Chicago_3

August 27, 2024

1 Data Management and Predictions

Let's create an advanced aggregated dataset and then use it for predictive analysis. We'll do this in two steps: first, we'll aggregate the data and save it as a CSV file, and then we'll use that file for our predictive analysis.

1.1 1. Aggregation

This script creates a panel data by the following steps:

1. Loads the original crime data.
2. Creates a function to aggregate the data, counting total crimes, total arrests, and individual counts for each crime type and location description.
3. Aggregates the data by District and Date.
4. Saves the aggregated data to a new CSV file.

[1]:

```
import pandas as pd
import numpy as np
```

[10]:

```
# Load the original data
file_path = '/Users/YigitAydede/Library/CloudStorage/Dropbox/Documents/Courses/
  ↪MBAN/NLPBootcamp/PythonBC/Crimes_-_2024_20240804.csv'
df = pd.read_csv(file_path, parse_dates=['Date'])

# Convert Date to datetime
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y %I:%M:%S %p',␣
  ↪errors='coerce')
```

```
/var/folders/b2/gpnsjh9j6bv5prtx7w5lsym80000gp/T/ipykernel_85973/3313344229.py:3
: UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  df = pd.read_csv(file_path, parse_dates=['Date'])
```

[13]:

```
def aggregate_data(group):
    # Initialize the dictionary with total crime and total arrest
    agg = {
        'Total_Crime': group['ID'].count(),
        'Total_Arrest': group['Arrest'].sum()
    }

    # Add counts for each crime type
    for crime_type in df['Primary Type'].unique():
        agg[f'Crime_{crime_type.replace(" ", "_")}'] = (group['Primary Type']␣
  ↪== crime_type).sum()

    # Handle NaN values in 'Location Description' and add counts for each␣
  ↪location description
    group['Location Description'] = group['Location Description'].
  ↪fillna('Unknown')
    for location in df['Location Description'].unique():
        agg[f'Location_{location.replace(" ", "_")}'] = (group['Location␣
  ↪Description'] == location).sum()

    return pd.Series(agg)
```

Ask Copilot for help to see what this code does.

[14]:

```
# Aggregate data by District and Date
aggregated_df = df.groupby(['District', df['Date'].dt.date]).
  ↪apply(aggregate_data)
```

```
/var/folders/b2/gpnsjh9j6bv5prtx7w5lsym80000gp/T/ipykernel_85973/1280444553.py:2
: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns.
This behavior is deprecated, and in a future version of pandas the grouping
columns will be excluded from the operation. Either pass `include_groups=False`
to exclude the groupings or explicitly select the grouping columns after groupby
to silence this warning.
  aggregated_df = df.groupby(['District',
df['Date'].dt.date]).apply(aggregate_data)
```

