None

Analysis of Crime Data in Washington, D.C.

Bo He

This is the final project of CMSC320 taken in Spring 2023 at the University of Maryland. The project intends to create a tutorial that will guide users through the entire data science pipeline. More detailed requirements can be found at here (https://www.cs.umd.edu/class/spring2023/cmsc320-0101/files/tutorial.pdf).

Table of Contents

Introduction

As the capital of the United States, Washington, D.C., is known for its rich history and diverse population. However, like other areas, it faces challenges that include crime. Understanding and analyzing crime data can help identify times and places where crime rates are high, thus keeping people out of harm's way. Likewise, these data can help policymakers develop effective strategies and allocate resources to ensure public safety.

The purpose of this report is to provide a comprehensive analysis of crime data in Washington, D.C., and to shed light on the trends and patterns that influence criminal activity in the city. I hope that after reading you will gain a better understanding of the issue and implement necessary measures to enhance safety.

Data Collection

In this section, we are going to collect data from the website https://opendata.dc.gov/ (https://opendata.dc.gov/). This is an official website that serves as the open data portal for Washington, D.C. It provides public access to a wide range of datasets and information related to various aspects of the city. It is clear that the data we downloaded is legitimate.

The website provides several types of data for download. To facilitate our analysis, we are using .csv files, which are commonly used for organizing and analyzing large sets of data. We downloaded all available data in the database about crime incidents, which spans from 2008 to 2022. This approach allows us to examine the trends and patterns of crime over the time period.

In [ ]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
```

In [ ]:

```
# Save data to the DataFrame

# List to store the dataframes
dfs = []
base_path = "/content/drive/MyDrive/Colab Notebooks/CMSC320/Final Tutorial/Crime_Incidents_in_"

# Read the CSV files
for year in range(2008, 2023):
    file_path = base_path + str(year) + ".csv"
    crime_data = pd.read_csv(file_path)
    crime_data['YEAR'] = year # Add a column for the year
    dfs.append(crime_data)

# Given an example of what the data looks like
dfs[0]
```

Out[ ]:

	X	Y	CCN	REPORT_DAT	SHIFT	METHOD	OFFENSE	B
0	-76.950343	38.893545	8054727	2008/04/25 08:16:00+00	MIDNIGHT	KNIFE	SEX ABUSE	3900 BLO MINNE AVEN
1	-76.950343	38.893545	8054727	2008/04/25 08:16:00+00	MIDNIGHT	KNIFE	SEX ABUSE	3900 BLO MINNE AVEN
2	-77.016199	38.899067	8137907	2008/09/28 03:50:00+00	MIDNIGHT	OTHERS	SEX ABUSE	700 BLO STREE
3	-77.016199	38.899067	8137907	2008/09/28 03:50:00+00	MIDNIGHT	OTHERS	SEX ABUSE	700 BLO STREE
4	-76.986918	38.900203	8181733	2008/12/25 17:00:00+00	DAY	KNIFE	SEX ABUSE	1300 BLO H ST
...	...	...	...	...	...	...	...	
34321	-77.030804	38.897353	9176740	2008/12/15 16:00:00+00	DAY	OTHERS	THEFT/OTHER	1300 BLO F ST
34322	-76.994365	38.903185	9176766	2008/12/15 14:00:00+00	DAY	OTHERS	THEFT/OTHER	800 BLO VIR AVEN
34323	-76.986783	38.879817	9176824	2008/12/15 16:45:00+00	DAY	OTHERS	THEFT/OTHER	1350 BLO POT AVEN
34324	-77.024572	38.883891	9176834	2008/12/15 17:00:00+00	DAY	OTHERS	THEFT/OTHER	400 BLO L'EN PLAZ
34325	-77.001662	38.877529	9176923	2008/12/15 20:30:00+00	EVENING	OTHERS	THEFT/OTHER	300 BLO L ST

34326 rows × 26 columns

Data Processing

In this section, we are going to do a data processing, which include data cleaning, data transformation and so on.

We first combine our sperate dataframes(each dataframe contains one year's crime data) to one dataframes. Then, we drop any unuseful data and extract needed data(make new column parameters) to the dataframe. After that, we need to delete duplicate values(exactly same crime report), reports that not in the year of 2008 to 2022, reports that not have occurred in the report year to ensure accurate of our data.Then, sort the report based on the start data time.

In [ ]:

```
# Create a new DataFrame that combine all dataframes and removes unneede
d data
df = pd.concat(dfs, axis=0, ignore_index=True)
df = df.drop(['X', 'Y', 'CCN', 'BLOCK', 'XBLOCK', 'YBLOCK', 'BLOCK_GROU
P', 'BID', 'METHOD', 'YEAR',
              'OBJECTID', 'WARD', 'ANC', 'DISTRICT', 'PSA', 'OCTO_RECORD
_ID', 'NEIGHBORHOOD_CLUSTER'], axis = 1)

# Modify the dataframe, to extract needed data and delete unneeded data
further
df['REPORT_DAT'] = pd.to_datetime(df['REPORT_DAT'])
df['START_DATE'] = pd.to_datetime(df['START_DATE'])
df['REPORT_YEAR'] = df['REPORT_DAT'].dt.year
df['START_TIME'] = df['START_DATE'].dt.hour
df['WEEKDAY'] = df['START_DATE'].dt.day_name()

df['LOCATION'] = np.array(df[['LATITUDE','LONGITUDE']]).tolist()
df = df.drop(['LATITUDE', 'LONGITUDE', 'CENSUS_TRACT'], axis = 1)

# Delete duplicate values
df.drop_duplicates(subset=df.columns.difference(['LOCATION']))

# Delete reports that not in the year of 2008 to 2022
years_to_keep = range(2008, 2023)
df = df[df['REPORT_YEAR'].isin(years_to_keep)]

# Delete reports that not have occurred in the report year
df_cleaned = df[df['REPORT_YEAR'] == df['START_DATE'].dt.year]

