

Analyzing Patterns and Outcomes of Crimes and Arrests in Urban Areas of Los Angeles in USA

Mohammad Robaitul Islam Bhuiyan (23277555)

1 Introduction

This project leverages two comprehensive datasets of crime and arrest reports to explore patterns and correlations within urban crime and law enforcement activities in Los Angeles from 2020 to present. The project intends to optimize policing tactics and improve public safety measures by applying data-driven insights. The analysis aims to ascertain if aggressive law enforcement actions are linked to noticeable drops in crime by comparing the number and kind of arrests with the crime rates in particular neighborhoods. This analysis is crucial for city planners, police departments, and community leaders striving to allocate resources effectively and improve safety in urban environments.

2 Main Questions

- Does the frequency and type of arrests correlate with reductions in crime rates in specific neighborhoods over time?
- Are there identifiable patterns in victim demographics (age, sex, descent) for specific types of crimes?

3 Data Sources

Both the dataset reflects the crime and arrest incidents in the City of Los Angeles from 2020 to present respectively. This datasets are transcribed from original arrest reports that are typed on paper and therefore there may be some inaccuracies within the data. Some location fields with missing data are noted as (0°, 0°). Address fields are only provided to the nearest hundred block in order to maintain privacy.

- **Dataset-1:**
Data Source: The Home of the U.S. Government's Open Data
Metadata URL:Crime Data from 2020 to present
Data URL:Download CSV Dataset

Data Type: CSV

License and Permission: Licensed under the CC0 1.0 Universal Public Domain Dedication, allowing unrestricted use, modification, and distribution.

- **Dataset-2:**

Data Source:The Home of the U.S. Government's Open Data

Metadata URL:Arrest Data from 2020 to present

Data URL:Download CSV Dataset

Data Type: CSV

License and Permission: Licensed under the CC0 1.0 Universal Public Domain Dedication, allowing unrestricted use, modification, and distribution.

4 Data Pipeline Workflow

4.1 Extraction

The pipeline starts with the ingestion of crime data from a public open data portal provided by the city's police department. The data is loaded into a pandas DataFrame using the `read_csv` function, which directly reads CSV files into a DataFrame, handling different encodings and large files efficiently.

4.2 Data Cleaning

- Includes handling missing values, data type conversions, and ensuring data quality for analysis.: The pipeline checks for missing values in crucial columns such as 'Crm Cd Desc' (Crime Description) and 'Date of Crime' from crime dataset
- Converting data types where necessary, for example, ensuring that dates are parsed into date-time objects to facilitate time-based analysis.

4.3 Data Transformation

Crime descriptions are categorized into broader groups to facilitate analysis. This is done using a pre-defined mapping dictionary that groups similar types of crimes under a single category label. Standardizing data formats, especially for geographical coordinates and time, to ensure consistency across the dataset.

4.4 Output Preparation

Processed data is stored in SQLite for querying and the choice of CSV and SQLite as the output format is driven by its wide acceptance and ease of use in almost all data handling and analysis tools

5 Encountered Problems and Solutions

5.1 Missing and Inconsistent Data

Problem: The raw data may have missing values or inconsistencies, particularly in critical fields like crime descriptions and dates.

Solution: The `preprocess_crime` and `preprocess_arrest` functions handle missing data by dropping rows where essential data is missing (using the `dropna()` function) and ensuring date fields are correctly parsed (using the `pd.to_datetime()` function). This ensures that analyses are based on complete and accurately parsed entries.

5.2 Complexity in Merging DataFrames

Problem: Merging large datasets based on multiple columns can be computationally intensive and may lead to errors if key columns don't match perfectly.

Solution: The merge operation is conducted on common fields (date of crime, area name, crime description), which must be consistently formatted and indexed for successful joins. Proper preprocessing and ensuring that the merge keys are in the same format and data type are crucial.

5.3 Error Handling in Database Operations

Problem: Database operations may fail due to reasons like schema conflicts, type mismatches, or connection issues.

Solution: Adding robust error handling around database operations, including connection attempts, data insertion, and commit operations. Using context

managers (with statement) for managing database connections can ensure that resources are properly freed even if errors occur.

5.4 Large number of crime descriptions

Problem: Crime and arrest data comes with detailed and varied descriptions of incidents. These varied types of descriptions is a problems to analyze effectively and specifically on a large scale.

Solution: Categorizing crimes simplifies the data by grouping similar incidents into broader categories, making it easier to identify patterns and trends. This simplification helps analysts, policymakers, and the public to understand the data more clearly.

6 Meta-Quality Measures

- **Enhanced Error Handling in Data Loading:** Implement error handling when reading CSV files to manage exceptions related to file access issues or data corruption.

```
def load_datasets():
    try:
        crime_data = pd.read_csv("https://data.lacity.org/api/views/zhrs-ntv8/rows.csv?accessType=DOWNLOAD")
        arrest_data = pd.read_csv("https://data.lacity.org/api/views/amf-fr72/rows.csv?accessType=DOWNLOAD")
    except pd.errors.EmptyDataError:
        print("Error: No data found in the source.")
        return None, None
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return None, None
    return crime_data, arrest_data
```

Figure 1: Snippet of Data Loading Error Handling Code

- **Validation Checks Post-Data Loading:** After loading the data, add checks to confirm that essential columns are present and have the correct data types.

```
def validate_data(df, required_columns):
    missing_columns = [col for col in required_columns if col not in df.columns]
    if missing_columns:
        logging.error(f"Missing columns: {', '.join(missing_columns)}")
        raise ValueError(f"Dataframe is missing columns: {', '.join(missing_columns)}")
    return df
```

Figure 2: Snippet of Post-data validation check

- **Dynamic Handling of Schema Changes:** Use a flexible schema approach, especially when merging datasets, to ensure compatibility even if some columns are renamed or reformatted.
- **Logging:** Integrate logging throughout the pipeline to track the flow and pinpoint issues quickly.

