******

*The NUST logo is protected by copyright and trademarks*

**National University of Sciences & Technology**

**Complex Engineering Problem**

**Introduction to Data Science**

**GIE 409**

**Submitted by:**

**Hijab Zahra\_421132 and Faatima Aamir\_404214**

**Submitted to:**

**Sir Abdullah bin Ahmed**

**Introduction:**

As the population is increasing, so does the different problems, and the major one is the increase in crime rate. Since, the past twenty years, crime rate has increased significantly, the crimes varying from a small street robbery to large drug dealings and even murders. So, in order to deal with the problem in future, we predicted the crime hotspots based on spatial and historical data. We visualised the crime data by using different techniques to analyze the trends and to get the better understanding of the data. The final results were in the form of hotspots-areas where crime is most likely to occur- and krigging maps-areas divided in polygons for most to least likeliness of crimes.

**Objective:**

The aim was to accurately detect and predict the areas that would most likely be classified as crime hotspot, this can largely help the security departments to control and have an efficient plan, and the local people to keep them aware of crime hotspots.

**Dataset Overview:**

We took the dataset of “Chicago, Illinois, US ” from Kaggle, it included ID, Date, Month, Year, Latitude, Longitude, Location Description, Arrest(True/False), District Number, Beat Number(Patrol Car Number), Domestic(Yes/No), Community Area Number. The time period for the dataset was from 2000 to 2012, and it originally had 191641 rows, and after cleaning the new dataset had 189365 rows.

**Methodology:**

***Data Cleaning and Preprocessing:***

The tools used were:

* Python (Jupyter Lab)- Libraries (pandas, numpy)

**Steps:**

* Importing the libraries required:

import pandas as pd

import numpy as np

* Loading the csv file:

file\_path = 'CrimeData.csv'

df = pd.read\_csv(file\_path, low\_memory = False)

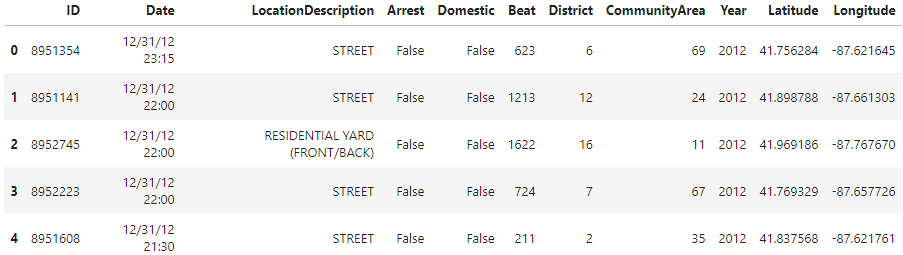
* the Dataset:

print("Initial Data Shape:", df.shape)

df.head()

*Output:*





* Printing the columns to know the Column Titles, so they are easily referenced:

print(df.columns)

*Output:*



* Dropping the Null Values:

df.dropna(subset=['Latitude', 'Longitude'], inplace=True)

print("After dropping nulls:", df.shape)

*Output:*



* Correcting the date format for Date Column:

df['Date'] = pd.to\_datetime(df['Date'], format='%m/%d/%Y %I:%M:%S %p', errors='coerce')

df = df[df['Date'].notna()]

* Extracting the Time features:

df['Year'] = df['Date'].dt.year

df['month'] = df['Date'].dt.month

df['day'] = df['Date'].dt.day

df['hour'] = df['Date'].dt.hour

* Saving the cleaned data:

# ========== Step 9: Save Cleaned Dataset ==========

df.to\_csv('New\_CrimeData.csv', index=False)

print("Cleaned data saved as 'New\_CrimeData.csv'.")

*Output:*



* Generating a summary ,to get a quick overview:

print("\nTop Crime Types:")

print(df['Arrest'].value\_counts().head())

print("\nDate Range:", df['Date'].min(), "to", df['Date'].max())

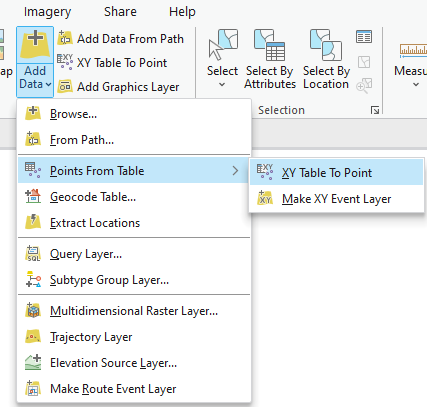
*Output:*



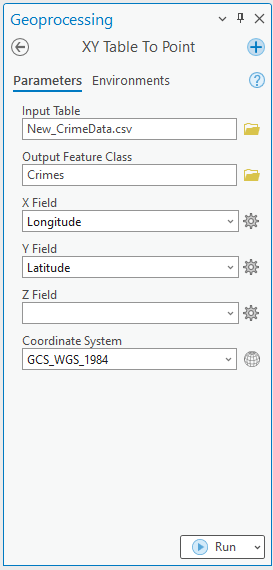
***Spatial Analysis in ArcGIS pro:***

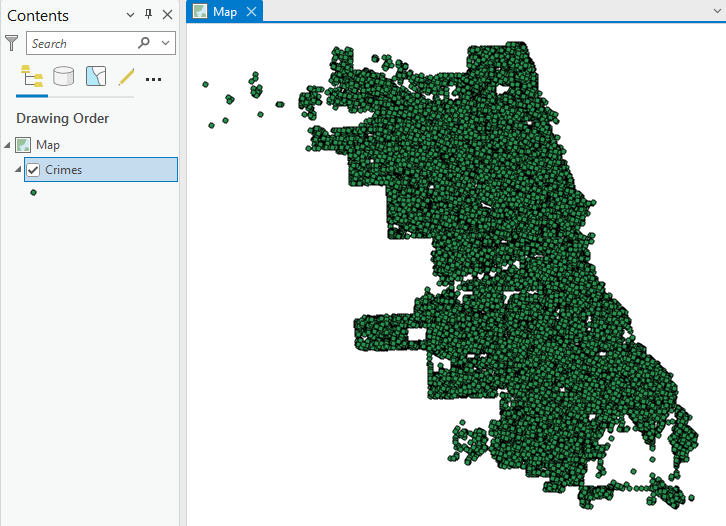
Steps:

* Importing Data into ArcGIS Pro:

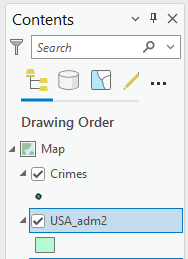


* Importing the State and Cities shapefile:

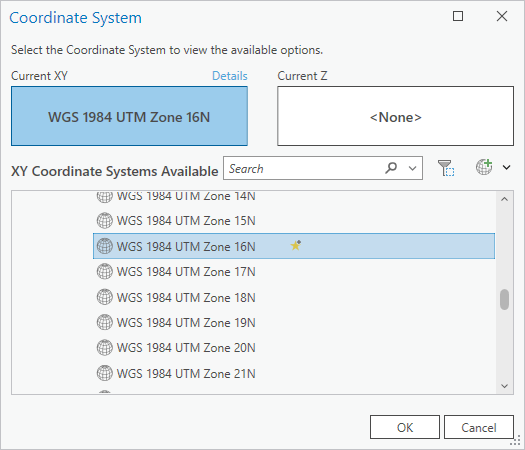




* Added the USA administration boundary-Level 2 (Cities), after extracting the boundary for Chicago by drag drop:

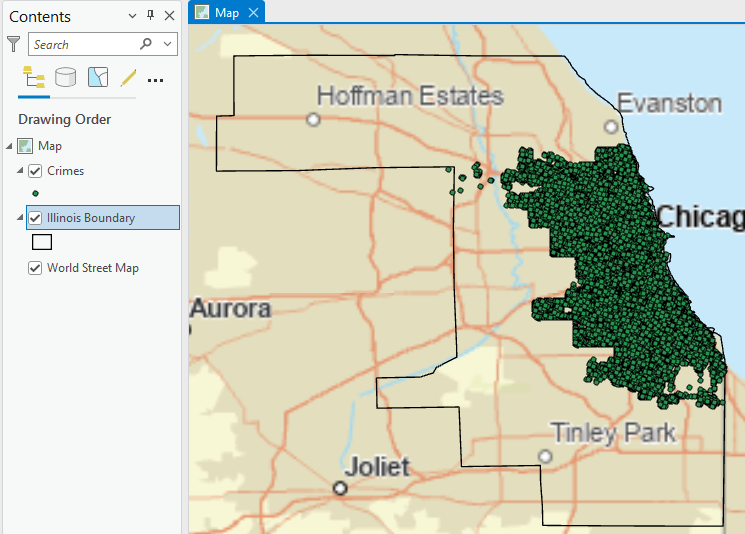


* Setting the Coordinate Reference system to UTM Zone 16, which covers the area of Chicago:



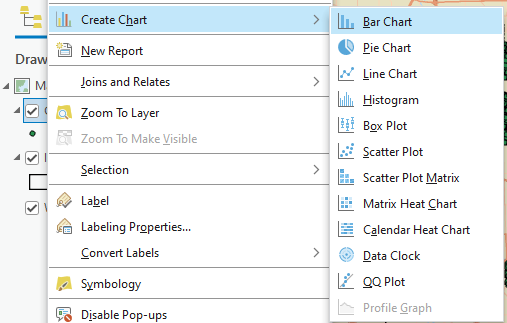
* Projecting the shapefiles to the set coordinate system, to have no spatial referencing errrors:





***Exploratory Data Analysis:***

* Creating the charts:



* + Bar Chart- Crime counts by Year and Month:

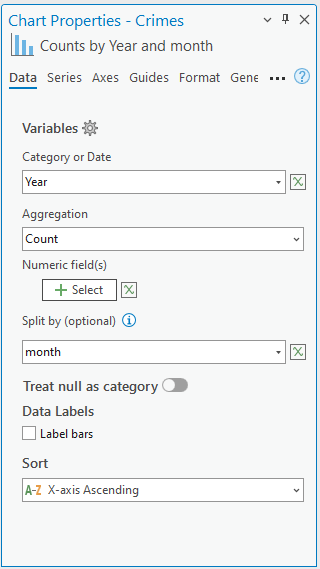
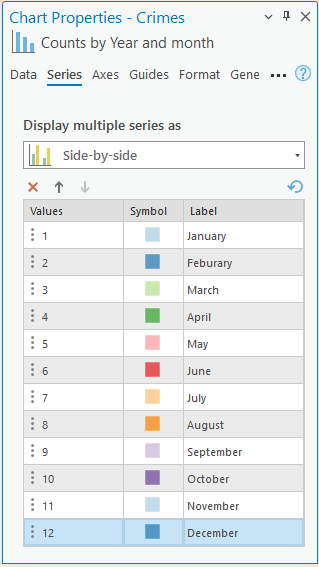
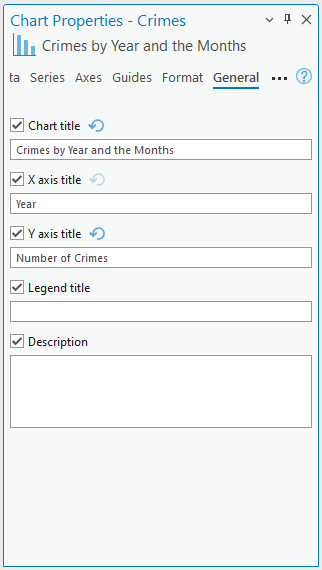
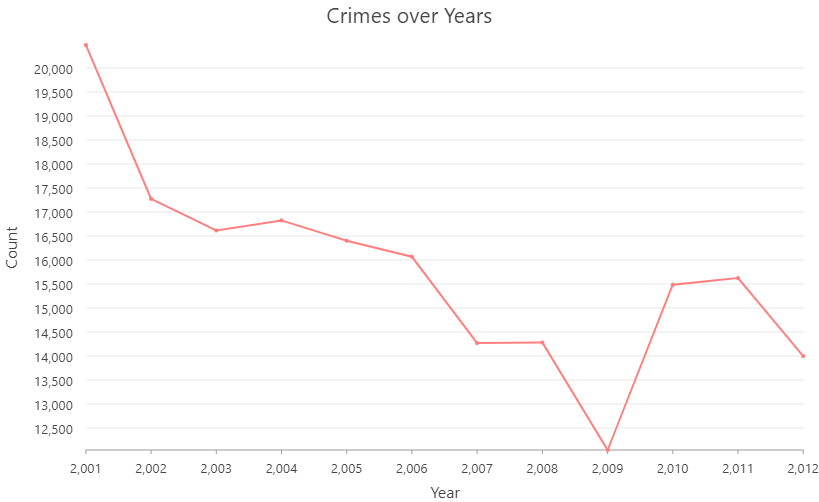
  