# Sort the data based on the start data time
df = df_cleaned.sort_values(by='START_DATE')
df = df.reindex(columns=['REPORT_YEAR',  'REPORT_DAT', 'START_DATE', 'ST
ART_TIME', 'END_DATE', 'WEEKDAY', 'SHIFT', 'OFFENSE', 'LOCATION', 'VOTIN
G_PRECINCT'])
df
```

Out[ ]:

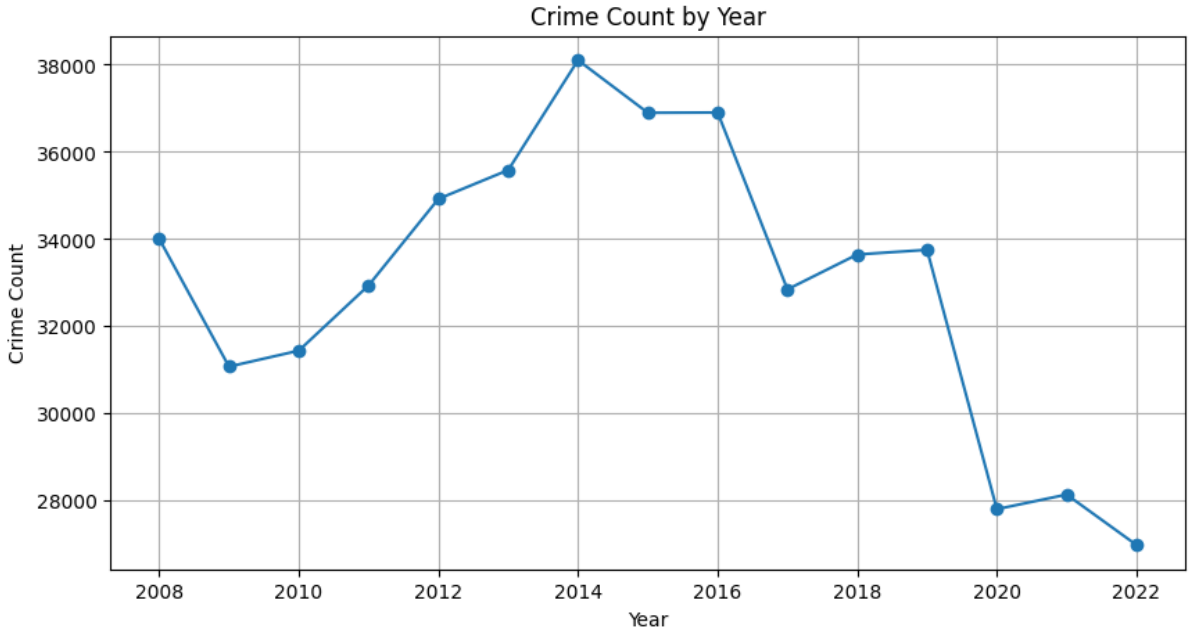
	REPORT_YEAR	REPORT_DAT	START_DATE	START_TIME	END_DATE	WEEKDAY	
401	2008	2008-01-02 17:30:00+00:00	2008-01-01 00:00:00+00:00	0.0	2008/01/02 14:00:00+00	Tuesday	
115	2008	2008-01-01 17:00:00+00:00	2008-01-01 00:30:00+00:00	0.0	2007/12/31 05:00:00+00	Tuesday	
1534	2008	2008-01-01 07:10:00+00:00	2008-01-01 01:30:00+00:00	1.0	2007/12/31 05:00:00+00	Tuesday	MID
121	2008	2008-01-01 18:13:00+00:00	2008-01-01 02:00:00+00:00	2.0	2008/01/01 06:30:00+00	Tuesday	
4258	2008	2008-02-22 19:31:00+00:00	2008-01-01 02:00:00+00:00	2.0	2008/02/22 17:00:00+00	Tuesday	
...	...	...	...	...	...	...	
494616	2022	2022-12-31 20:49:29+00:00	2022-12-31 20:21:00+00:00	20.0	2022/12/31 20:26:00+00	Saturday	EV
471824	2022	2022-12-31 22:04:04+00:00	2022-12-31 20:59:00+00:00	20.0	2022/12/31 21:40:00+00	Saturday	EV
482655	2022	2022-12-31 22:29:00+00:00	2022-12-31 21:07:00+00:00	21.0	2022/12/31 22:29:00+00	Saturday	EV
483920	2022	2022-12-31 23:10:21+00:00	2022-12-31 22:37:00+00:00	22.0	2022/12/31 23:00:00+00	Saturday	EV
474342	2022	2022-12-31 23:21:11+00:00	2022-12-31 22:45:00+00:00	22.0	2022/12/31 23:10:00+00	Saturday	EV

495055 rows × 10 columns

Data Visualization

In this section, we are doing data visualization to present data in a visually appealing and understandable way. We are going to graph the data to gain a better understanding of the data. Also, we attempt to perform statistical analyses in this section to gain mathematical evidence for the trends we may discover.

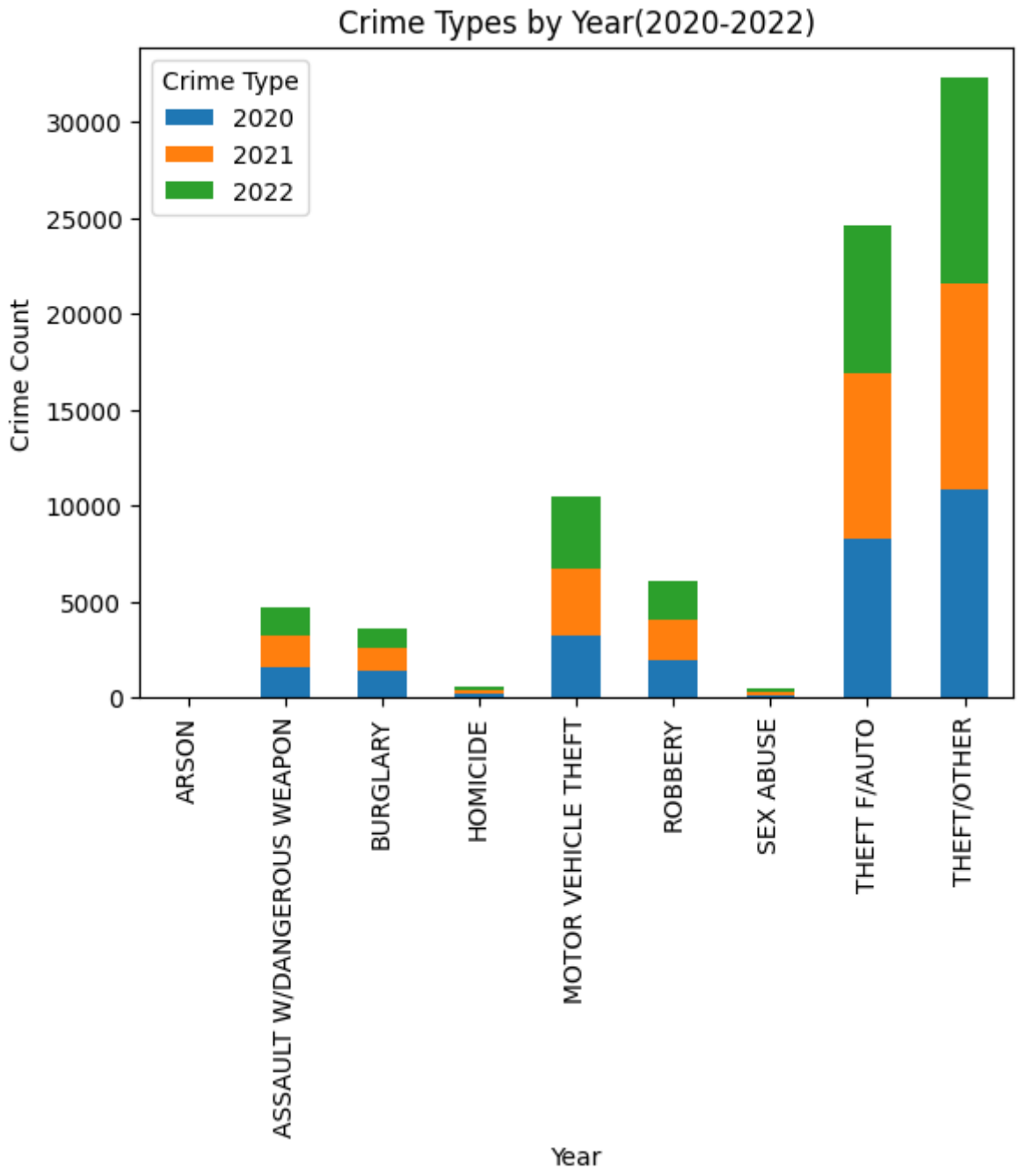
In [ ]:
```
# Line Graph - Crime Count by Year
crime_count = df.groupby('REPORT_YEAR').size()