By integrating these meta-quality measures, data pipeline is not only be more robust and error-tolerant

```
C:\Users\hp\AppData\Local\Temp\ipykernel_22208\36888693.py:46: UserWarning: Could not infer format, so each element will be
  e parsed individually. Falling back to 'dateutil'. To ensure parsing is consistent and as-expected, please specify a format.
  arrest_data['arrest_date'] = pd.to_datetime(arrest_data['arrest_date'])
2024-11-27 19:35:57.880 - INFO - arrest data preprocessing completed.
2024-11-27 19:35:08.731 - INFO - dataframes successfully merged.
2024-11-27 19:37:18.622 - INFO - data successfully saved to SQLite database.
```

Figure 3: Output of Logging at different steps of the program

but also adaptable to changes in data sources, ensuring long-term reliability and usefulness of the analysis outputs.

7 Result and Limitation:

7.1 Description of Output Data

The output of the data pipeline is saved into CSV and DB file. DB file contains two separate table named **arrest_data** and **crime_data** containing arrest and crime related information. The CSV contains the merged dataset that combines detailed crime incident records with arrest records based on common attributes like occurrence dates, area names and description. This merged dataset includes key information such as the type of crime, date and location of occurrence, arrest details, demographic data of suspects, and other pertinent details that were filtered and processed during the pipeline stages.

7.2 Data Structure and Quality

Structure: The final output structure of the CSV file dataset includes columns from both the crime and arrest data.

	DATE OCC	AREA NAME	Crn Cd Desc	Vict Age	Vict Sex	Vict Descent	Arrest Area Name	Arrest Date	Charge Group Description	Age	Sex Code	Descent Code
0	2020-03-01	Wiltshire	VEHICLE - STOLEN	0	M	O	Wiltshire	2020-03-01	Narcotic Drug Laws	19	M	B
1	2020-03-01	Wiltshire	VEHICLE - STOLEN	0	M	O	Wiltshire	2020-03-01	Vehicle Theft	38	M	H
2	2020-03-01	Wiltshire	VEHICLE - STOLEN	0	M	O	Wiltshire	2020-03-01	Miscellaneous Other Violations	29	M	W
3	2020-03-01	Wiltshire	VEHICLE - STOLEN	0	M	O	Wiltshire	2020-03-01	Fugitive/Courtesy	31	M	H
4	2020-03-01	Wiltshire	VEHICLE - STOLEN	0	M	O	Wiltshire	2020-03-01	Aggravated Assault	29	F	B

Figure 4: Final Merged Dataset

The DB file contains two separate table to store the crime and arrest information separately. The **crime_data** table has 623104 entries and **arrest_data** has 251907 entries and both of the table has no-null values.

	Date of Crime	AREA NAME	Crn Cd Desc	Vict Age	Vict Sex	Vict Descent
0	2020-03-01	Wiltshire	Vehicle Theft	0	M	O
1	2020-02-08	Central	Burglary	47	M	O
2	2020-11-04	Southwest	Vehicle Theft	19	X	X
3	2020-03-10	Van Nuys	Larceny	19	M	O
4	2020-03-01	Hollenbeck	Burglary	27	M	W

Figure 5: Table containing crime information

Quality: The pipeline ensures accuracy by pre-processing data, handling missing values, and stan-

dardizing entries. The categorization of crime types and the standardization of date formats enhance the consistency of the dataset.

7.3 Data Format

The final dataset is stored in two formats:

- **SQLite Format:** This format is chosen for robust data management, ease of querying, and scalability. It supports complex queries and can handle large datasets efficiently, which is suitable for dynamic data analysis and integration into applications. This format allow has to access the arrest and crime table separately as per our requirements.
- **CSV File:** Chosen for its simplicity and compatibility with various data analysis tools, making it accessible for analysts who might use different software. CSV files are easy to share and understand, and they do not require specialized database software to view or process.

7.4 Critical Reflection and Potential Issues

- The data pipeline successfully integrates and processes data from multiple sources, providing a unified view that can be used for comprehensive crime analysis. The categorization of crimes enhances the analytical value of the dataset by enabling aggregate analysis of crime types.
- Using an SQLite database for storage adds a layer of sophistication, allowing for more complex queries and potentially supporting real-time data analysis applications.

8 Conclusion

While the pipeline provides a solid foundation for crime and arrest data analysis, ongoing monitoring and updates are essential to accommodate changes in data sources and to address any new challenges that arise. Future enhancements could include implementing machine learning models to predict crime trends or developing interactive dashboards for real-time data visualization. These steps would further extend the utility of the pipeline and support more proactive crime prevention strategies.