[15]:

```
print(aggregated_df.head())
```

```
                     Total_Crime  Total_Arrest  Crime_MOTOR_VEHICLE_THEFT  \
District Date
1        2024-01-01           34             9                          4
         2024-01-02           29            11                          2
         2024-01-03           39             3                          3
         2024-01-04           34             7                          0
         2024-01-05           41             8                          0

                     Crime_DECEPTIVE_PRACTICE  Crime_THEFT  Crime_BATTERY  \
District Date
1        2024-01-01                         1            8             12
         2024-01-02                         2            9              5
         2024-01-03                         6           18              3
         2024-01-04                         4           11              7
         2024-01-05                         7           15              4

                     Crime_SEX_OFFENSE  Crime_CRIMINAL_DAMAGE  \
District Date
1        2024-01-01                  0                      0
         2024-01-02                  0                      2
         2024-01-03                  0                      1
         2024-01-04                  0                      1
         2024-01-05                  0                      1

                     Crime_CRIMINAL_SEXUAL_ASSAULT  \
District Date
1        2024-01-01                              0
         2024-01-02                              0
         2024-01-03                              0
         2024-01-04                              0
         2024-01-05                              0

                     Crime_OFFENSE_INVOLVING_CHILDREN  …  \
District Date                                          …
1        2024-01-01                                 0  …
         2024-01-02                                 0  …
         2024-01-03                                 0  …
         2024-01-04                                 0  …
         2024-01-05                                 0  …

                     Location_CREDIT_UNION  \
District Date
1        2024-01-01                      0
         2024-01-02                      0
         2024-01-03                      0
         2024-01-04                      0
         2024-01-05                      0

                     Location_VEHICLE_-_COMMERCIAL:_TROLLEY_BUS  \
District Date
1        2024-01-01                                           0
         2024-01-02                                           0
         2024-01-03                                           0
         2024-01-04                                           0
         2024-01-05                                           0

                     Location_HALLWAY  Location_GARAGE  Location_OFFICE  \
District Date
1        2024-01-01                 0                0                0
         2024-01-02                 0                0                0
         2024-01-03                 0                0                0
         2024-01-04                 0                0                0
         2024-01-05                 0                0                0

                     Location_RETAIL_STORE  Location_LIQUOR_STORE  \
District Date
1        2024-01-01                      0                      0
         2024-01-02                      0                      0
         2024-01-03                      0                      0
         2024-01-04                      0                      0
         2024-01-05                      0                      0

                     Location_CHA_HALLWAY  Location_CTA_"L"_TRAIN  \
District Date
1        2024-01-01                     0                       0
         2024-01-02                     0                       0
         2024-01-03                     0                       0
         2024-01-04                     0                       0
         2024-01-05                     0                       0

                     Location_CTA_TRACKS_-_RIGHT_OF_WAY
District Date
1        2024-01-01                                   0
         2024-01-02                                   0
         2024-01-03                                   0
         2024-01-04                                   1
         2024-01-05                                   0

[5 rows x 156 columns]
```

[16]:

```python
# Write the aggregated DataFrame to a CSV file
output_file_path = 'aggregated_crime_data.csv'  # Replace with your desired output file path
aggregated_df.to_csv(output_file_path)
```

1.2 2. Chicago Crime Predictive Analysis

This script performs the following analysis:

1. Calculates and visualizes a correlation matrix for all crime types.
2. Defines a function to predict one crime type based on another using linear regression.
3. Finds the most correlated crime types and attempts to predict one based on the other.
4. As an example, it also predicts ASSAULT based on THEFT.

[17]:

```python
import pandas as pd
import numpy as np
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.metrics import mean_squared_error, r2_score
```

[18]:

```
# Load the aggregated data
file_path = '/Users/YigitAydede/Library/CloudStorage/Dropbox/Documents/Courses/
 ↪MBAN/NLPBootcamp/PythonBC/aggregated_crime_data.csv'
df = pd.read_csv(file_path)
```

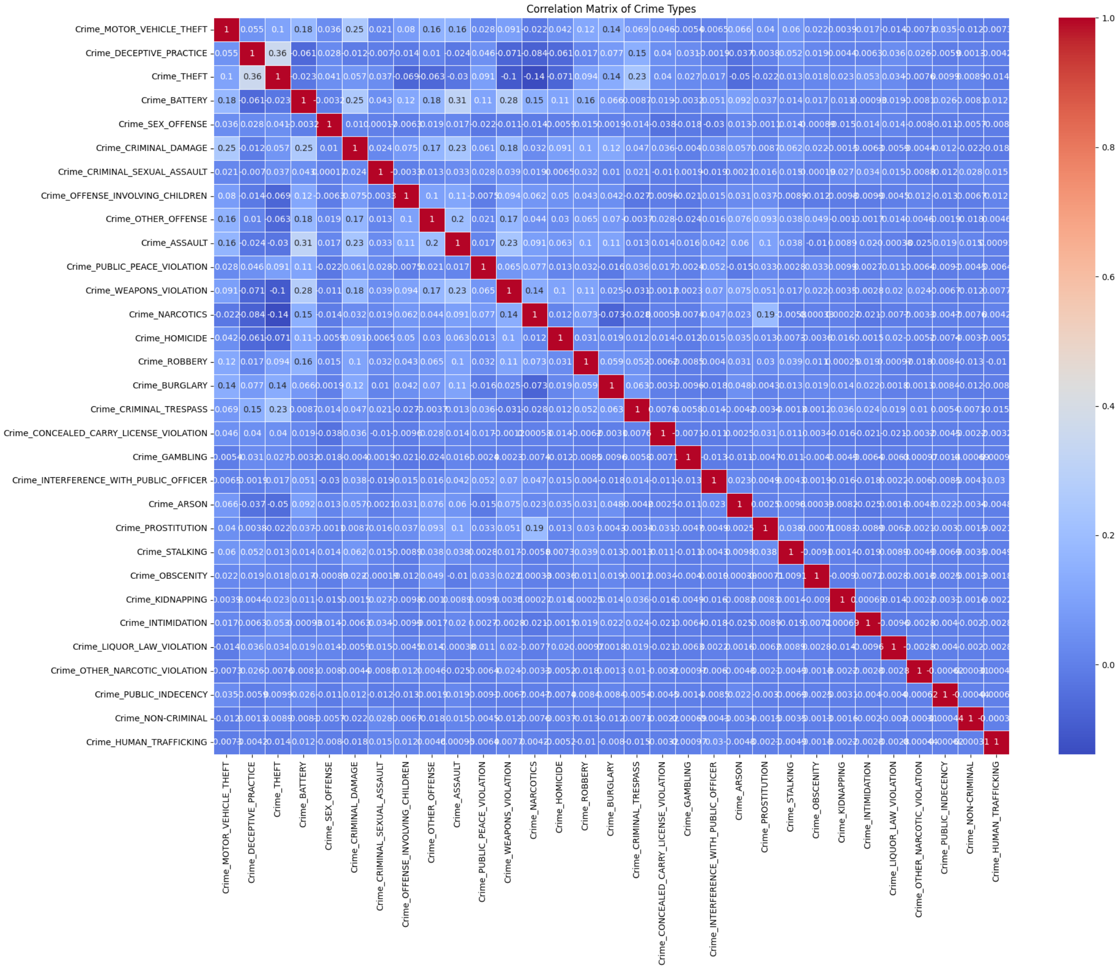
1.2.1 2.1 Correlation

[19]:

```
# Select crime type columns (assuming they start with 'Crime_')
crime_columns = [col for col in df.columns if col.startswith('Crime_')]

# Calculate correlation matrix for crime types
correlation_matrix = df[crime_columns].corr()

# Visualize the correlation matrix
plt.figure(figsize=(20, 16))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Crime Types')
plt.tight_layout()
plt.show()
```



```
[23]: # Select crime type columns (assuming they start with 'Crime_')
crime_columns = [col for col in df.columns if col.startswith('Crime_')]

# Calculate correlation matrix for crime types
correlation_matrix = df[crime_columns].corr()

# Find pairs with correlation greater than 0.2
correlation_threshold = 0.2
high_correlation_pairs = []
for i in range(len(crime_columns)):
    for j in range(i+1, len(crime_columns)):
        if correlation_matrix.iloc[i, j] > correlation_threshold:
            high_correlation_pairs.append((crime_columns[i], crime_columns[j],␣
 ↪correlation_matrix.iloc[i, j]))

# Print the high correlation pairs
for pair in high_correlation_pairs:
    print(f"Pair: {pair[0]} - {pair[1]}, Correlation: {pair[2]}")
```

```
Pair: Crime_MOTOR_VEHICLE_THEFT - Crime_CRIMINAL_DAMAGE, Correlation:
0.24962841130447888
Pair: Crime_DECEPTIVE_PRACTICE - Crime_THEFT, Correlation: 0.35668458424264426
Pair: Crime_THEFT - Crime_CRIMINAL_TRESPASS, Correlation: 0.23231920724746682
Pair: Crime_BATTERY - Crime_CRIMINAL_DAMAGE, Correlation: 0.2531180711829242
Pair: Crime_BATTERY - Crime_ASSAULT, Correlation: 0.31328139117196047
Pair: Crime_BATTERY - Crime_WEAPONS_VIOLATION, Correlation: 0.27826439102933564
Pair: Crime_CRIMINAL_DAMAGE - Crime_ASSAULT, Correlation: 0.2336953556787837
Pair: Crime_OTHER_OFFENSE - Crime_ASSAULT, Correlation: 0.2020758494835316
Pair: Crime_ASSAULT - Crime_WEAPONS_VIOLATION, Correlation: 0.23104828841621122
```

[26]:

```
# Find the most correlated crime types
top_correlations = correlation_matrix.unstack().sort_values(ascending=False)
top_correlations = top_correlations[(top_correlations > 0.2)]
```

1.2.2 2.2 Predicting Theft

To create a predictive model that predicts one type of crime (e.g., theft) using all other crime types and location descriptions as features, you can follow these steps:

Let's have only the theft data. We predict by using its past 3 days within the district. Then we use more complex data to predict the next day.

[30]:

```
# Load the aggregated data
file_path = '/Users/YigitAydede/Library/CloudStorage/Dropbox/Documents/Courses/
 ↪MBAN/NLPBootcamp/PythonBC/aggregated_crime_data.csv'
aggregated_df = pd.read_csv(file_path, parse_dates=['Date'])

# Ensure the data is sorted by date for each district
aggregated_df.sort_values(by=['District', 'Date'], inplace=True)

# Inspect the data
print(aggregated_df.head())
```

```
   District       Date  Total_Crime  Total_Arrest  Crime_MOTOR_VEHICLE_THEFT  \
0         1 2024-01-01           34             9                          4
1         1 2024-01-02           29            11                          2
2         1 2024-01-03           39             3                          3
3         1 2024-01-04           34             7                          0
4         1 2024-01-05           41             8                          0

   Crime_DECEPTIVE_PRACTICE  Crime_THEFT  Crime_BATTERY  Crime_SEX_OFFENSE  \
0                         1            8             12                  0
1                         2            9              5                  0
2                         6           18              3                  0
3                         4           11              7                  0
4                         7           15              4                  0
```

```
   Crime_CRIMINAL_DAMAGE  …  Location_CREDIT_UNION  \
0                      0  …                      0
1                      2  …                      0
2                      1  …                      0
3                      1  …                      0
4                      1  …                      0

   Location_VEHICLE_-_COMMERCIAL:_TROLLEY_BUS  Location_HALLWAY  \
0                                           0                 0
1                                           0                 0
2                                           0                 0
3                                           0                 0
4                                           0                 0

   Location_GARAGE  Location_OFFICE  Location_RETAIL_STORE  \
0                0                0                      0
1                0                0                      0
2                0                0                      0
3                0                0                      0
4                0                0                      0

   Location_LIQUOR_STORE  Location_CHA_HALLWAY  Location_CTA_"L"_TRAIN  \
0                      0                     0                       0
1                      0                     0                       0
2                      0                     0                       0
3                      0                     0                       0
4                      0                     0                       0

   Location_CTA_TRACKS_-_RIGHT_OF_WAY
0                                   0
1                                   0
2                                   0
3                                   1
4                                   0

[5 rows x 158 columns]
```

Create lag features

[31]:

```python
def create_lag_features(df, target, lags, groupby_col):
    for lag in range(1, lags + 1):
        df[f'{target}_lag_{lag}'] = df.groupby(groupby_col)[target].shift(lag)
    return df

# Define the number of lag days
num_lags = 3
```

```
# Create lag features for the 'Crime_THEFT' column
aggregated_df = create_lag_features(aggregated_df, 'Crime_THEFT', num_lags,␣
 ↪'District')

# Drop rows with NaN values that result from lag creation
aggregated_df.dropna(inplace=True)

# Inspect the data with lag features
print(aggregated_df.head())
```

```
   District       Date  Total_Crime  Total_Arrest  Crime_MOTOR_VEHICLE_THEFT  \
3         1 2024-01-04           34             7                          0
4         1 2024-01-05           41             8                          0
5         1 2024-01-06           30             6                          2
6         1 2024-01-07           22             5                          3
7         1 2024-01-08           34             8                          3

   Crime_DECEPTIVE_PRACTICE  Crime_THEFT  Crime_BATTERY  Crime_SEX_OFFENSE  \
3                         4           11              7                  0
4                         7           15              4                  0
5                         4            9              2                  0
6                         3            7              2                  0
7                         1           10              4                  0

   Crime_CRIMINAL_DAMAGE  …  Location_GARAGE  Location_OFFICE  \
3                      1  …                0                0
4                      1  …                0                0
5                      5  …                0                0
6                      4  …                0                0
7                      1  …                0                0

   Location_RETAIL_STORE  Location_LIQUOR_STORE  Location_CHA_HALLWAY  \
3                      0                      0                     0
4                      0                      0                     0
5                      0                      0                     0
6                      0                      0                     0
7                      0                      0                     0

   Location_CTA_"L"_TRAIN  Location_CTA_TRACKS_-_RIGHT_OF_WAY  \
3                       0                                   1
4                       0                                   0
5                       0                                   0
6                       0                                   0
7                       0                                   0

   Crime_THEFT_lag_1  Crime_THEFT_lag_2  Crime_THEFT_lag_3
3               18.0                9.0                8.0
4               11.0               18.0                9.0
```

```
5                15.0               11.0               18.0
6                 9.0               15.0               11.0
7                 7.0                9.0               15.0

[5 rows x 161 columns]
```

[32]:

```python
# Define the target variable
target = 'Crime_THEFT'

# Define the feature set by excluding the target variable and other non-feature␣
 ↪columns
feature_cols = [f'Crime_THEFT_lag_{i}' for i in range(1, num_lags + 1)]

# Select the features and target from the DataFrame
X = aggregated_df[feature_cols]
y = aggregated_df[target]
```

Split the data into training and testing sets

[33]:

```python
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

A Simple linear regression model

[34]:

```python
from sklearn.linear_model import LinearRegression

# Initialize the model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

[34]:

```
LinearRegression()
```

Evaluate the model

[35]:

```python
from sklearn.metrics import mean_squared_error, r2_score

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R^2 Score: {r2}")
```

```
Mean Squared Error: 11.396217555270395
R^2 Score: 0.46141715844335973
```

Make predictions

[37]:

```python
# Make predictions on new data (e.g., the test set)
predictions = model.predict(X_test)

# Create a DataFrame with the actual and predicted values along with the dates and districts
result_df = X_test.copy()
result_df['Actual_Crime_THEFT'] = y_test
result_df['Predicted_Crime_THEFT'] = predictions
result_df['District'] = aggregated_df.loc[X_test.index, 'District']
result_df['Date'] = aggregated_df.loc[X_test.index, 'Date']

# Display the results
print(result_df.head())
```

```
      Crime_THEFT_lag_1  Crime_THEFT_lag_2  Crime_THEFT_lag_3  \
2438                9.0               14.0                7.0
1940                2.0                4.0                3.0
975                 7.0                2.0                0.0
2384                9.0               10.0               13.0
4147                5.0                4.0                4.0

      Actual_Crime_THEFT  Predicted_Crime_THEFT  District       Date
2438                   8               9.370871        12 2024-05-22
1940                   7               3.808013        10 2024-03-02
975                    6               4.077546         5 2024-05-19
2384                  14               9.867449        12 2024-03-29
4147                  13               4.958095        22 2024-06-30
```