plt.figure(figsize=(10, 5))
plt.plot(crime_count.index, crime_count.values, marker='o')
plt.title('Crime Count by Year')
plt.xlabel('Year')
plt.ylabel('Crime Count')
plt.grid(True)
plt.show()
```



The plot about crime count by year shows that the number of crimes have an upward and then downward trend, and peaked in 2014 at about 38,000 cases. We can saw an annual increase about 2000 cases between 2009 and 2014, with a significant decrease from 2016 to 2017 and 2019 to 2020. It is good to aware that 2019 - 2022 are in the Covid period, so the crime count are in a low period.

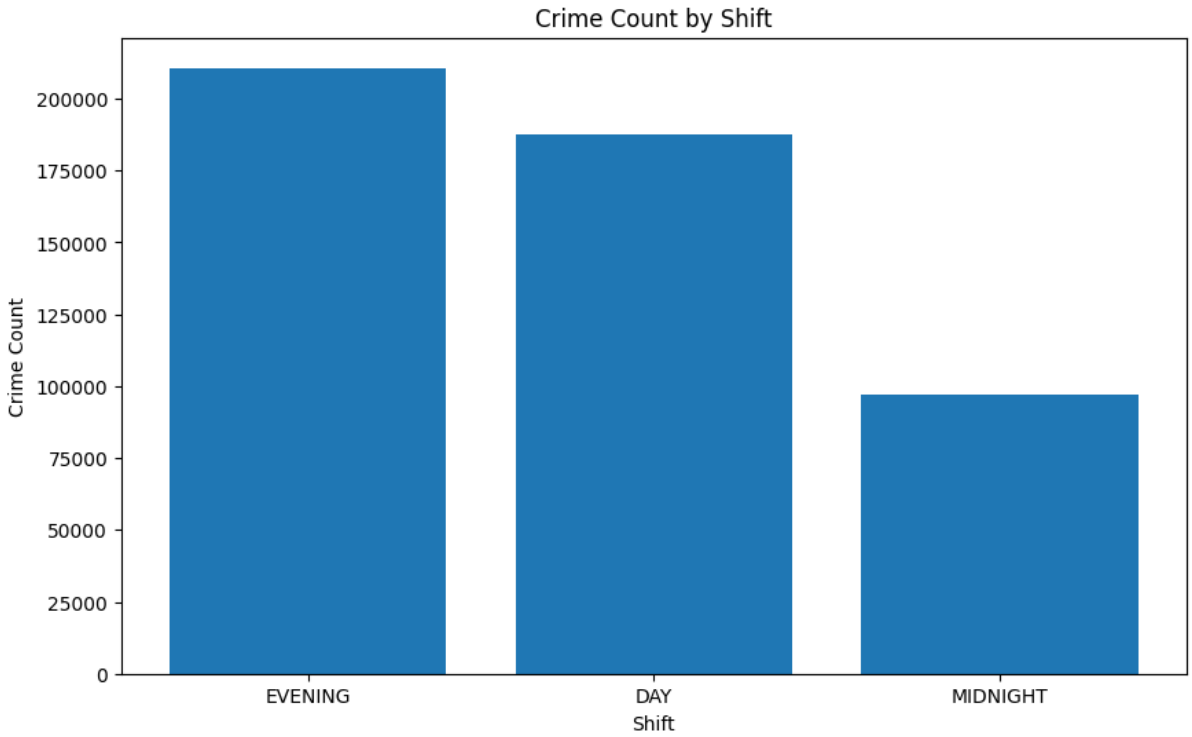
In [ ]:
```
# Bar chart of crime types by year
# We are considerig the Covid time for this period
crime_count = df.groupby(['OFFENSE', 'REPORT_YEAR']).size().unstack()
crime_count = crime_count.loc[:, 2020:2022]

# Create a bar chart
plt.figure(figsize=(10, 6))
crime_count.plot(kind='bar', stacked=True)
plt.title('Crime Types by Year(2020-2022)')
plt.xlabel('Year')
plt.ylabel('Crime Count')
plt.legend(title='Crime Type')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



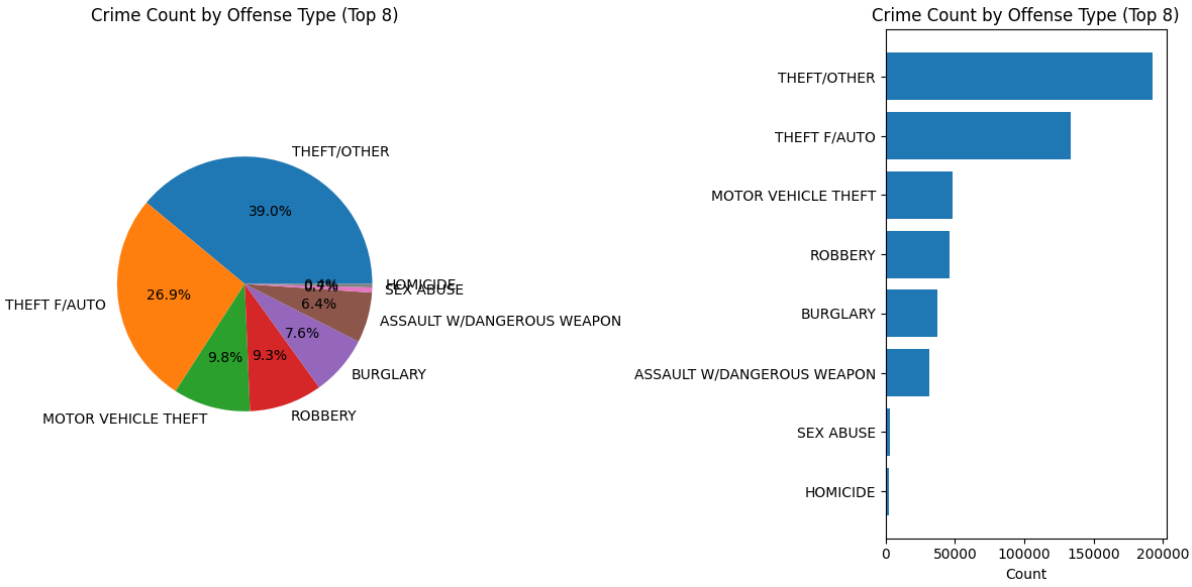
We can see that Theft are the top crime type in the Covid time, it seems like crime numbers are about equal for each type of crime for years.

