Discipline: Big Data in Law Enforcement

Endterm. Report of a Project

Berikova Malika, Kurmangazynova Asiya

The objective of this project is to analyze a real dataset related to law enforcement using Numpy, Pandas, SQL, and Apache PySpark. For this analysis, we have chosen the "Crimes – 2023" dataset of Chicago crimes (https://data.cityofchicago.org/Public-Safety/Crimes-2023/xguy-4ndq/about_data), which provides comprehensive information about various crime incidents reported in Chicago over the years. The goal of this analysis is to gain insights into crime trends, identify high-crime areas, understand demographic patterns, and provide valuable information for law enforcement agencies to enhance public safety measures.

Data Preparation:

Load Dataset:

Ввод [2]:
```
# Load the dataset into Python environment
import pandas as pd
df = pd.read_csv("Crimes_-_2023.csv")
print(df.head(5))
```

```
         ID Case Number                    Date                Block  IUCR  \
0  13204489    JG416325  09/06/2023 11:00:00 AM       0000X E 8TH ST  0810
1  13045102    JG226663  03/30/2023 09:16:00 AM   080XX S DREXEL AVE  1544
2  13074891    JG262771  05/10/2023 12:43:00 PM    028XX N MANGO AVE  1754
3  13099339    JG291745  04/01/2023 11:13:00 AM  020XX N LAPORTE AVE  1751
4  13121127    JG313964  06/22/2023 06:52:00 PM    015XX W NORTH AVE  1153

                Primary Type  \
0                      THEFT
1                SEX OFFENSE
2  OFFENSE INVOLVING CHILDREN
3  OFFENSE INVOLVING CHILDREN
4          DECEPTIVE PRACTICE

                                      Description  \
0                                        OVER $500
1                    SEXUAL EXPLOITATION OF A CHILD
2  AGGRAVATED SEXUAL ASSAULT OF CHILD BY FAMILY M...
3             CRIMINAL SEXUAL ABUSE BY FAMILY MEMBER
4                FINANCIAL IDENTITY THEFT OVER $ 300

                    Location Description  Arrest  Domestic  ...  Ward  \
0  PARKING LOT / GARAGE (NON RESIDENTIAL)   False     False  ...   4.0
1                              APARTMENT   False      True  ...   8.0
2                              RESIDENCE   False      True  ...  30.0
3                              RESIDENCE   False      True  ...  26.0
```

The dataset has been successfully loaded into the Python environment, and initial data cleaning and preprocessing have been performed to ensure data quality and consistency using Pandas and Numpy.

Now let's see some basic information about the dataset with these functions:

```
# Let's see some basic information about the dataset.
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260968 entries, 0 to 260967
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   ID                    260968 non-null  int64  
 1   Case Number           260968 non-null  object 
 2   Date                  260968 non-null  object 
 3   Block                 260968 non-null  object 
 4   IUCR                  260968 non-null  object 
 5   Primary Type          260968 non-null  object 
 6   Description           260968 non-null  object 
 7   Location Description  259676 non-null  object 
 8   Arrest                260968 non-null  bool   
 9   Domestic              260968 non-null  bool   
 10  Beat                  260968 non-null  int64  
 11  District              260968 non-null  int64  
 12  Ward                  260965 non-null  float64
 13  Community Area        260968 non-null  int64  
 14  FBI Code              260968 non-null  object 
 15  X Coordinate          260883 non-null  float64
 16  Y Coordinate          260883 non-null  float64
 17  Year                  260968 non-null  int64  
 18  Updated On            260968 non-null  object 
 19  Latitude              260883 non-null  float64
 20  Longitude             260883 non-null  float64
 21  Location              260883 non-null  object 
dtypes: bool(2), float64(5), int64(5), object(10)
memory usage: 40.3+ MB
```

This method provides a concise summary of the DataFrame, including the column names, data types, non-null counts, and memory usage. It's useful for quickly understanding the structure of the dataset and identifying any missing values.

```
: # This method shows the number of rows, columns in a dataset
  print(df.shape)
```

```
(260968, 22)
```

This attribute returns a tuple representing the dimensions of the DataFrame, i.e., the number of rows and columns.

```
print(df.describe())
```

