Data Cleaning and Analysis Pipeline for the CRIMES dataset

Data and Information Quality

A.Y. 2024-2025

Project ID 83

Dataset 14

Roman Jules Tiédrez 10991814

Hadi 10946669

Ghulam Abbas Zafari 10944305

Table of Contents

1. Introduction

In this project, we aimed at constructing a robust data cleaning analysis pipeline tailored to preprocess, analyze, and evaluate the complex CRIMES dataset for deriving further insights. The CRIMES dataset, a repository of historical crime records for the city of Boston, MA, contains diverse features such as the nature of each crime, its locations, time, and other associated metadata, presenting a valuable opportunity for data-driven decision-making and policy formulation. However, its inherent challenges, including missing data, inconsistencies, and large size, required human-driven bespoke preprocessing methodologies.

This report outlines the entire pipeline we developed, emphasizing the systematic steps undertaken to clean and analyze the dataset, in that order. The pipeline integrates standard practices in data science, namely data wrangling and visualization, ensuring the replicability and transparency, and showing the reliability of our results, enabling the pipeline to adapt to similar datasets with minimal modifications.

Additionally, the methodologies chosen were guided by the goals of uncovering crime patterns, identifying correlations, and facilitating predictive modeling to assist in actionable decision-making. Examples include the treatment of missing values to preserve data integrity, and location-based imputations to ensure data quality. This report provides a detailed account of these methodologies, their justifications, and the outcomes achieved.

The notebook which accompanies this report is quite detailed and completely commented, and illustrated with extra analysis, more than the present document, which acts as a summary. Any reader pressed for time may find it more efficient to read through the notebook first.

Our implementation mainly uses the following Python libraries:

- *pandas*, *seaborn*, *numpy*, and *matplotlib* need no introduction
- *missingno* was used for initial missing values visualization
- *sklearn* for all machine learning-related code, from imputations to model selection and evaluation
- Since our dataset contains geospatial data, we used *geopandas*, *geopy*, and *folium* for coordinates manipulation and visualization
- *tensorflow* with *keras* for our Neural Network-based classification pipeline

---

2. Objectives

The key objectives of this project were as follows:

- **To design and implement a data preprocessing pipeline in view of a data analysis task**: The pipeline was built to handle missing values, outliers, and normalization issues efficiently, ensuring the dataset's readiness for further analysis and machine learning applications. Special attention was given to preserving data integrity, that is not resorting to direct suppression of problematic rows or naive imputations, and not introducing bias.
- **To provide actionable insights through an example of data analysis**: Here we performed multi-class linear classification on the cleaned dataset to show how the cleaned dataset can be used to refine knowledge. The data analysis pipeline was run on both the initial dataset and its cleaned version to highlight the usefulness of our cleaning pipeline.
- **Flexibility and Replicability**: The pipeline follows a clearly-defined outline so that its main principles and code can easily be understood and reused.

---

3. Data exploration

Here we summarize the main findings of the exploration phase. Please see the notebook for detailed figures and illustrations.

All in all, the 'messy' dataset is actually already pretty clean. It is likely that most of its data was entered automatically. Only the STREET column is guaranteed to have been manually typed, since it contains typos we will have to fix. Then, the dataset contains many geospatial attributes, which are pretty redundant (one can get a street name from its coordinates, for instance). However, these attributes contain many missing values. Our guess is that the automated process in charge of finding, for example, the name of a district from some coordinates might have failed at some points. But since these are redundant, we can fill in the gap ourselves with the aforementioned geospatial libraries.

Also, there is redundancy between the OCCURRED_ON_DATE column and other time attributes. We later show that that attribute can be dropped without loss of information.

Finally, the SHOOTING attribute contains only 'Y' and *NaN* data. We assume the extremely likely hypothesis that *NaN* here stands for the absence of shooting.

There are only a few duplicated rows, but that number might change after a round of cleaning. We will see that there are more duplicates than it seems, although their number will stay small enough for us to drop them without consequences. They are likely due to some system error or humans pressing some button twice instead of once.

To conclude, what will matter the most is the completeness of data: we need to fix missing values in the most correct way possible, that is deriving them from other attributes rather than wild guessing. Since the dataset is simply a log of police operations, it has little to no prominent commercial value, and its timeliness does not matter, though looking only at the most recent crimes might be more interesting for predicting future trends.

4. Pipeline implementation

Preprocessing is a necessary step in data analysis that ensures data integrity, enhances feature relevance, and supports analysis. Following the "garbage in - garbage out" principle, using a non-cleaned dataset would likely result in a terrible performance or wrong insights when analyzing. This section outlines the systematic steps undertaken to prepare the dataset for analysis, in the order they were performed.