In [ ]:
```
# Crime Count by Shift
shift_counts = df['SHIFT'].value_counts()

# Bar plot for crime count by shift
plt.figure(figsize=(10, 6))
plt.bar(shift_counts.index, shift_counts.values)
plt.xlabel('Shift')
plt.ylabel('Crime Count')
plt.title('Crime Count by Shift')
plt.show()
```



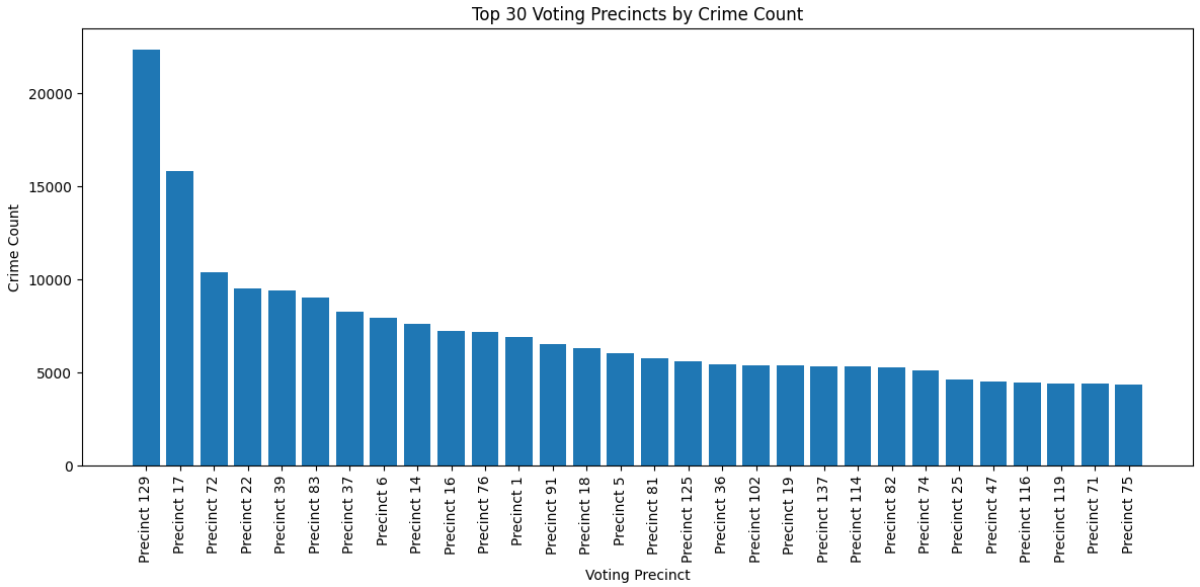
The bar plot for crime count by shift shows that crime generally occurs on evenings and days, with crime occurring in midnight being about half of crime occurring in evening and day.

In [ ]:
```
# Crime Count by Offense Type
crime_count_offense = df['OFFENSE'].value_counts().head(8)

plt.figure(figsize=(12, 6))

# Pie Chart
# Part of the code for plotting this image are from stack overflow
plt.subplot(121)
plt.pie(crime_count_offense, labels=crime_count_offense.index, autopct
='%1.1f%%')
plt.title('Crime Count by Offense Type (Top 8)')
plt.axis('equal')

# Bar Plot
# Part of the code for plotting this image are from stack overflow
plt.subplot(122)
plt.barh(crime_count_offense.index[::-1], crime_count_offense.values[::-
1])
plt.title('Crime Count by Offense Type (Top 8)')
plt.xlabel('Count')

plt.tight_layout()
plt.show()
```



From these two plots we can know that the major offense type are Theft/Other with 39% and Theft F/Auto with 26.9%. Motor vehicle theft are dropped dramatically to 9.8%. Therefore, in order to be safe, we should not reveal our wealth and protect our belongings.

In [ ]:
```
# Top 30 Voting Precincts by Crime Count
# Group the data by voting precinct and calculate the crime count for ea
ch precinct
precinct_counts = df['VOTING_PRECINCT'].value_counts().sort_values(ascen
ding=False)

# Select the top 10 precincts with the highest crime count
top_precincts = precinct_counts.head(30)

# Bar plot for crime count by voting precinct
plt.figure(figsize=(12, 6))
plt.bar(top_precincts.index, top_precincts.values)
plt.xlabel('Voting Precinct')
plt.ylabel('Crime Count')
plt.title('Top 30 Voting Precincts by Crime Count')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



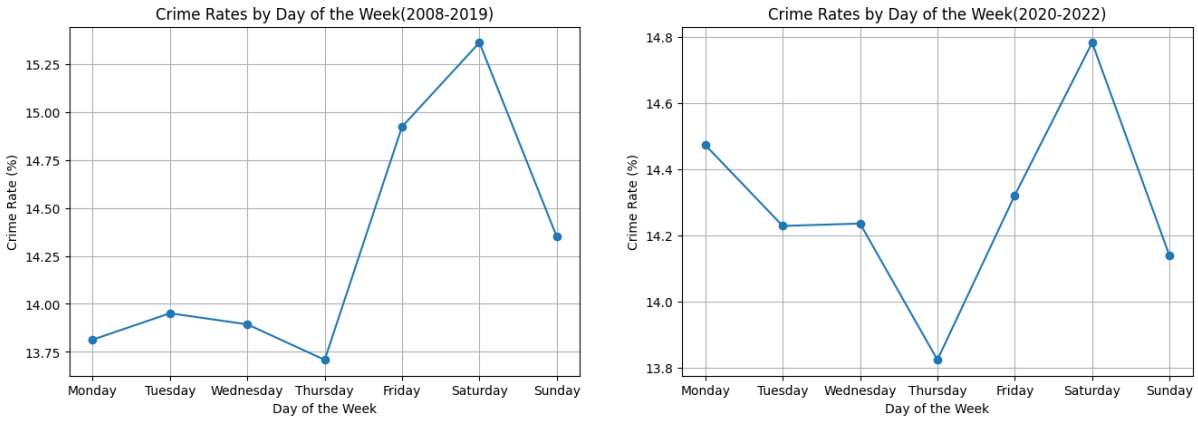
Precinct 129 and Precinct 17 have higher crime count than other voting precincts. Policymakers and community leaders should develop effective strategies and allocate resources to ensure public safety, and also their voting.

Exploratory Analysis

This section aims to examining and summarizing the data using statistical methods, data manipulation techniques, and visualizations.

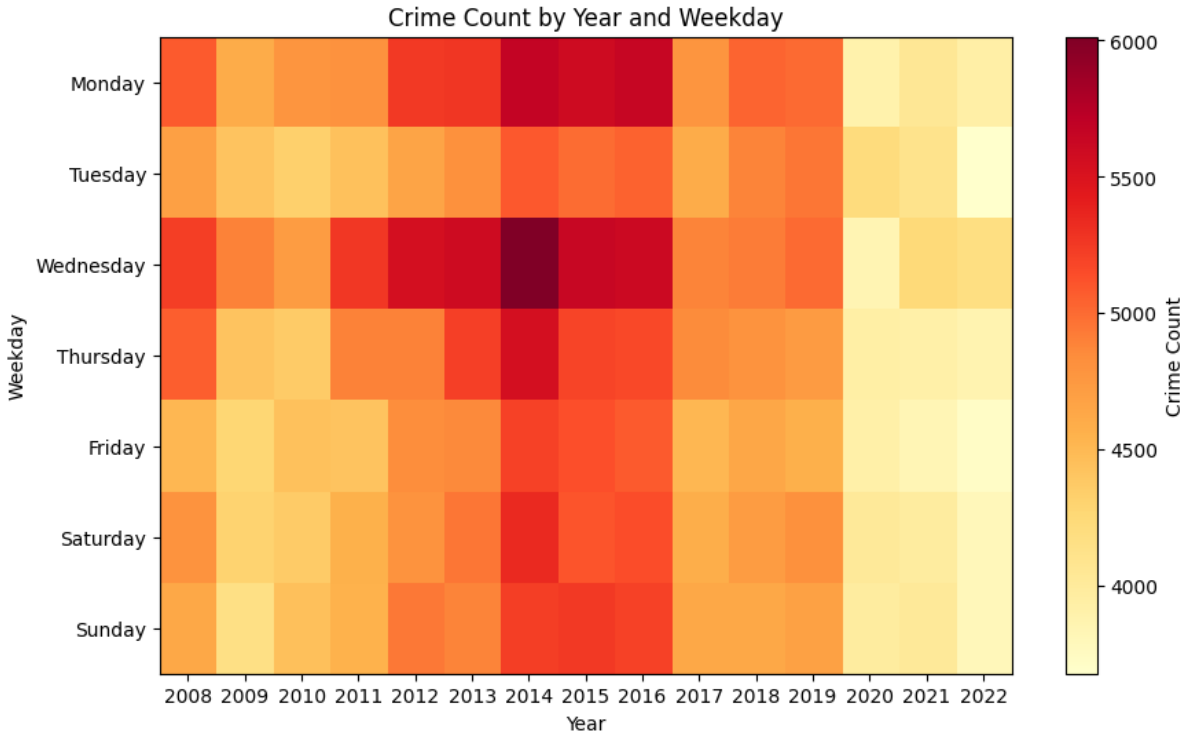
In [ ]:

```
# Crime Rates by Day of the Week
# Considering the influence of COVID, we seperate the data to 2008-2019
and 2020-2022
df_normal = df[df['REPORT_YEAR'] <= 2019]
df_covid = df[(df['REPORT_YEAR'] >= 2020) & (df['REPORT_YEAR'] <= 2022)]


# Calculate the crime rates by day of the week
weekday_counts_n = df_normal['WEEKDAY'].value_counts()
crime_rates_n = (weekday_counts_n / weekday_counts_n.sum()) * 100

weekday_counts_c = df_covid['WEEKDAY'].value_counts()
crime_rates_c = (weekday_counts_c / weekday_counts_c.sum()) * 100

# Reindex the crime rates by day of the week based on the desired order
crime_rates_n = crime_rates_n.reindex(['Monday', 'Tuesday', 'Wednesday',
                                       'Thursday', 'Friday',
'Saturday', 'Sunday'])
crime_rates_c = crime_rates_c.reindex(['Monday', 'Tuesday', 'Wednesday',
                                       'Thursday', 'Friday',
'Saturday', 'Sunday'])

# Create a line graph
plt.figure(figsize=(16, 5))
plt.subplot(121)
plt.plot(crime_rates_n.index, crime_rates_n.values, marker='o')
plt.title('Crime Rates by Day of the Week(2008-2019)')
plt.xlabel('Day of the Week')
plt.ylabel('Crime Rate (%)')
plt.grid(True)

plt.subplot(122)
plt.plot(crime_rates_c.index, crime_rates_c.values, marker='o')
plt.title('Crime Rates by Day of the Week(2020-2022)')
plt.xlabel('Day of the Week')
plt.ylabel('Crime Rate (%)')
plt.grid(True)

plt.show()
```



We can see that either for normal time or Covid time, Saturday always have the highest crime rate. Normal time with a percentage around 15.5%, Covid time with a percentage around 14.8%. On the other hand, Thursay always be the lowest crime rate, with about 13.7% on normal time and around 13.9% on Covid time.

In [ ]:
```
# Heapmap about crime count by year and weekday
# Calculate the crime count for each year and weekday
crime_count = df.groupby(['REPORT_YEAR', 'WEEKDAY']).size().reset_index
(name='COUNT')

# Pivot the data to create a pivot table
pivot_table = crime_count.pivot(index='WEEKDAY', columns='REPORT_YEAR',
values='COUNT')