Figure : Formating options

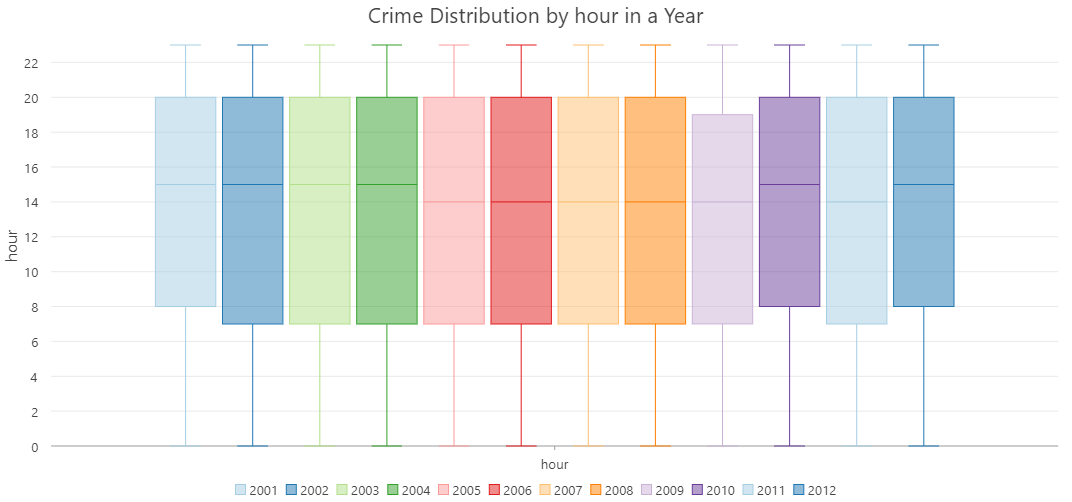
*Output:*



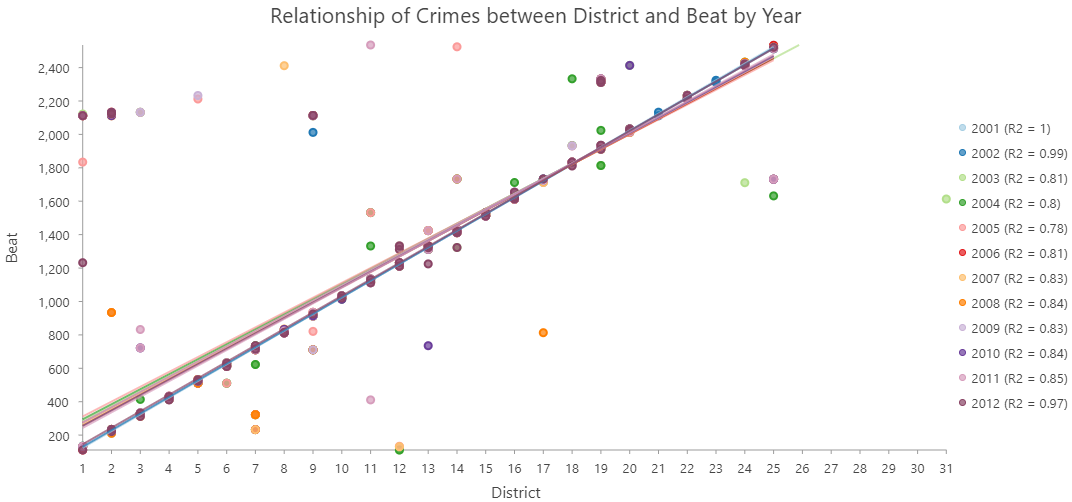
* + Line Graph- Count of crimes over the years

*Output:*

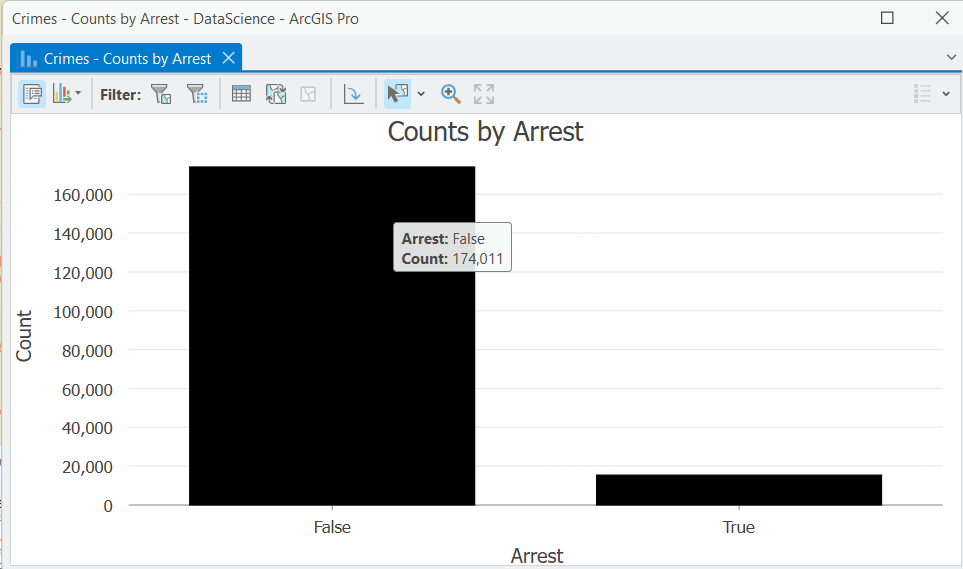
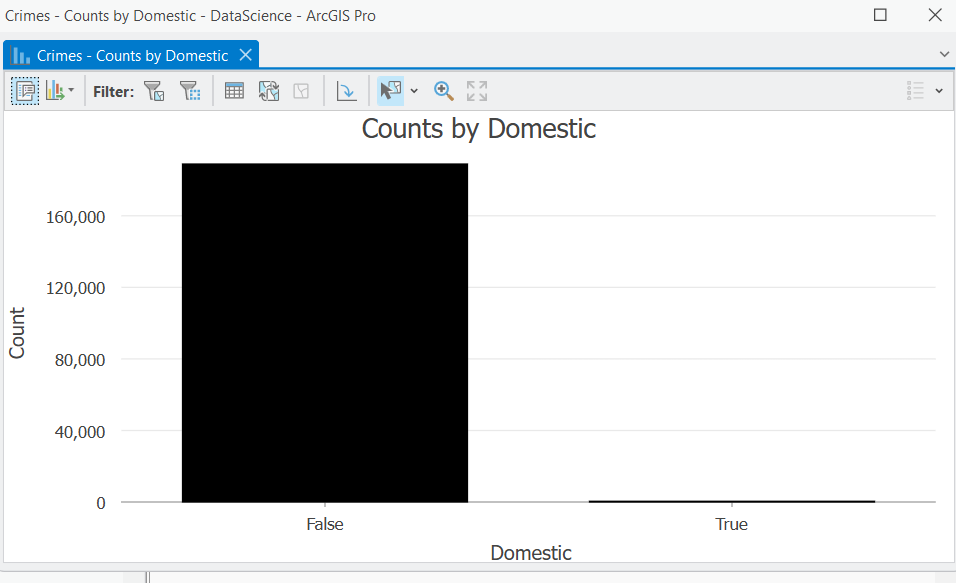
* + Boxplot- Crime Distribution by hours in each Year

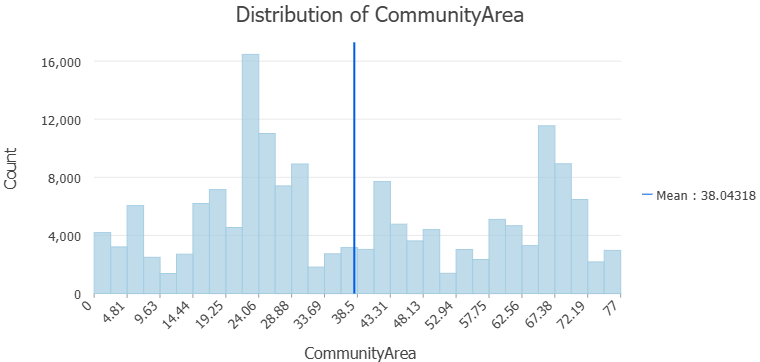
*Output:*

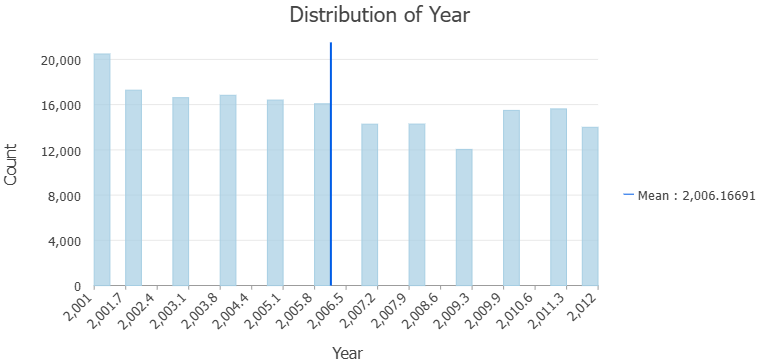
* + Scatterplot- Relationship of Crimes to Beats in a District by Year