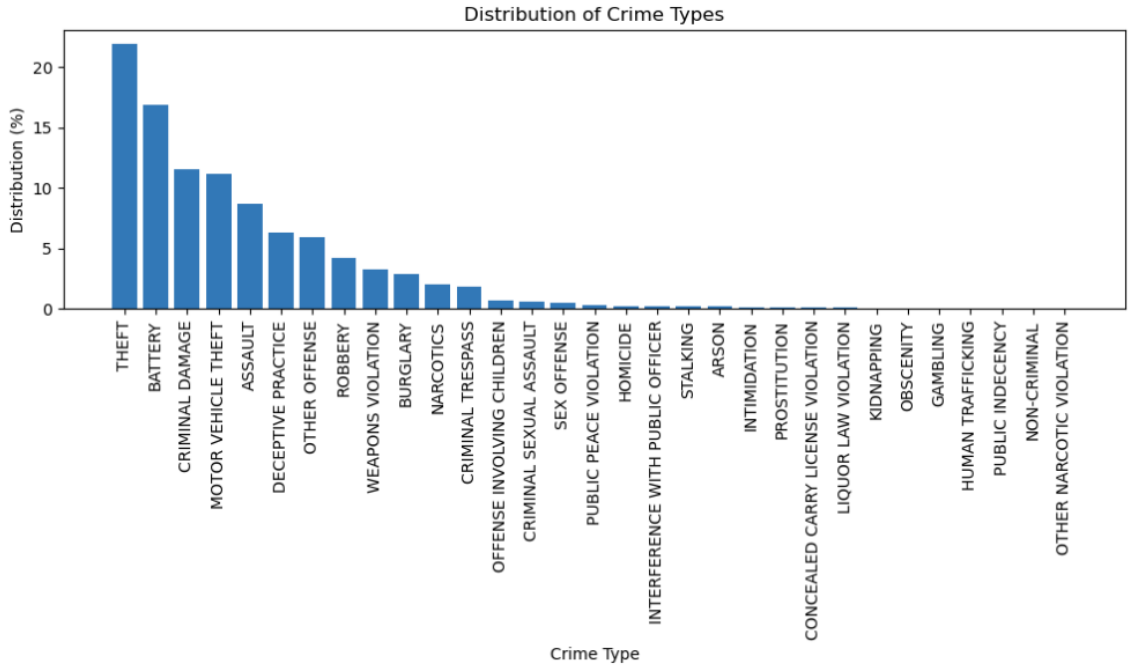
```
                ID           Beat       District           Ward  \
count  2.609680e+05  260968.000000  260968.000000  260965.000000   
mean   1.310459e+07    1155.843678      11.328991      23.132006   
std    6.499304e+05     712.344290       7.119471      14.009486   
min    2.727900e+04     111.000000       1.000000       1.000000   
25%    1.303878e+07     533.000000       5.000000      10.000000   
50%    1.313637e+07    1032.000000      10.000000      23.000000   
75%    1.323238e+07    1732.000000      17.000000      34.000000   
max    1.337538e+07    2535.000000      31.000000      50.000000   

       Community Area  X Coordinate  Y Coordinate      Year       Latitude  \
count   260968.000000  2.608830e+05  2.608830e+05  260968.0  260883.000000   
mean        36.276298  1.165332e+06  1.887358e+06    2023.0      41.846490   
std         21.576698  1.634465e+04  3.174150e+04       0.0       0.087288   
min          1.000000  1.091242e+06  1.813897e+06    2023.0      41.644590   
25%         22.000000  1.153935e+06  1.859887e+06    2023.0      41.770828   
50%         32.000000  1.167096e+06  1.893528e+06    2023.0      41.863529   
75%         53.000000  1.176796e+06  1.910412e+06    2023.0      41.909981   
max         77.000000  1.205119e+06  1.951506e+06    2023.0      42.022549   

           Longitude  
count  260883.000000  
mean      -87.668772  
std         0.059506  
min       -87.939733  
25%       -87.710265  
50%       -87.662228  
75%       -87.626819  
max       -87.524532
```

This method generates descriptive statistics for numerical columns in the DataFrame, such as count, mean, standard deviation, minimum, maximum, and quartiles. It's helpful for understanding the distribution of numerical data.

Data Analysis:

1. *Crime Type Analysis:*

```python
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt

# Connect to the SQLite database
conn = sqlite3.connect('crime_data.db')

# Crime Type Analysis
crime_type_query = """
    SELECT "Primary Type", COUNT(*) AS "Crime Count"
    FROM crimes
    GROUP BY "Primary Type"
    ORDER BY "Crime Count" DESC
"""
crime_type_analysis_df = pd.read_sql_query(crime_type_query, conn)

# Calculate the distribution of crime types as a percentage of total crimes
total_crimes = crime_type_analysis_df['Crime Count'].sum()
crime_type_analysis_df['Crime Distribution (%)'] = (crime_type_analysis_df['Crime Count'] / total_crimes) * 100

# Close the connection
conn.close()

# Plot the distribution of crime types
plt.figure(figsize=(10, 6))
plt.bar(crime_type_analysis_df['Primary Type'], crime_type_analysis_df['Crime Distribution (%)'])
plt.xlabel('Crime Type')
plt.ylabel('Distribution (%)')
plt.title('Distribution of Crime Types')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Here we used the **SQLite** to do the analysis, also the **pandas** and **matplotlib.**

The bar chart illustrates the distribution of different crime types, with "Theft" being the most common crime type, followed by "Battery" and "Criminal Damage." The Crime Type Analysis provides valuable insights into the distribution and frequency of different types of crimes occurring within a specific area or jurisdiction, it serves as a crucial tool for understanding the nature and extent of

crime in a community, guiding efforts to prevent and address crime effectively, and ultimately contributing to improved public safety and well-being.

2. *Location-Based Analysis:*

```
import sqlite3
import pandas as pd

# Connect to the SQLite database
conn = sqlite3.connect('crime_data.db')

# Location-Based Analysis
location_based_query = """
    SELECT "Location Description", COUNT(*) AS "Crime Count"
    FROM crimes
    GROUP BY "Location Description"
    ORDER BY "Crime Count" DESC
"""
location_based_analysis_df = pd.read_sql_query(location_based_query, conn)

# Close the connection
conn.close()

# Display the results
print("\nLocation-Based Analysis:")
print(location_based_analysis_df)
```

```
Location-Based Analysis:
                      Location Description  Crime Count
0                                   STREET        77542
1                                APARTMENT        48743
2                                RESIDENCE        31354
3                                 SIDEWALK        13021
4    PARKING LOT / GARAGE (NON RESIDENTIAL)         9947
..                                      ...          ...
128                             CHA LOBBY            1
129                           CHA HALLWAY            1
130                           CHA GROUNDS            1
131                                 BEACH            1
132               BARBER SHOP/BEAUTY SALON            1

[133 rows x 2 columns]
```

Implementation of this code gives us insights into the distribution of crimes across different location descriptions. It helps identify the most common locations where crimes occur, allowing law enforcement agencies to focus their resources and efforts on areas with higher crime rates.

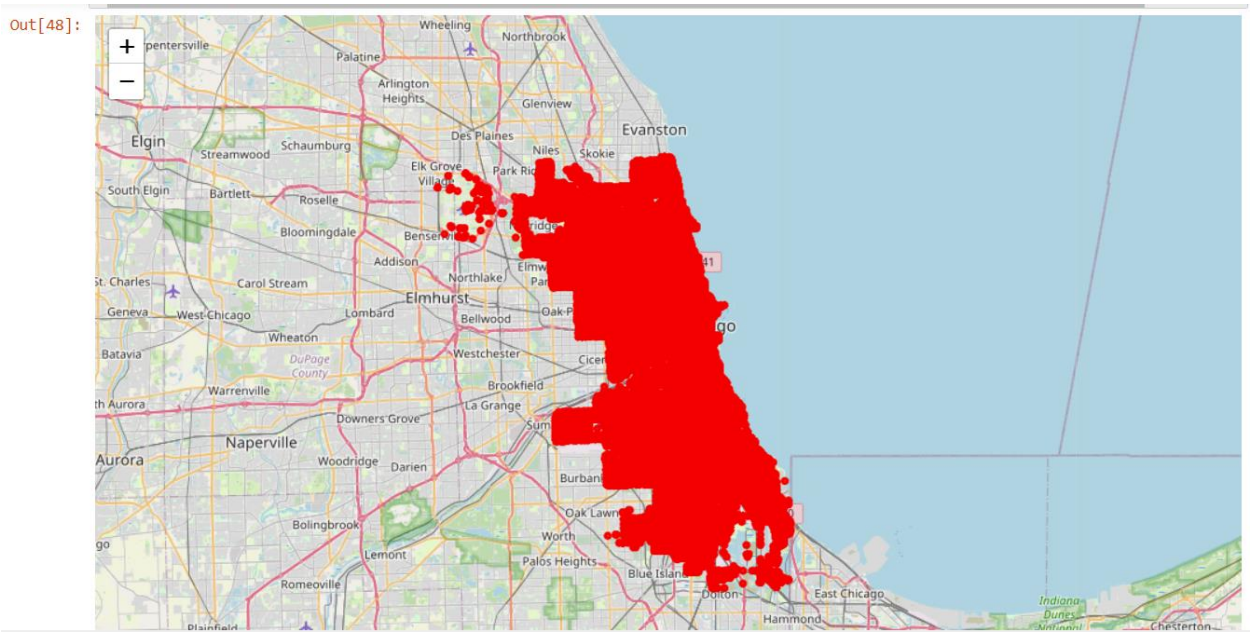
This analysis can also inform city planning and community safety initiatives by highlighting areas that may require additional attention or intervention.

3. *Spatial Analysis of Crime Hotspots:*

```
from pyspark.sql import SparkSession
import folium

# Initialize Spark session
spark = SparkSession.builder \
    .appName("SpatialAnalysis") \
    .getOrCreate()

# Load the dataset into a Spark DataFrame
df = spark.read.csv('Crimes_-_2023.csv', header=True, inferSchema=True)

# Select relevant columns containing latitude and longitude coordinates
location_df = df.select('Latitude', 'Longitude')

# Filter out rows with missing latitude or longitude values
location_df = location_df.filter(location_df['Latitude'].isNotNull() & location_df['Longitude'].isNotNull())

# Convert Spark DataFrame to Pandas DataFrame for visualization
location_pd_df = location_df.toPandas()

# Close the Spark session
spark.stop()

# Create a Folium map centered on the mean of latitude and longitude
crime_map = folium.Map(location=[location_pd_df['Latitude'].mean(), location_pd_df['Longitude'].mean()], zoom_start=10)

# Add markers for each crime hotspot
for index, row in location_pd_df.iterrows():
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']], radius=2, color='red', fill=True, fill_color='red').add_to(

# Display the map
crime_map
```

This code leverages Apache Spark for data loading and preprocessing and utilizes Folium for spatial visualization, allowing for efficient analysis and visualization of crime hotspots in the dataset.

Out[48]:



The Spatial Analysis of Crime Hotspots provides valuable insights into the geographical distribution and concentration of crime within a specific area or jurisdiction. It is critical for understanding the spatial distribution of crime, guiding targeted interventions and resource allocation, and fostering collaboration between law enforcement, policymakers, urban planners, and community stakeholders to create safer and more resilient communities.

Conclusion:

In conclusion, the analyses conducted on the crime dataset “Crimes_–_2023.csv” for Chicago in 2023 have yielded valuable insights into various aspects of crime within the city. The Crime Type Analysis revealed the distribution and frequency of different types of crimes, aiding in prioritizing resources for crime prevention. The Location-Based Analysis provided insights into the spatial patterns of criminal activity, guiding targeted interventions in high-crime areas. The Spatial Analysis of Crime Hotspots identified specific geographic areas with high concentrations of criminal activity, informing strategies to improve public safety. Overall, these analyses contribute to a better understanding of crime trends and support evidence-based decision-making for crime prevention and intervention efforts in Chicago.

The GitHub link:

https://github.com/molik-molik/Endterm_Big_Data