# Set the order of weekdays for proper sorting on the y-axis
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']

# Plot the heatmap
# Part of the code for plotting this image are from stack overflow
plt.figure(figsize=(10, 6))
plt.imshow(pivot_table.values, cmap='YlOrRd', aspect='auto')
plt.xticks(range(len(pivot_table.columns)), pivot_table.columns)
plt.yticks(range(len(pivot_table.index)), weekday_order)
plt.colorbar(label='Crime Count')
plt.title('Crime Count by Year and Weekday')
plt.xlabel('Year')
plt.ylabel('Weekday')
plt.show()
```

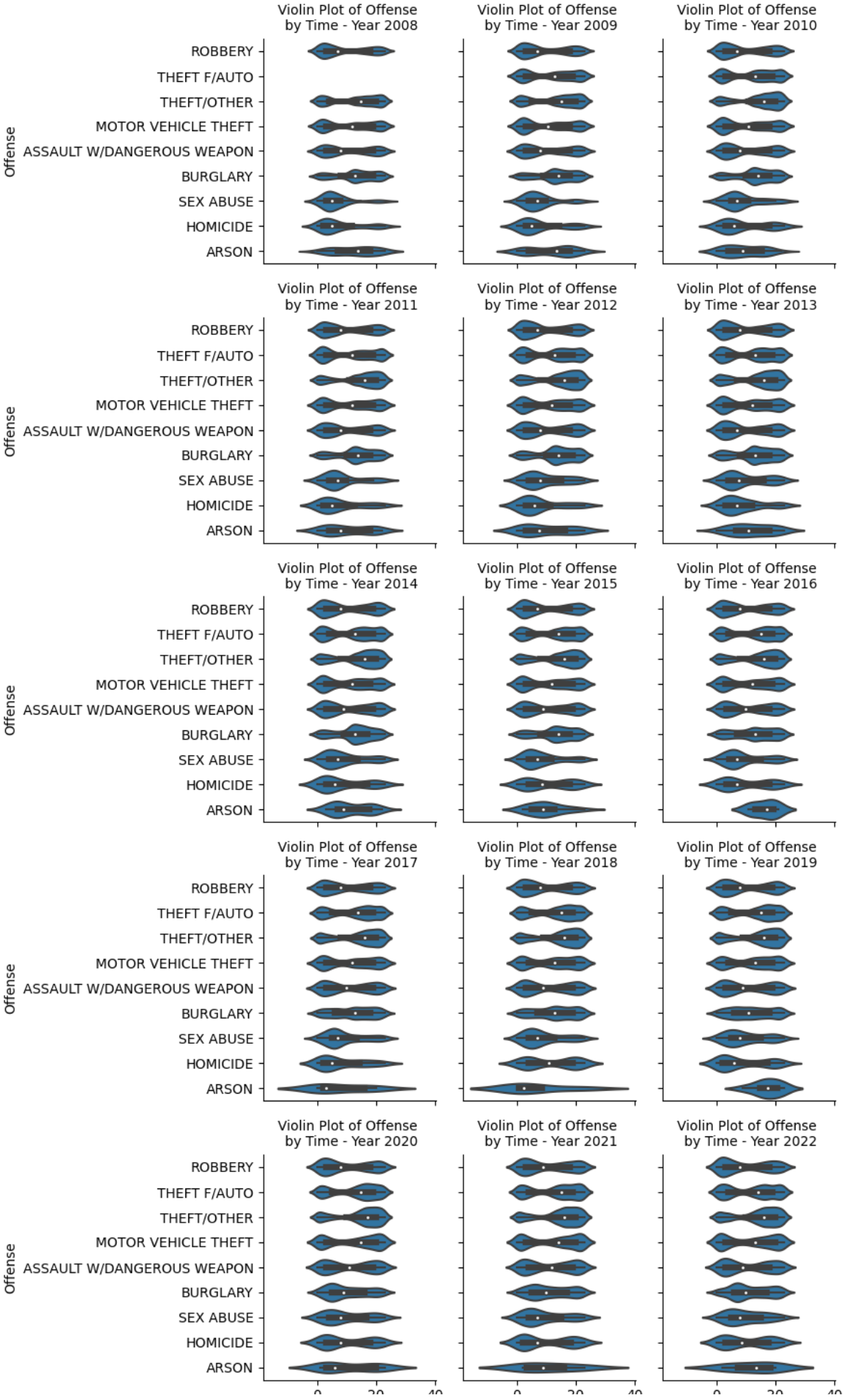


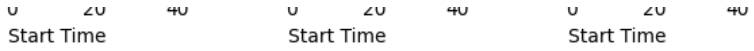
From year 2014 to 2016, the crime count are in the high period of the time range we collected. Highest number of crimes occurred on Wednesdays in 2014, with second highest number of crimes occurred on Monday in 2014. Agian, it is good to aware that 2019 - 2022 are in the Covid period, so the crime count are in a low period.

In [ ]:

```
# Part of the code for plotting this image are from stack overflow

plt.figure(figsize=(12,8))
# Specify the order of offense categories
offense_order = ['ROBBERY', 'THEFT F/AUTO', 'THEFT/OTHER', 'MOTOR VEHICL
E THEFT',
                 'ASSAULT W/DANGEROUS WEAPON', 'BURGLARY', 'SEX ABUSE',
'HOMICIDE', 'ARSON']

# Create a FacetGrid with separate violin plots for each year
g = sns.FacetGrid(df, col='REPORT_YEAR', col_wrap=3)
g.map(sns.violinplot, 'START_TIME', 'OFFENSE', order=offense_order)

# Set the titles and labels
g.set_titles("Violin Plot of Offense \n by Time - Year {col_name}")
g.set_xlabels('Start Time')
g.set_ylabels('Offense')

# Adjust the spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```


<Figure size 1200x800 with 0 Axes>


Violin Plot of Offense by Time - Year 2008
Violin Plot of Offense by Time - Year 2009
Violin Plot of Offense by Time - Year 2010
Violin Plot of Offense by Time - Year 2011
Violin Plot of Offense by Time - Year 2012
Violin Plot of Offense by Time - Year 2013
Violin Plot of Offense by Time - Year 2014
Violin Plot of Offense by Time - Year 2015
Violin Plot of Offense by Time - Year 2016
Violin Plot of Offense by Time - Year 2017
Violin Plot of Offense by Time - Year 2018
Violin Plot of Offense by Time - Year 2019
Violin Plot of Offense by Time - Year 2020
Violin Plot of Offense by Time - Year 2021
Violin Plot of Offense by Time - Year 2022
Offense
ROBBERY
THEFT F/AUTO
THEFT/OTHER
MOTOR VEHICLE THEFT
ASSAULT W/DANGEROUS WEAPON
BURGLARY
SEX ABUSE
HOMICIDE
ARSON
0
20
40



The violin plot about time and offence for each year shows the distribution of crime start time and type of offense. We can extract some distinguishing features from it. For example, robberies always happen in the morning, with theft often in the afternoon and evening. Arson are focuing on the afternoon for 2019 and 2022. etc. We can see many interesting data from this plot.

Machine Learning Implementation

We will use machine learning in two different ways in this project. First, we will utilize linear regression to predict the crime count each month in the future. Then we will use KMEANS to find clusters of crime and map them.

Linear Regression

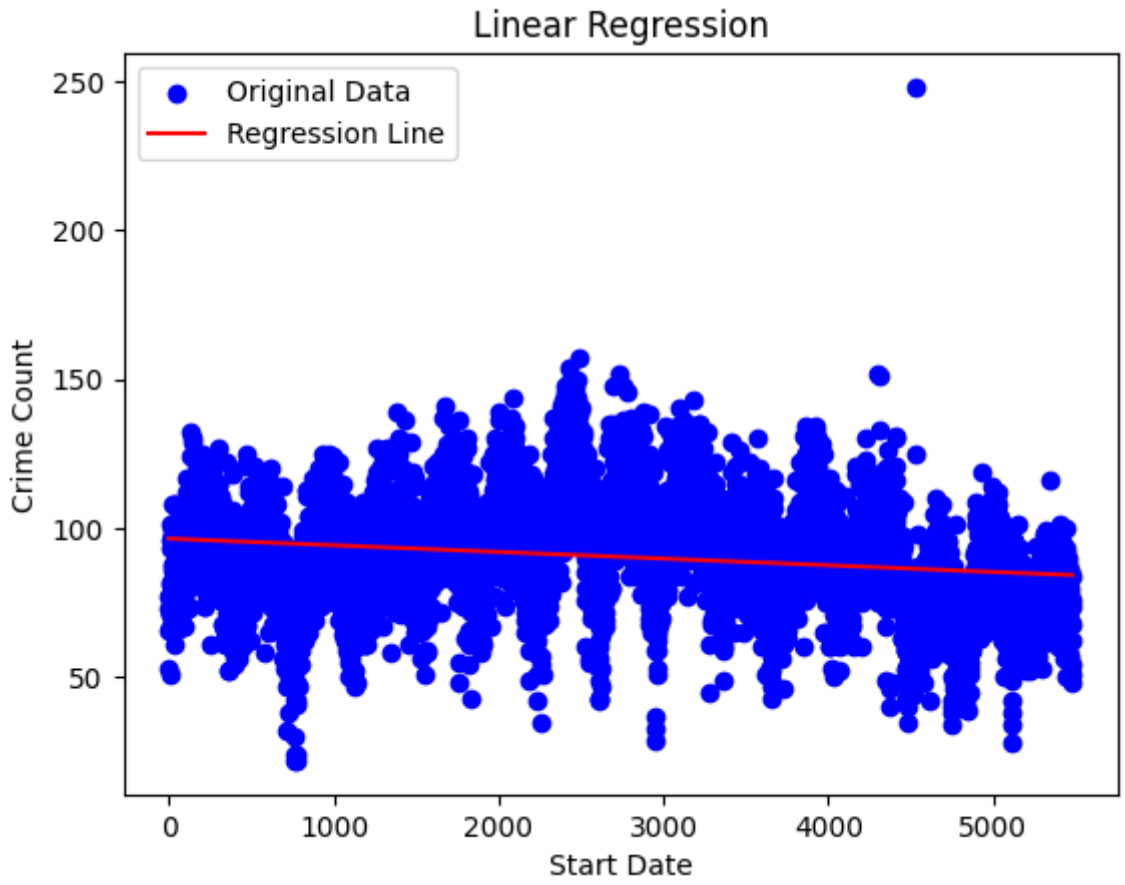
In [ ]:
```
# Predict the crime count everyday in the future using regression
# Some of the code are from stack overflow

df_ml = df.copy()
# Convert 'START_DATE' to the Day period
df_ml['START_DATE'] = df_ml['START_DATE'].dt.date

# We normalize the date in order to do the regression
df_ml['START_DATE'] = (df_ml['START_DATE'] - df_ml['START_DATE'].min())
/ np.timedelta64(1,'D')

# Calculate the crime count everyday
df_ml['CRIME_COUNT'] = df_ml.groupby('START_DATE')['REPORT_DAT'].transfo
rm('count')

# Drop the unnecessary columns
df_ml = df_ml.drop(['REPORT_YEAR','REPORT_DAT', 'START_TIME', 'END_DAT
E', 'WEEKDAY', 'SHIFT', 'OFFENSE', 'LOCATION', 'VOTING_PRECINCT'], axis=
1)
df_ml.drop_duplicates(subset=['START_DATE', 'CRIME_COUNT'], keep='firs
t', inplace=True)

df_ml
```