4.1 Data Cleaning

4.1.1 Null values and outliers

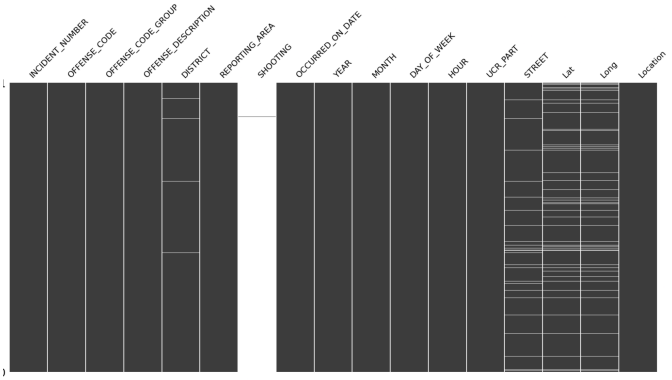

Figure 1: missing values visualization with *missingno*

At first glance, the dataset contains hundreds of thousands of missing values. In reality, the SHOOTING attribute uses a NULL value when no shooting occurs, so there are only about 50,000 missing values (about 1% of all values), all of which are geospatial values except UCR_PART.

A quick search on the FBI's website[1] showed that all missing UCR_PARTs should be 'Part One'.

Since they are geospatial values, a street can be obtained from coordinates, coordinates can be approximated from the street name, and so on. By iteratively using both geospatial libraries or the information in other columns, we managed to find approximations for almost all missing values. Concretely, we used STREET to approximate Lat

[1] https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/topic-pages/offense-definition

and Long, and in turn we used Lat and Long to find STREET. We also used STREET to guess REPORTING_AREA and DISTRICT. For instance, if we have the following rows:

…	DISTRICT	STREET
…	C6	COLUMBIA RD
…	*NaN*	COLUMBIA RD
…	C6	COLUMBIA RD

Table 1: imputation example

then we replace the *NaN* with “C6”.

Following these steps, there were still a few missing values, so we used a minibatch version of MICE for REPORTING_AREA, Lat and Long. Other algorithms such as random forest were experimented with, but the size of the dataset made them unusable with our limited RAM. MICE was more of a default choice for its ease of use here. We lastly ‘guessed’ the REPORTING_AREA from the STREET using the same techniques as before. After these steps, only a very small number of DISTRICTs were still missing, so we decided to fill them with the mode of the column.

4.1.1 Consistency

Some INCIDENT_NUMBERs are made up of two parts, so we split them. We also remove the leading ‘I’ they all share, e.g.

I030217815-08 -> 030217815,08

STREETs had a lot of typos that we had to identify and fix. Since STREETs contain streets, but also avenues, plazas, wharves, … we split it into the name of the location and its type, and for consistency, we replace abbreviations with what they stand for, e.g.

“COVENTRY ST” -> “COVENTRY”,”STREET”

4.1.2 Redundancy

Locations are fully identified by the Lat and Long columns, so we can drop the Location column. The exact same consideration applies to OCCURRED_ON_DATE, which perfectly matches YEAR, MONTH, DAY_OF_WEEK, and HOUR. We chose to keep YEAR, MONTH, DAY_OF_WEEK, and HOUR, and drop OCCURRED_ON_DATE, because we found them to be easier to use in our analysis pipeline. Depending on the usage, the opposite might be better. Many OFFENSE_DESCRIPTION could be split in two, three, or even four, but we do not have any reason to do so and it would just make the dataset larger and its exploitation harder since not all of them can be split and splitting would create a lot of empty fields.

4.1.2 Record linkage

Here we look at OFFENSE_GROUP, OFFENSE_DESCRIPTION, and INCIDENT_NUMBER, and find that these attributes contain no errors and do not need wrangling.

4.1.3 Final touches

Here we convert all attributes to the correct data type and remove duplicates. Removing duplicates needed no intense reflection since two crimes that occur at the exact same place at the exact same time have to be the same crime, and because the percentage of duplicates was really low. Note that INCIDENT_NUMBERs are almost unique, so they have to be ignored when looking for duplicates: most duplicates actually had the exact same values for all columns but INCIDENT_NUMBER. Less than 1% or rows were duplicates.

4.2 Preparation for classification

We perform multi-class classification using OFFENSE_CODE_GROUP as target. In natural language, we want to predict the crime category depending mainly on its location and time, hopefully highlighting patterns.

The dataset is now considered cleaned. But before running any machine learning algorithm on it, additional steps are needed. Note that this introduces some bias to the final comparison between performance on the initial and the cleaned dataset, since some amount of 'cleaning' on the messy dataset is still required for the algorithms to run properly.

Every step presented in the following paragraph is to be applied on both the initial dataset and the cleaned version.

4.2.1 Rare values removal

We remove rows that contain values that appear less than 5% of the time: these rows are only noise from the point of view of classification. 5% is a pretty standard value, and we do not have much domain knowledge or client specifications.

4.2.2 Anomaly detection

We use the isolation forest method to remove extreme latitudes and longitudes. Isolation forest is an excellent method when we have an envelope of inliers surrounded by outliers, which is exactly our case.

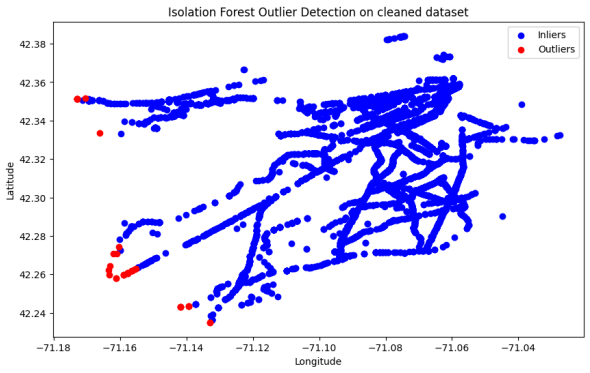

Figure 2: anomaly detection with isolation forest visualization on cleared dataset

4.2.3 Scaling, Encoding, …

We set the target variable and normalize the data to make it compatible with machine learning analysis. We use label encoding on categorical variables for feature selection because one-hot encoding would make the dataset too large for that, but then revert back to one-hot encoding to not introduce any bias.

4.2.4 Feature selection

We use the random forest technique to rank features according to their importance. We drop the "most useless" features, primarily for faster computing at a negligible cost in performance. Random forest is quite flexible and provides a ranking of attributes so we can simply choose a cut-off value and remove the least useful ones, and we actually know which ones we are removing. Additionally, it takes the possible relationships between features into account.

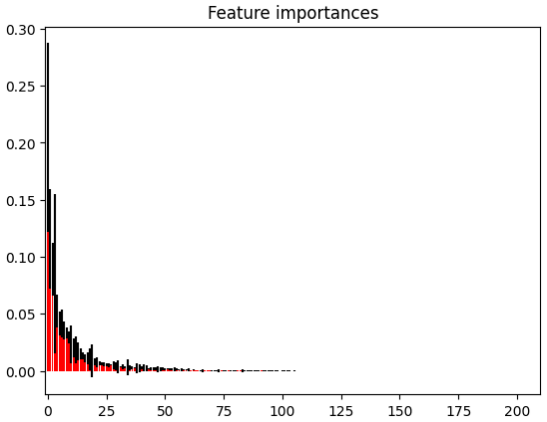

Figure 3: feature importance with random forest visualization on cleaned dataset

5. Results

We use a pretty basic feed-forward neural network for multi-class classification. Here the focus was not on the machine learning technique but on comparing the performances of the exact same algorithm on the initial dataset versus its cleaned version.

Model	Messy	Trained
Test accuracy	0.84	0.91
Precision (avg)	0.77	0.86
Recall (avg)	0.84	0.91
F1 score (avg)	0.78	0.88

Table 2: average performance metrics after five runs of the full pipeline. Confidence intervals have been omitted.

The metrics are not too bad for the messy dataset. The necessary cleaning we did as preparation for the model is likely involved. We observed that our model was particularly bad at correctly classifying drug violations.

Our simple neural network is pretty correct at predicting OFFENSE_CODE_GROUP on the cleaned dataset. And as expected, the performance is unequivocally significantly better than on the messy dataset, which proves our cleaning was useful. Unsurprisingly, our model has a harder time discriminating between general larceny and larceny from motor vehicle, and between located missing people and reported missing people.

Note that we use OFFENSE_CODE and OFFENSE_DESCRIPTION in our analysis, which might make classification a bit too easy to be realistic.

---

6. Challenges and Future Directions

Since our resources were limited, the dataset was too large for some techniques to be applied, and we also had to perform analysis on a smaller subset of the dataset. With professional-grade infrastructures we could lift these restrictions, on top of making the whole pipeline faster. We also could have used expensive cross-validation, which is standard practice, to enhance our analysis pipeline. Nonetheless, with more experience, we could have used modern caching algorithms on strategically-defined batches of data to bypass RAM limitations. Additionally, smarter techniques exist for geospatial data, but are dependent on the reliability of both one's internet bandwidth and the server that provides the data, and can be extremely slow. In a professional setting, we could have requested an API key to use an industry-grade geolocator with Google's API, for instance.

---

8. Conclusions

This project successfully achieved its objectives by developing a comprehensive data analysis pipeline for preprocessing, analyzing, and deriving insights from the dataset. The pipeline incorporated various techniques from data cleaning to model evaluation, ensuring high flexibility even on a large and complex dataset.

The most important point of the project, which we successfully observed, was to show that cleaning indeed made analysis more efficient.