test, y_test, normalize='true', cmap=plt.cm.Blues,
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```




```
xgb_gcls = XGBClassifier(random_state = 42)
mean_xgb_cv_score = np.mean(cross_val_score(rf_clf, X_train, y_train, cv=3))

print(f"Mean Cross Validation Score for XGBoost Classifier: {mean_rf_cv_score :.2%}")
```

```
    Mean Cross Validation Score for XGBoost Classifier: 30.06%
```

```
param_grid = {
    'learning_rate': [0.1, 0.2],
    'max_depth': [6],
    'min_child_weight': [1, 2],
    'subsample': [0.5, 0.7],
    'n_estimators': [100],
}
```

```
xgb_grid_search = GridSearchCV(xgb_gcls, param_grid, scoring='accuracy', cv=None, n_jobs=1)
xgb_grid_search.fit(X_train, y_train)

print(f"Training Accuracy: {xgb_grid_search.best_score_ :.2%}")
print("")
print(f"Optimal Parameters: {xgb_grid_search.best_params_}")
```

```
fig, ax = plt.subplots(figsize=(15, 15))
plot_confusion_matrix(gxb_grid_search, X_test, y_test, normalize='true', cmap=plt.cm.Blues,
                      xticks_rotation=90,ax=ax)
plt.grid(False)

plt.show()
```

XGBoost Gridsearch was discontinued since it was very computationally expensive, limited by the capabilities of our system.

Results

The results for our models were as follows:

- Random Forest Model: 24%
- Decision Trees Model: 32%
- XGBoost Model: 27%

Although these are not satisfactory results, there is understanding as to the challenges that were faced during modeling. The complexity of the models with all the features created challenges correlated with predictive models for crime. Research has failed to produce efficient models when it comes to large datasets such as the one that we worked with. In some models, high levels of accuracy were able to be achieved, however these results were limited to small datasets with limited features. The important features that need to be considered when using

Machine Learning in order to predict crime is the computational speed, robustness and the scalability of our data. It is important to note as well

Recommendations

Although our models did not produce as robust as results as we would have liked, 32% (Decision Trees), 27% (XGBoost), and 32% (Random Forest) there are some recommendations we can make.

- Median earnings was an important feature, and this can in both directions. More crime prevention and community services should be offered in low median earning locations and more patrolling should be done in high median earning areas as much of the crime in these areas involve larceny.
- Geolocation plays a big role, and crime tends to occur in an area where criminals are familiar with the location. This should lead to greater patrolling in areas where there is geographically more crime
- As highlighted by our exploratory analysis and our XGBoost model, neighborhood plays an important role in identifying and predicting where crime might happen. Extra patrolling and community outreach services should be implemented in the areas of Five Points, Montbello, and East Colfax. This can also include community education in term of how to keep oneself, our neighbors and communities safe. Also using apps like Nextdoor can help community members stay informed of what is going on in their neighborhood.

Future Work

In an extension to the work that has been completed, it is planned to add more classification models to hopefully increase the crime prediction accuracy and hopefully enhance overall performance. It is also the hope to include previously excluded features from the demographic portion of our dataset. We have the option to add gender, education level, earnings levels and race into our model. It will be interesting to see if any of these demographics make a sizable impact on the performance of our model.

Also, it will be helpful in the future to continue including more data from the Denver crime dataset to see if there are more noticeable trends in the future. This framework can also hopefully be extended to other major metropolitan cities such as Chicago and Los Angeles to see if there may be other pivotal features that can increase model performance. The goal is to hopefully increase the audience to the work of using Machine Learning programs to help inform local law enforcement and community leaders to keep neighborhoods safe.