Out[ ]:

	START_DATE	CRIME_COUNT
401	0.0	77
790	1.0	53
505	2.0	66
907	3.0	73
1184	4.0	71
...	...	...
478774	5474.0	68
478246	5475.0	62
474336	5476.0	84
496888	5477.0	74
472533	5478.0	51

5479 rows × 2 columns

In [ ]:
```
# Initialize a linear regression model
model = LinearRegression()

X = df_ml['START_DATE'].values.reshape(-1, 1)
y = df_ml['CRIME_COUNT'].values

# Train the model
model.fit(X, y)

# Generate a range of dates for prediction
dates = np.arange(df_ml['START_DATE'].min(), df_ml['START_DATE'].max()+
1).reshape(-1, 1)

# Make predictions using the trained model
predictions = model.predict(dates)

# Plot the original data and the regression line
plt.scatter(df_ml['START_DATE'], df_ml['CRIME_COUNT'], color='b', label
='Original Data')
plt.plot(dates, predictions, color='r', label='Regression Line')
plt.xlabel('Start Date')
plt.ylabel('Crime Count')
plt.title('Linear Regression')
plt.legend()
plt.show()
```

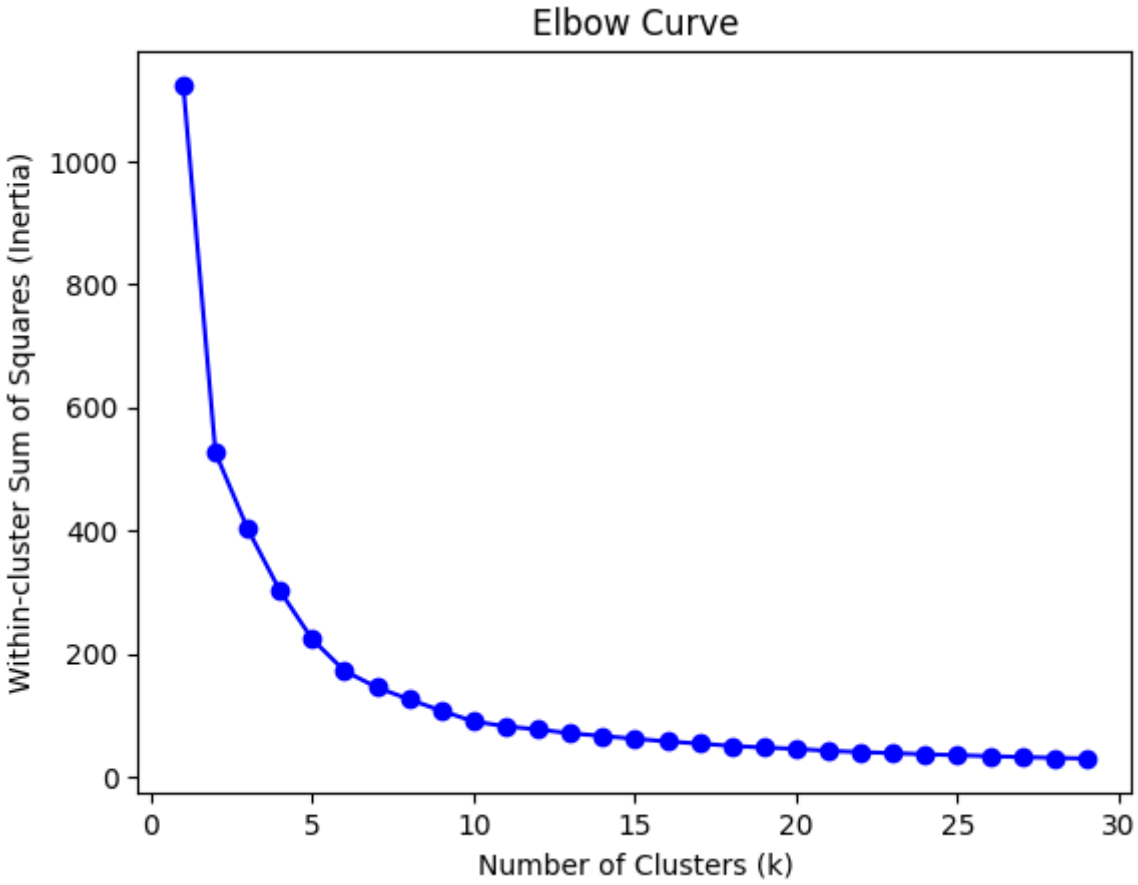


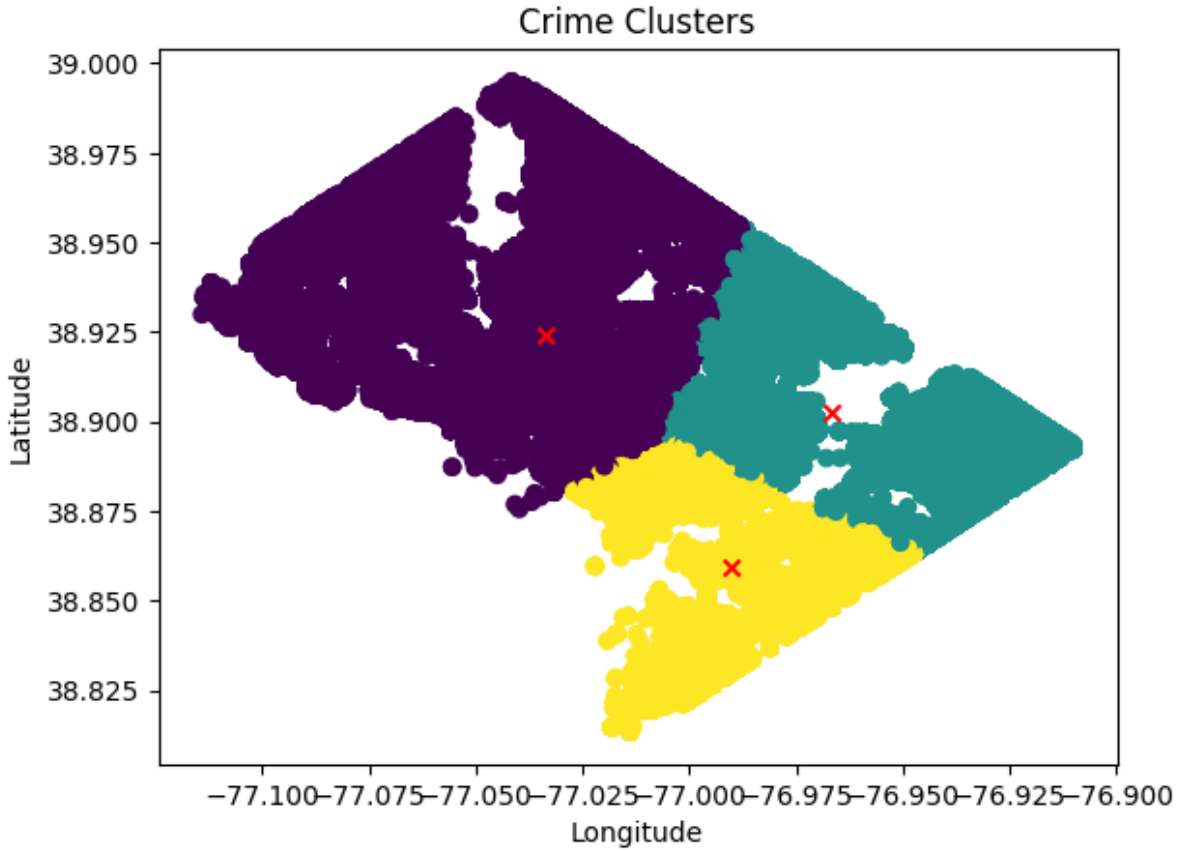
This is the linear regression generated from the data we collected. As we can see, majority of the data are in the range, however there are some noise points that extra large/extra small, which also says this is only a prediction model.

KMEANS

In [ ]:

```
# Extract the coordinate
cords = df['LOCATION']
lat = [cord[0] for cord in cords]
long = [cord[1] for cord in cords]

# Perform K-means clustering on lat-long data
X = pd.DataFrame({'latitude': lat, 'longitude': long}).values

# Determine the optimal number of clusters using the elbow method
# Some code are generated by ChatGPT
inertia = []
k_range = range(1, 30)  # Test different numbers of clusters
for k in k_range:
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve to find the optimal number of clusters
plt.plot(k_range, inertia, 'bo-')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-cluster Sum of Squares (Inertia)')
plt.title('Elbow Curve')
plt.show()

# Choose the optimal number of clusters based on the elbow curve
k_optimal = 3  # Adjust the value based on the elbow curve

# Perform K-means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=k_optimal,  n_init=10, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Plot the crime clusters
plt.scatter(X[:, 1], X[:, 0], c=labels, cmap='viridis')
plt.scatter(centroids[:, 1], centroids[:, 0], c='red', marker='x')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Crime Clusters')
plt.show()
```


Elbow Curve
Within-cluster Sum of Squares (Inertia)
1000
800
600
400
200
0
0
5
10
15
20
25
30
Number of Clusters (k)


Crime Clusters
Latitude
39.000
38.975
38.950
38.925
38.900
38.875
38.850
38.825
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
−76.950
−76.925
−76.900
Longitude

We can see that there are three colors in the scatter plot, which indicates that the K-means algorithm has identified three distinct clusters in the data based on the coordinates we have.