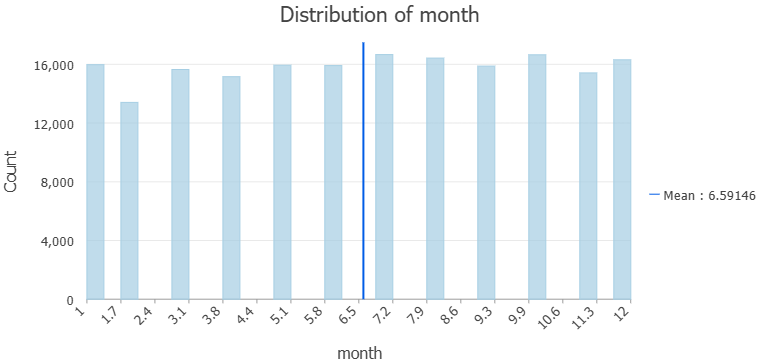
*Output:*

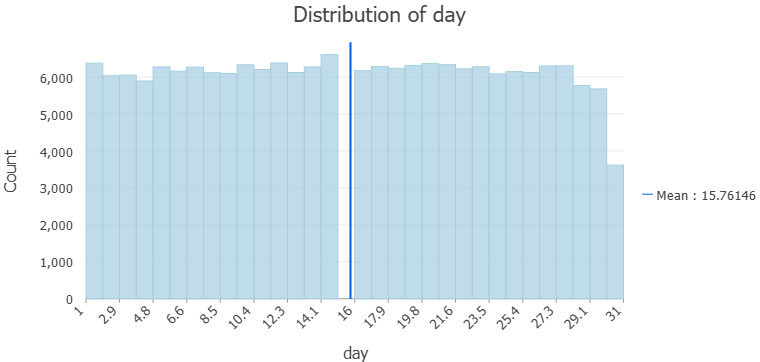
***Summary statistics:***

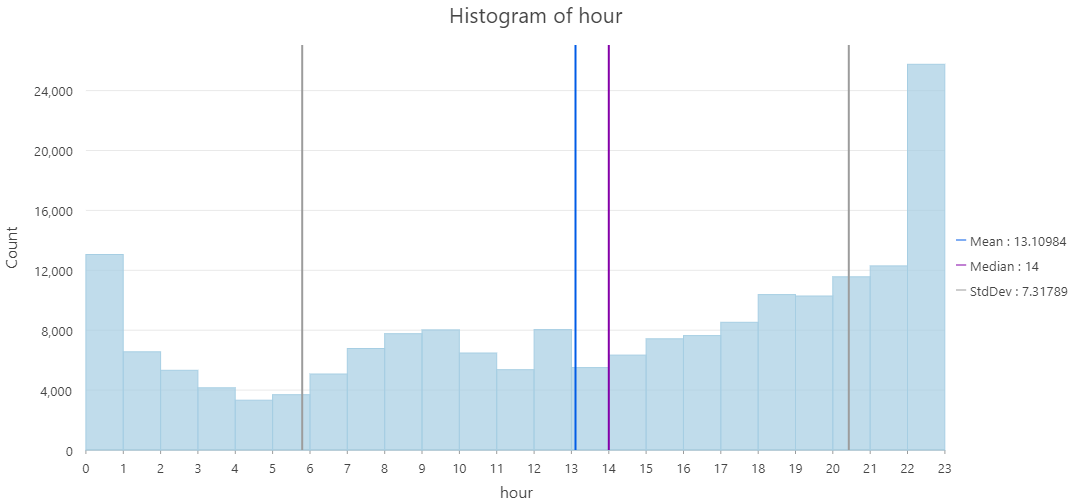
 

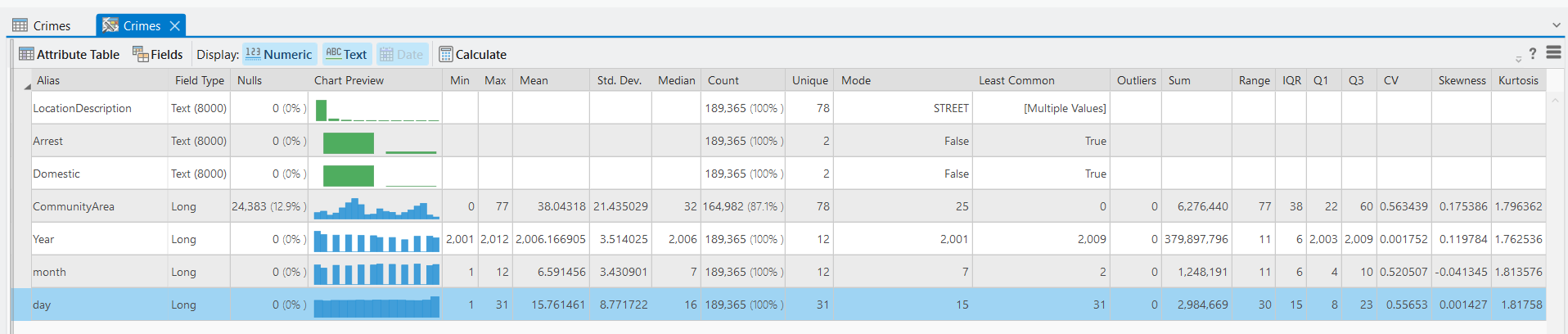


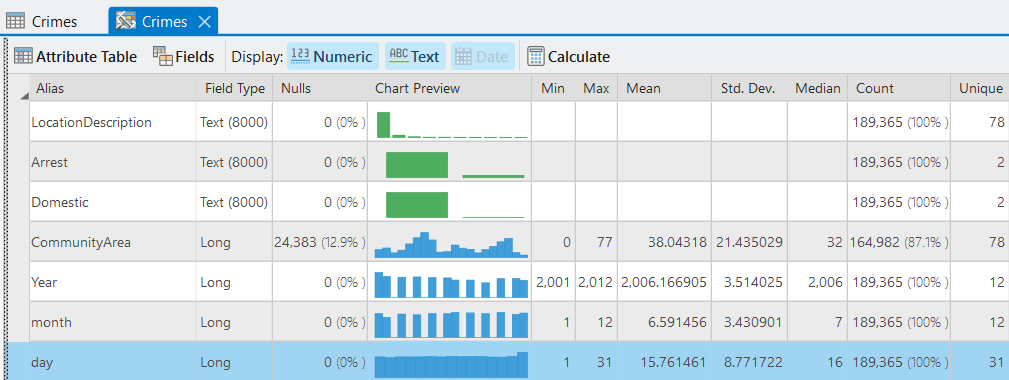


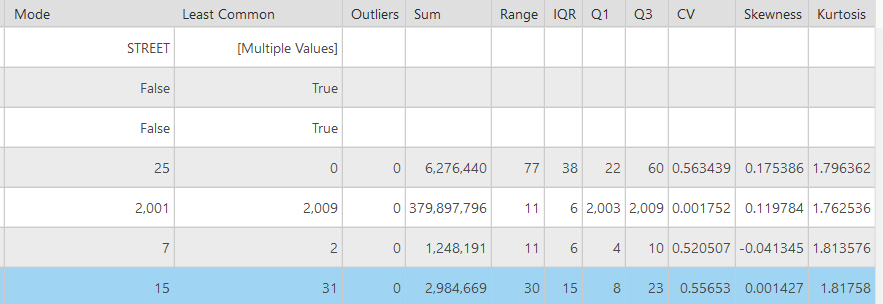






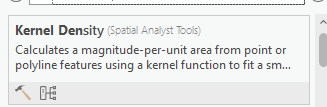


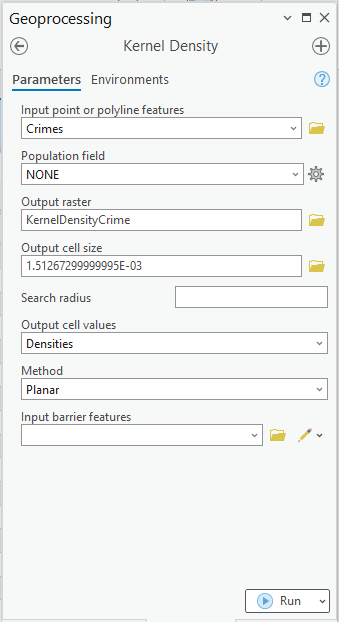




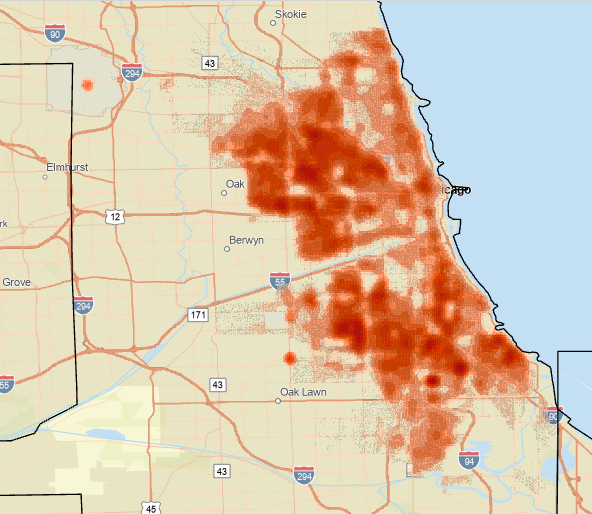
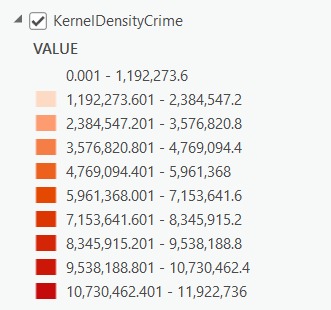
***Hotspot Analysis:***

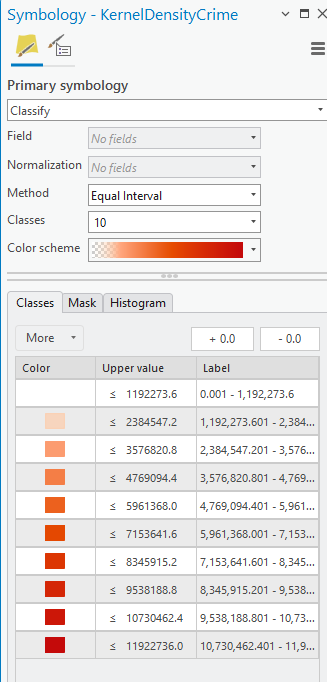
* Generate a heat map through kernel density by Year

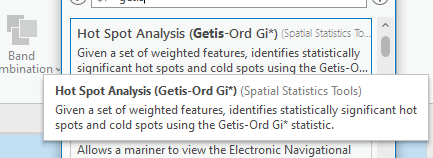




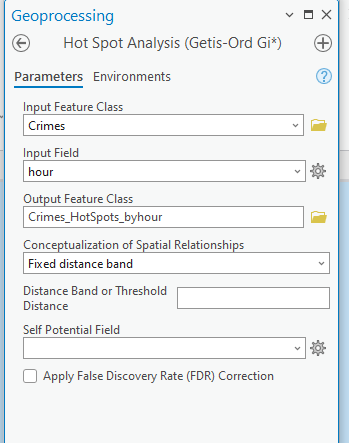
*Output:*



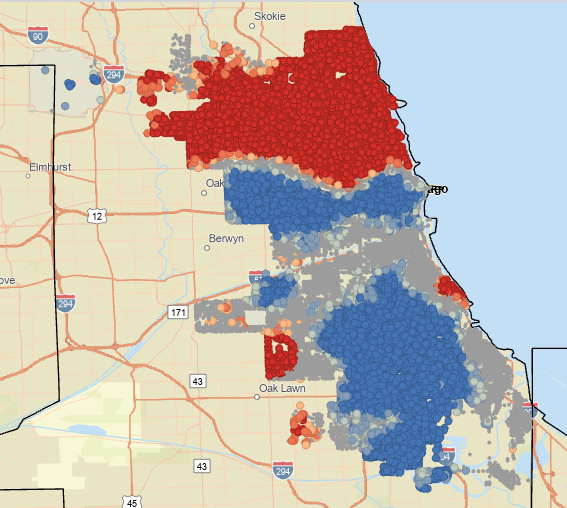
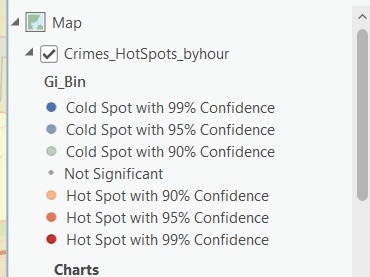


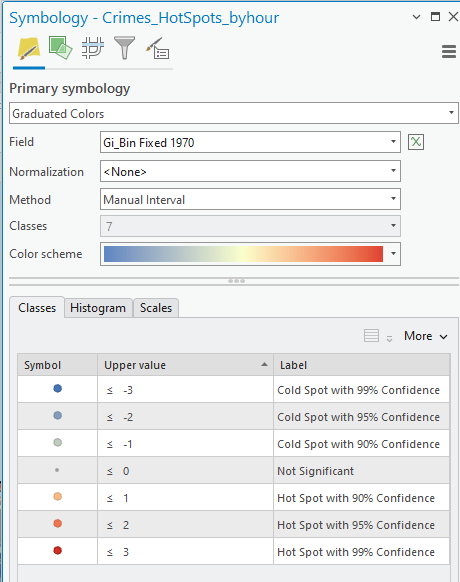
* Hotspot detection using spatial statistics (Getis Ord Gi\*) 

By hour



*Output:*





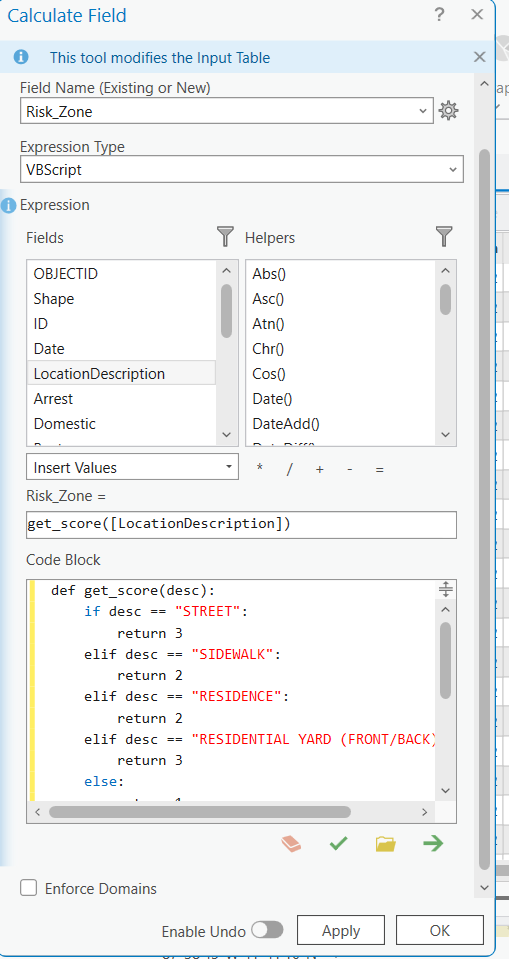
***Predictive Modelling through Kriging:***

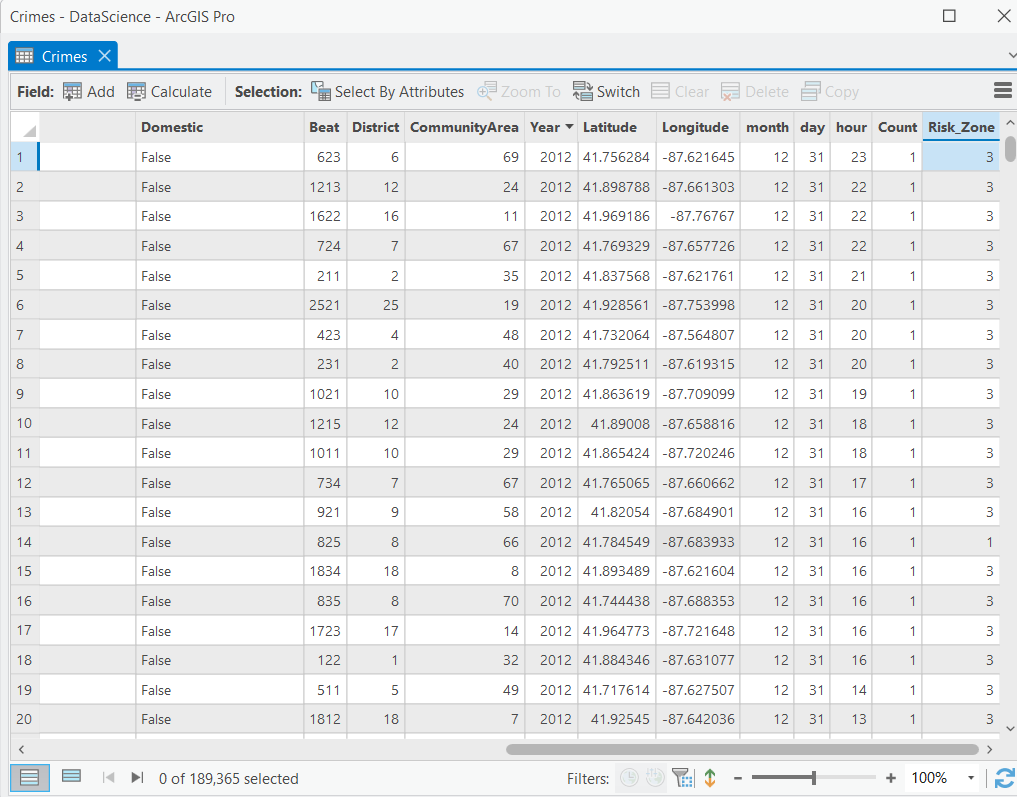
**What is Kriging?**

Geo-statistical method for spatial interpolation, by specifying value of a variable at unobserved locations.

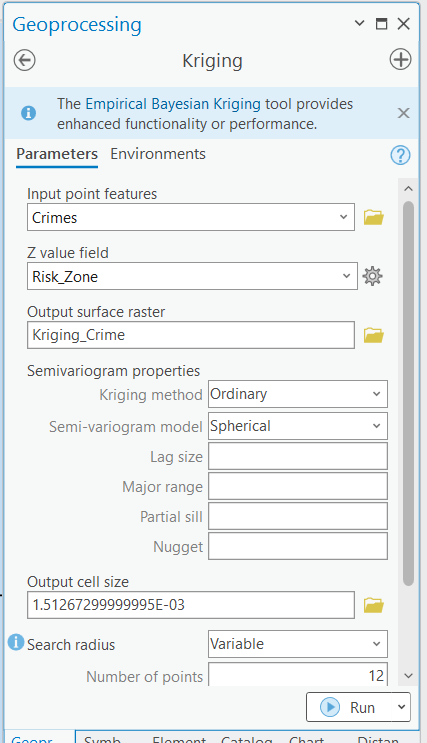
**Steps:**

* Calculation a new field ‘Risk Zone’ to assign the values according to risk, based on location description:



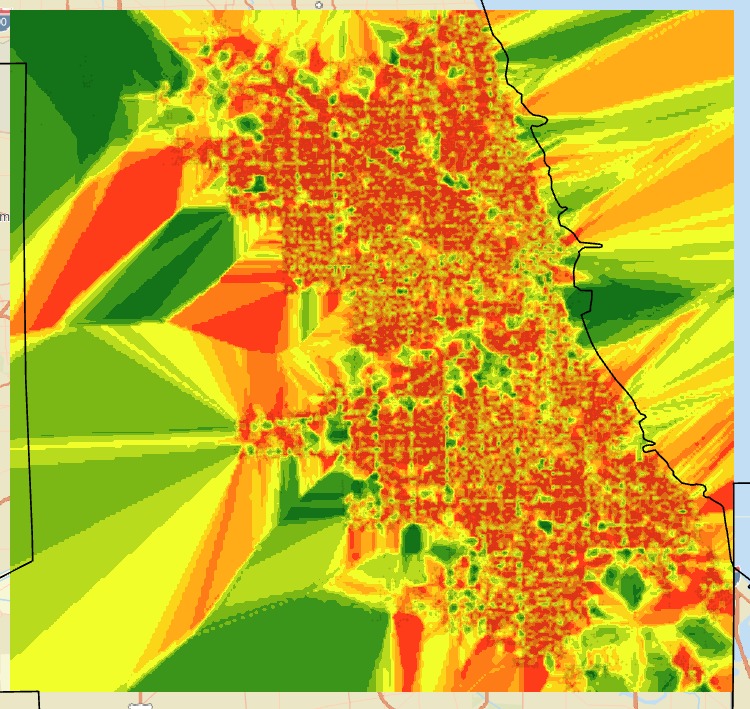


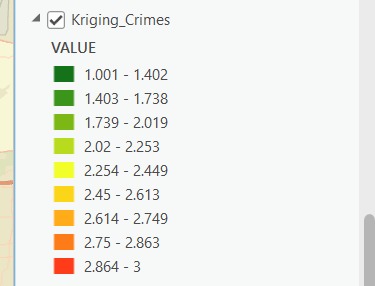
* Using the Kriging tool:





The kriging result shows how much risk is in each zone according to the type of crime. This has generated a predictive surface that shows the hotspots for crime based on the crime type. This is useful since classification of the crime type has been done on the basis of 1 to 3 classes (going from low risk to high risk). This helps us develop a relationship between crime types along with hotspot zones.

*Output:*



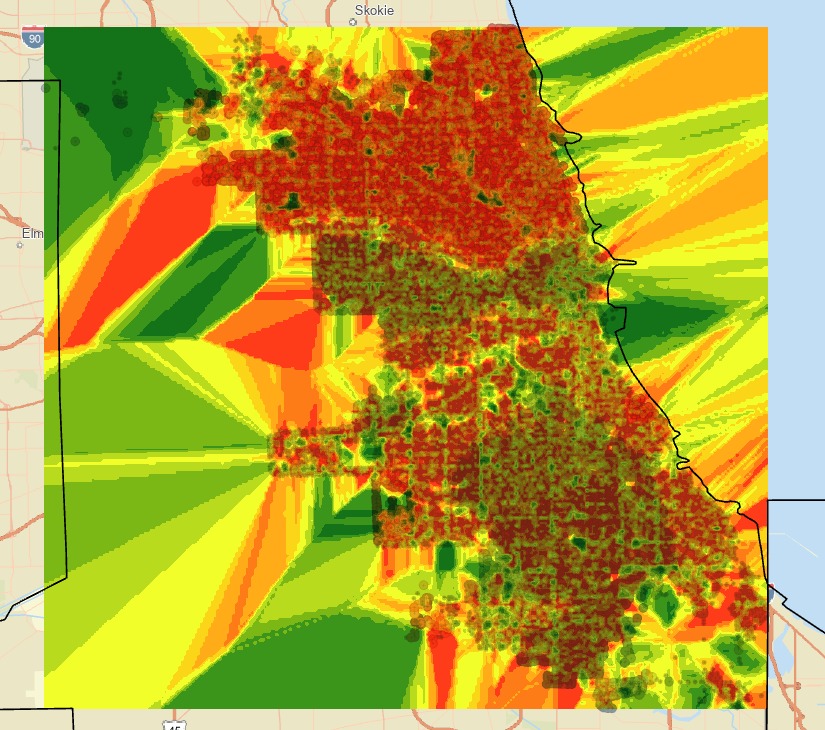
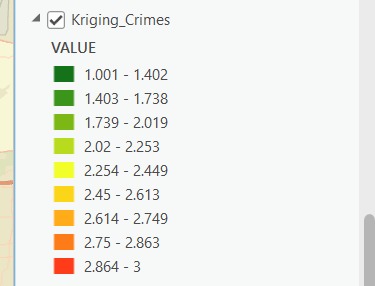


Figure : Final Overlayed Result for Kriging and Getis ord

***Predictive Modelling:***

We made use of jupyter notebook for predictive modelling. Initially, our intention was to use ArcGIS Pro Notebook with python but unfortunately we didn’t have it in our license so we switched back to jupyter notebook.

**Steps:**

* Import Libraries (pandas, numpy, sklearn, matplot, joblib)

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import joblib

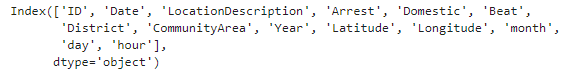
* Load Cleaned Crime Dataset

csv\_path = "New\_CrimeData.csv"

df = pd.read\_csv(csv\_path, low\_memory=False)

print(df.columns)

*Output:*



* Drop null spatial values to create a feature set

df\_geo = df.dropna(subset=['Latitude', 'Longitude'])

* Feature Engineering

hotspot\_crimes = ["STREET", "RESIDENTIAL YARD (FRONT/BACK)"]

df\_geo['is\_hotspot'] = df\_geo['LocationDescription'].apply(lambda x: 1 if x in hotspot\_crimes else 0)

# Select features and labels

features = df\_geo[['Year', 'month', 'hour']]

labels = df\_geo['is\_hotspot']

* Train Test Split

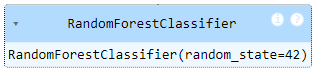
X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.3, random\_state=42)

* Train ML Model (Using Random Forest)

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

*Output:*



* Evaluate Model

y\_pred = clf.predict(X\_test)

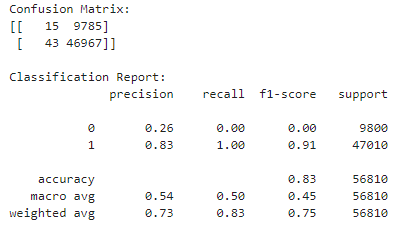
print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

*Output:*



* Visualize the prediction Results

# Step 8: Visualize Prediction Results (Clean Bar Chart)

%pip install seaborn

import seaborn as sns

# Count occurrences of 0 and 1

from collections import Counter

pred\_counts = Counter(y\_pred)

actual\_counts = Counter(y\_test)

# Align bar heights

labels = [0, 1]

pred\_vals = [pred\_counts.get(label, 0) for label in labels]

actual\_vals = [actual\_counts.get(label, 0) for label in labels]

# Plot side-by-side bars

x = np.arange(len(labels))

width = 0.35

fig, ax = plt.subplots()

bar1 = ax.bar(x - width/2, actual\_vals, width, label='Actual')

bar2 = ax.bar(x + width/2, pred\_vals, width, label='Predicted')

ax.set\_xticks(x)

ax.set\_xticklabels(['Not Hotspot (0)', 'Hotspot (1)'])

ax.set\_ylabel('Number of Records')

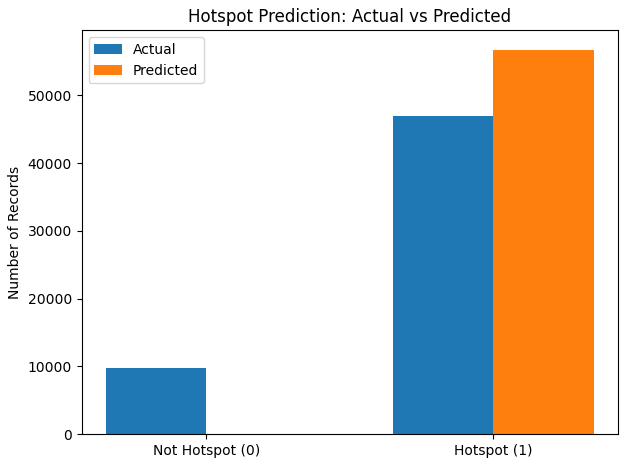
ax.set\_title('Hotspot Prediction: Actual vs Predicted')

ax.legend()

plt.tight\_layout()

plt.show()

*Output:*



* Export the predictions for visulization

df\_geo['predicted\_hotspot'] = clf.predict(features)

output\_path = 'predicted\_hotspots.csv' # Replace with your path

df\_geo.to\_csv(output\_path, index=False)

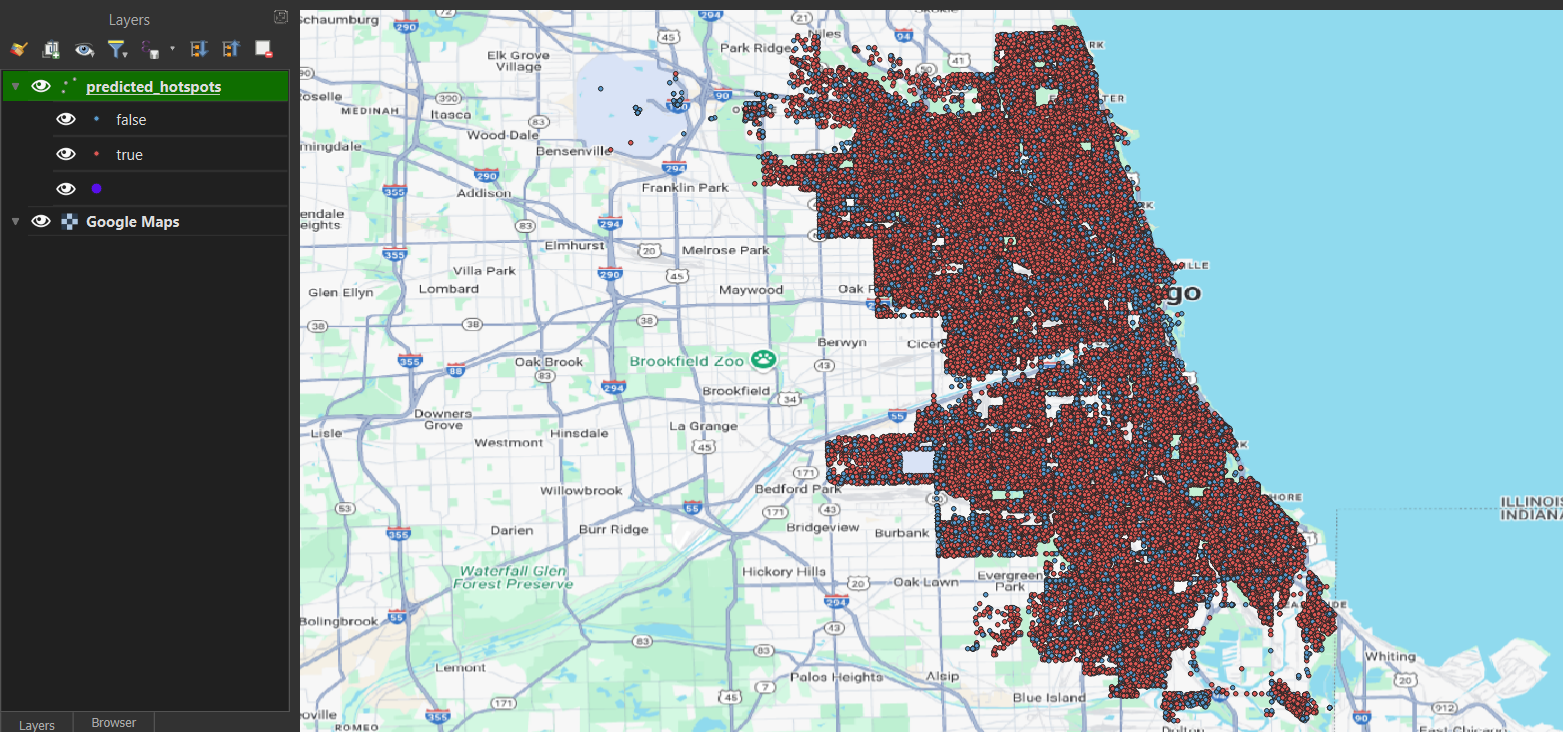
print(f"Predicted crime data saved to: {output\_path}")

*Output:*

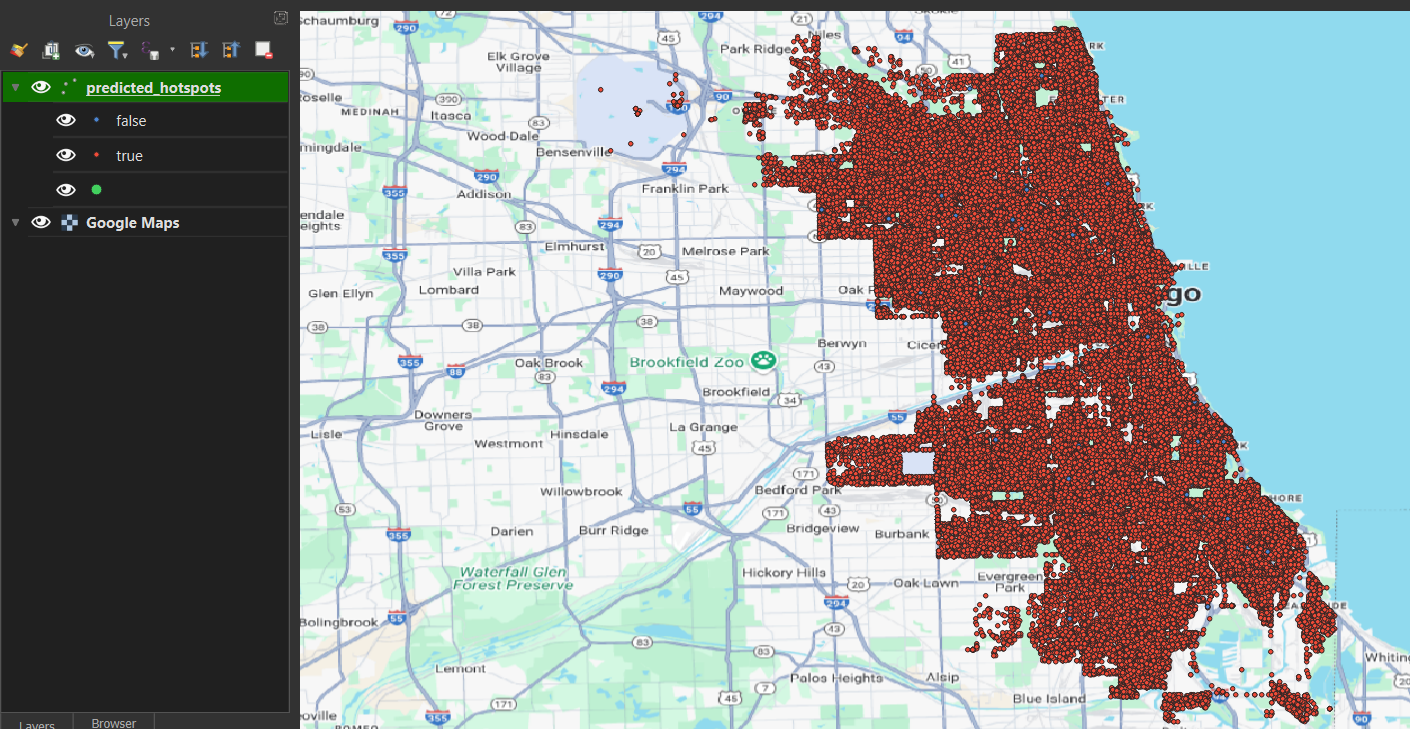


***Visualized in QGIS:***

* This shows if the actual (is\_hotspot):



* This is the predicted hotspot:



After visualizing both it can be seen that the model is predicting more hotspots as true than they actually are.

* Bonus: Calculated summary statistics:

print("=== Summary Statistics ===")

print(df\_geo[['Year', 'month', 'hour']].describe())

print("\n=== Top Crime Locations ===")

print(df\_geo['LocationDescription'].value\_counts().head(10))

print("\n=== Hotspot Class Distribution ===")

print(df\_geo['is\_hotspot'].value\_counts())

*Output:*