Also, in the website we colloecting data, there is also a 3D zoning map of Washington, D.C., you can check it at here (https://opendata.dc.gov/apps/zoning-map-in-3d/explore). The plot we made is similiar to the map on the website.

Interpretation: Insight & Policy Decision

This is the last part of the data science pipeline. We are going to do a conclusion of the data we visualized in this tutorial. Also, as the title of this part, we need a further analysis of the data and outcomes to find some potentially infers.

Based on our observations throughout our analysis and modeling, we can conclude that:

- Precinct 129 and Precinct 17 are crime hotspots, which means these areas have a higher crime rates. Identified these, we can help policymakers and community leaders to develop effective strategies and allocate resources on such areas to ensure public safety
- Avoid going out alone in evening, as evening is the highest crime period. If you have to do so, protect yourself safely.
- Theft are the most common offense type for crime in Washington, D.C.Implementing preventive measures can be effective in reducing the occurrence of such crimes, include promoting community awareness, improving security systems in residential and so on.
- Crime rate rises to hightest on Saturday in D.C. keep this in mind and try to avoid any potential risks.

If you encounter any crime or in risks, remember: **RUN! HIDE! FIGHT!**

If you are interested in further research, we might do the following task:

- Conduct a comprehensive analysis of crime data specifically focusing on Saturdays. Explore different types of crimes, their distribution across neighborhoods, and their temporal patterns throughout the day.
- Explore the demographic characteristics of both offenders and victims involved in crimes. Analyze age groups, gender, socioeconomic status, and other relevant factors to identify any specific population segments that are more susceptible to being involved in or affected by those crimes.
- ...

Be aware that - in order to do such analysis, we need more detailed data regarding crime reports.

To learn more about a given topic check the following links:

- NeighborhoodScout (https://www.neighborhoodscout.com/dc/washington/crime#:~:text=With%20a%20crime%20rate%20of%2(
- U.S. news & World Report (https://realestate.usnews.com/places/district-of-columbia/washington/crime)
- DC.gov (https://mpdc.dc.gov/page/district-crime-data-glance)
- CityData (https://www.city-data.com/crime/crime-Washington-District-of-Columbia.html)
- CrimeGrade (https://crimegrade.org/violent-crime-washington-dc-metro/)

In [ ]:
```
%%shell
jupyter nbconvert --to html --template full /content/Analysis_of_Crime_Data.ipynb
```

```
[NbConvertApp] Converting notebook /content/Analysis_of_Crime_Data.ipynb to html
[NbConvertApp] Writing 1366474 bytes to /content/Analysis_of_Crime_Data.html
```

Out[ ]: