Data Mining Project Report

Master Degree in Computer Science, University of Pisa
Prof. Anna Monreale
Academic Year 2023-2024

Giacomo Aru
g.aru@studenti.unipi.it
Roll number: 597700

Giulia Ghisolfi
g.ghisolfi@studenti.unipi.it
Roll number: 664222

Luca Marini
l.marini11@studenti.unipi.it
Roll number: 578543

Irene Testa
i.testa@studenti.unipi.it
Roll number: 582061

1 Data Understanding and Preparation

1.1 Gun Incidents Dataset

We noticed that the dataset 'incidents.csv' contained 1% duplicate entries. Given that the columns, particularly those related to attributes such as latitude, longitude, date, and incident characteristics, should uniquely identify each record, we removed the duplicate entries. The percentage of records with at least a 'NaN' value amount to 78.31%. Additionally, an exploratory analysis, facilitated by visualizations accessible on our website, revealed numerous syntactic and semantic inconsistencies. In the following sections, we describe how we handled each attribute.

1.1.1 Date attribute

We observed that 9.6% of the incidents have out-of-range values for the attribute date (i.e., the year is equal to 2028, 2029, or 2030). To address this inconsistency, we initially examined the dataset for duplicates with valid dates, but found none. Assuming that dates were manually entered using a numeric keypad and errors could be typos (e.g., 2030 instead of 2020), we attempted correction by subtracting 10 or 11 from the erroneous year values. Additionally, we explored replacing the incorrect values with the mean or median values. Unfortunately, all of these corrections significantly altered the distribution. We also visualized the number of incidents per day over the years to explore alternative correction methods, but no significant insights were gained. Therefore, we opted not to use them for data correction. Instead, we divided the date attribute into day, month, and year attributes, setting values out of range for the year attribute to 'NaN'. This allowed us to consider the day and month attributes as correct, including records in the analysis where the year attribute is not taken into account. Further analysis revealed that months with incident counts below the 25^{th} percentile are April, June, and November, while those above the 75^{th} percentile are January, March, and August. Thursdays and Fridays are the days of the week with incident counts below the 25^{th} percentile, whereas Saturdays and Sundays have counts above the 75^{th} percentile. Exploring the federal holidays calendar in the USA, we found that New Year's Eve has the highest number of incidents, followed by Independence Day. Christmas and Thanksgiving Day are the holidays with the fewest incidents.

1.1.2 Geospatial attributes

Correction of the attributes Plotting latitude and longitude on a map, we discovered some points in China that are clearly incorrect. Furthermore, we observed that, given a certain value for the attributes `latitude` and `longitude`, the attribute `city_or_county` does not always have the same value. Sometimes, the attribute `city_or_county` takes on the value of the city, while at other times, it takes on the value of the county. Additionally, even when the attribute refers to the same county, it may be written in different ways (e.g., Key West or Key West (Stock Island)). The `address` attribute may also be written in various forms (e.g., "Avenue" written as "Ave," or "Highway" as "Hwy").

To address these inconsistencies, we used the `GeoPy` library[1]. This library allowed us to retrieve the actual address corresponding to a given latitude and longitude. We queried the library using all the latitudes and longitudes of the points in the dataset and used the results to correct inconsistencies. Among all the attributes returned by `GeoPy`, we selected and used the following: (1) `importance`, a numerical value $\in [0, 1]$, indicating the importance of the location compared to other locations; (2) `address_type` (e.g., "house," "street," "postcode"); (3) state (referred to as `state_geopy`); (4) county (`county_geopy`); (5) suburb (`suburb_geopy`); (6) city (`city_geopy`); (7) town (`town_geopy`); (8) village (`village_geopy`); (9) address (`display_name`), a user-friendly representation of the location, often formatted as a complete address. Additionally, we downloaded the list of counties (or their equivalents) in each state from Wikipedia[2]. This dataset served as a fallback in scenarios where incident data didn't align with `GeoPy` information or when latitude and longitude were unavailable, allowing us to check whether the county actually belonged to the state.

To correct inconsistencies, we followed these steps:

[1] https://geopy.readthedocs.io/en/stable/

[2] https://en.wikipedia.org/wiki/County_(United_States)

(1) we converted to lowercase the attributes `state`, `county`, and `city`; (2) if `city_or_county` contained values for both city and county, we split them into two different fields; (3) we removed from `city_or_county` the words 'city of' and 'county'; (4) We removed punctuation and numerical values; (5) We removed frequent words from `address` and `display_name`. Then, when `latitude` and `longitude` were available, and `GeoPy` provided information for the corresponding location, we checked for equality between `state` and `state_geopy`, between `county` and either `county_geopy` or `suburb_geopy`, and between `city` and either `city_geopy`, or `town_geopy`, or `village_geopy`. If these comparisons failed, we checked for potential typos in the string using the Damerau-Levenshtein distance implementation from the `jellyfish` library[3]. This involved setting specific thresholds to determine the maximum allowable distance for two strings to be considered equivalent. Distinct thresholds were applied for `state` and `city_or_county`. If the comparison still failed, we compared the `address` field from our dataset with `GeoPy`'s `display_name`, identifying matches through the utilization of the Damerau-Levenshtein distance. In case of a match with `GeoPy` data, we set the values for the fields `state`, `county`, `city`, `latitude`, `longitude`, `importance`, and `address_type` to the corresponding fields provided by `GeoPy`. Otherwise, we checked for matches with the Wikipedia data using again the Damerau-Levenshtein distance.

We observed that in the dataset provided in the file 'year_state_district_house.csv', congressional districts in states with a single congressional district are labeled with '0'. However, this convention was not consistent in the 'incidents.csv' dataset. To address this discrepancy, we corrected these instances in the 'incidents.csv' dataset by setting the corresponding entries to '0'. Additionally, the count of distinct congressional districts in each state exhibited inconsistency. To address this, we assigned 'NaN' to the values of non-consistent congressional districts in the 'incidents.csv' dataset. Refer to the following paragraphs for a detailed description of how missing values were inferred. Moreover, for a given pair of values in the latitude and longitude attributes, the `congressional_district` attribute did not consistently maintain the same value. To rectify this, we adjusted the entries by setting the most frequently occurring value for the `congressional_district` attribute.

Inconsistencies were also identified in the attributes `state_house_district` and `state_senate_district`. However, as these attributes were not utilized in further analysis, they were left unaltered.

Inference of the attribute city To fill in missing values for the attribute `city` when `latitude` and `longitude` were available, we initially computed 'city centroids' by averaging the latitude and longitude for all the points belonging to each city. Subsequently, we recovered the missing cities by searching for the closest city centroid. If the distance fell within the 75^{th} percentile of the distances between all the points assigned to that centroid and the centroid itself, we assigned the corresponding city. Distances between centroids and incident locations were computed using the the geodesic distance implementation from the `GeoPy`[4] library. Through this process, we inferred 2249 missing values. For instance, Figure 1 zooms in on the USA map, focusing on Orange County in Florida. Colored points represent those whose city we were able to infer (e.g., incidents in blue were assigned to the city of Orlando, incidents in green to the city of Alafaya, incidents in light blue to the city of Lockhart). Black points denote those whose city was left as 'NaN' even after the inference process.

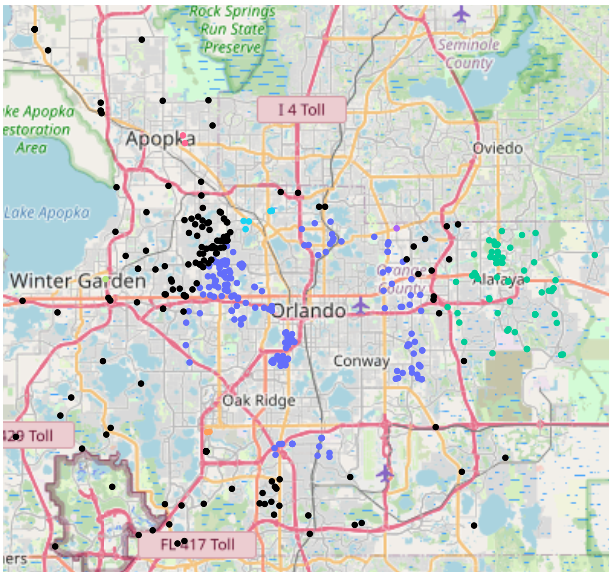


Figure 1: Incidents in Orange County in Florida. Colored dots represents incidents whose city was inferred. Black points denote those whose city was left as 'NaN' even after the inference process.

Inference of the attribute congressional district As the attribute `congressional_district` is essential for merging the 'incidents.csv' dataset with the 'year_state_district_house.csv' dataset, and considering that congressional district borders are delineated to represent equivalent population sizes, making them valuable for computing local indicators, we found it necessary to infer missing values for this attribute. Initially, we attempted to infer values by examining whether, given a `state`, `city`, and `state_senate_district`, or given a `state`, `city`, and `state_house_district`, the `congressional_district` attribute consistently

[3] https://jamesturk.github.io/jellyfish/

[4] https://geopy.readthedocs.io/en/stable/#module-geopy.distance

held the same value. However, this approach did not yield consistent results. Consequently, we pursued an alternative strategy. Plotting the incident locations on a map revealed that many points with missing `congressional_district` values were often surrounded by points belonging to the same congressional district. Since every possible congressional district in a state had at least one incident, we employed the K Nearest Neighbor Classifier with $K = 1$ and geodesic distance as distance metric, to label the congressional district for points with missing values. This approach allowed us to recover all missing values for the `congressional_district` attribute for points with non-null latitude and longitude, amounting to 2893 values. As an illustration of the inference results, Figure 2b depicts incident locations in the state of Alabama, color-coded according to the values of the congressional district after the application of K Nearest Neighbors. Furthermore, Figures 2c and 2d provide a comparison between the decision boundaries of the classifier and the actual boundaries of congressional districts.

1.1.3 Attributes about the characteristics of the participants in the incident

We identified numerous syntactic and semantic inconsistencies in the attributes related to the characteristics of participants in the incidents, encompassing: (1) instances of strings in numerical fields; (2) negative values for counts or ages; (3) ages exceeding 150 years; (4) minimum ages surpassing the maximum or average age; (5) incongruent values between the attributes pertaining to the minimum, maximum, and average age of the participants and the number of children, teens, and adults; (6) situations where the total number of participants was less than the number of individuals within a subcategory. Consequently, we initially replaced inconsistent values with 'NaN'. Subsequently, we endeavored to infer missing values. In cases where the number of males and females was available, we set the total number of participants as the sum of the two (this recovered 2763 missing values). When data about a random participant of the incident was consistent, and the number of participants equaled 1, we utilized that data to set the attributes related to age and gender (this recovered x missing values).

1.1.4 Attributes about the characteristics of the incident

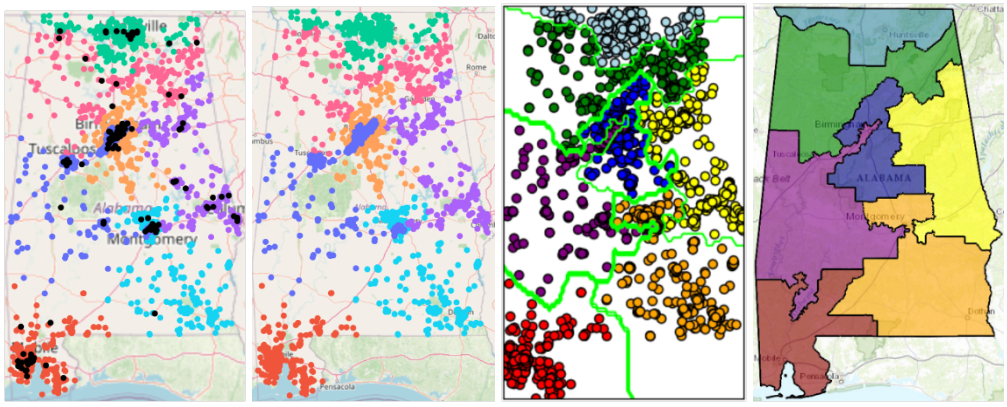
The dataset contains 52 unique values for the attribute `incidents_characteristics1`, with the most frequent being "Shot - Wounded/Injured" (indicating that, at the time of data collection, the possible values for this attribute were likely limited to a predefined set). There are 90 unique values for the attribute `incidents_characteristics2`, with the most frequent being "Officer Involved Incident" (suggesting again that, at the time of data collection, the possible values for this attribute were likely limited to a predefined set). Among entries with non-NaN values for `incident_characteristics1`, 59.3% have 'NaN' values for `incidents_characteristics2`. Additionally, when both `incidents_characteristics1` and `incidents_characteristics2` are non-NaN, they always have different values. The most frequent value for the attribute notes is "man shot," but there are 136,652 unique values for this attribute (indicating that, at the time of data collection, the potential values for this attribute were not well-defined). We displayed the most frequent values assumed by this attribute in a Word Cloud. Such visualization is available in our website.

To leverage the information in the `incidents_characteristics1` and `incidents_characteristics2` attributes for further analysis, we created boolean variables to categorize the characteristics of each incident, such as: (1) `firearm` (True if firearms were involved in the incident); (2) `shots` (True if the incident involved shots); (3) `air_gun` (True if air guns were involved in the incident); (4) `aggression` (True if the incident involved aggression, with or without a gun); (5) `defensive` (True if a gun was used for defensive purposes); (6) `suicide` (True if the incident was suicidal, including attempted suicides); (7) `unintentional` (True if the incident was accidental); (8) `abduction` (True if the incident involved any form of abduction); (9) `injuries` (True if one or more subjects got injured); (10) `death` (True if one or more subjects died); (11) `illegal_holding` (True if the incident involved a stealing act or if a gun was illegally possessed); (12) `drugs` (True if the incident involved drugs); (13) `officers` (True if one or more officers were involved in the incident); (14) `organized` (True if the incident was planned by an organization or a group); (15) `social_reasons` (True if the incident involved social discriminations or terrorism); (16) `road` (True if the incident happened in road context); (17) `house` (True if the incident happened in a house); (18) `school` (True if the incident happened next to a school); (19) `children` (True if the incident involved one or more children); (20) `workplace` (True if the incident happened in a workplace). In instances where there was insufficient information to determine whether a specific characteristic was present in an incident, we opted to default the corresponding variables to False.

Subsequently, we computed statistics regarding the values of these variables.In fatal incidents, murders account for 22.4%, suicides for 8%, defensive incidents for 1.3%, accidental incidents for 2.2%, and for 66.1%, characteristics are unknown or belong to other categories. Incidents characteristics for events involving women exhibit a distribution similar to those involving males only, with the main differences being a higher frequency of suicides and a lower involvement of officers.

1.2 Poverty Dataset

Verifying whether the tuple `<state, year>` uniquely identified each row in the dataset provided in the file



(a) (b) (c) (d)

Figure 2: (a) Incidents' locations in Alabama. Points are colored according to the congressional district values before the inference. Incidents with missing congressional district are indicates in black. (b) Incidents' locations in Alabama after the inference. (c) Decision boundaries of the K Nearest Neighbor model. (d) Actual boundaries of congressional districts.

'poverty_by_state_year.csv', as it should, revealed an inconsistency. The state of Wyoming exhibited two entries for the year 2009 and lacked an entry for 2010. Leveraging the fact that the CSV file is ordered by year and state, we corrected one of the entries by replacing its year. Additionally, we noted the absence of data for the year 2012. To address this gap, we calculated the average of the `povertyPercentage` values for the preceding and succeeding years. Apart from these adjustments, no other data was missing or exhibited anomalies. Each state (including the District of Columbia) and year combination had a corresponding record. Notably, New Hampshire exhibited a significantly lower average poverty percentage compared to other states, while Mississippi's average poverty percentage was notably higher. These trends are visually represented in the interactive line plot and animations available on our website.

1.3 Elections Dataset

In the dataset provided in the file 'year_state_district_house.csv,' the triple <`year`, `state`, `congressional_district`> uniquely identifies each row. In each record the values of the variable `candidateVotes` are always less than or equal to the values of the variable `totalVotes`, and all states (including the District of Columbia) are present. However, several corrections were necessary to ensure data integrity.

For the District of Columbia, which only provided data from 2020, missing values were obtained from Wikipedia. While the number of votes received by the winning party for the 2020 elections aligned with the information from Wikipedia, the number of `totalVotes` differed. To maintain consistency with the other data, we replaced the `totalVotes` value from 2020 with the one obtained from Wikipedia[5].

An upper outlier in `candidateVotes` in Maine was identified and corrected by consulting again Wikipedia[6]. Some records, such as certain congressional districts in Florida in 2014, showed values of 0, 1, or -1 in the fields `candidateVotes` and `totalVotes`. Further investigation on Wikipedia[7] revealed that in such cases, no candidates filed to challenge the incumbent representative, so we copied `candidateVotes` and `totalVotes` values from the previous year.

To facilitate later analysis, the party 'DEMOCRATIC FARMER LABOR' was replaced with 'DEMOCRATIC' since it is the affiliate of the Democratic Party in the state of Minnesota[8] .

In some districts, the winning party obtained 100% of the votes, and this information aligned with Wikipedia, which reported the same percentage in cases where there was no opponent party. In certain entries, the winning party obtained less than 50% of the votes. A search on Wikipedia[9] revealed that the number of `candidateVotes` refers to the votes obtained by the winner at the final runoff, while the number of `totalVotes` refers to the voters at the runoff plus the votes for the other candidates at the primary election. Since these cases were few and the vote counts were not utilized in further analysis, no corrections were made for these errors.

[5] https://en.wikipedia.org/wiki/2020_United_States_House_of_Representatives_election_in_the_District_of_Columbia

[6] https://en.wikipedia.org/wiki/2022_United_States_House_of_Representatives_elections_in_Maine

[7] https://en.wikipedia.org/wiki/2014_United_States_House_of_Representatives_elections_in_Florida

[8] https://en.wikipedia.org/wiki/Minnesota_Democratic-Farmer-Labor_Party

[9] https://en.wikipedia.org/wiki/2016_United_States_House_of_Representatives_elections_in_Louisiana

1.3.1 Analysis of the cleaned datasets

After the cleaning, the attributes no longer show outliers due to erroneous data. Upon examining the linear correlation of the attributes, we observed a negative correlation between `povertyPercentage` and `latitude`, consistent with the fact that states in the southern United States tend to be poorer. The other attributes with higher positive or negative correlations align with their expected semantics.

Another interesting finding is that the number of incidents per 100k inhabitants[10] in the District of Columbia is significantly higher than in other states, especially in January 2014. To investigate this, we checked for possible duplicates and explored the characteristics of the incidents to understand if it was due to different policies in data collection. However, we could not find a definitive explanation. Since the number of incidents in 2013 in the District of Columbia is only 5, this anomaly may be due to the fact that data collection started in 2014, and some incidents that occurred before 2014 were incorrectly registered as happening in January 2014 (note that, however, the day of the month assumes different values). We also compared these data with reports provided by the government[11] but could not find the reason. Furthermore, even in other months and years, the proportion of incidents relative to the population of the state is higher than in other states.

Apart from this, we conducted various types of visualizations to explore the data (please refer to the Python notebooks on our website). For example, Figure 3 illustrates the number of incidents per 100K in the US states for each month of the years (District of Columbia excluded). In 2013, the number of incidents is lower in all states. The highest number of incidents occurred in August 2014 in Delaware, and in June and July in Wyoming. States with the highest number of incidents include Alaska, Delaware, Illinois, Louisiana, and South Carolina. Conversely, states with the lowest number of incidents are Hawaii, Idaho, Arizona, and California.

2 Clustering Analysis

To explore the dataset we employed the following clustering algorithms: K-Means, X-Means, Agglomerative Hierarchical Clustering (with different linkage methods), DBSCAN and Self Organizing Maps (SOM). We used the X-Means and SOM implementations provided by the `pyclustering`[12] library. K-Means, X-Means and SOM were used to cluster all the incidents in the dataset, while Hierarchical Clustering and DBSCAN were used to cluster only the incidents that happened in the state of Illinois.

[10]The data about the USA population as of the 2010 census was sourced from Wikipedia at the following URL: `https://en.wikipedia.org/wiki/2010_United_States_census`

[11]`https://mpdc.dc.gov/sites/default/files/dc/sites/mpdc/publication/attachments/MPD%20Annual%20Report%202014_lowres_0.pdf`

[12]`https://github.com/annoviko/pyclustering/`

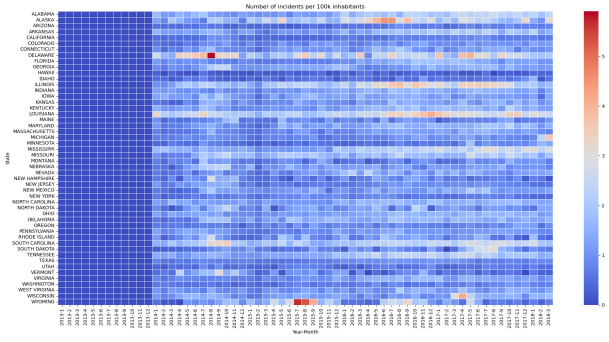


Figure 3: Number of incidents per 100K in the US states for each month of the years.

We chose to analyze this state because it had a substantial amount of records without missing values (17554 entries) and a feature distribution closely resembling that of the entire dataset. Notably, the Kullback-Leibler (KL) divergence for continuous attributes resulted in infinity, signifying identical distributions. The only feature showing a distinct distribution was `n_participants`, with a KL divergence of 0.4240. However, for the majority of features, Illinois exhibited behavior similar to that of the entire dataset, suggesting its potential suitability as a representative sample.

2.1 Features for clustering

The column 'Clustering' in Table 1 reports the feature we used in the clustering analysis. In addition to simple ratios between the number of people involved in the incident with a specific characteristic and the total number of participants, (e.g., `n_males_prop` or `n_teen_prop`), we also computed a set of indicators aimed at highlighting the degree of abnormality of the value of a specific feature of an incident with respect to the values of the same feature in the incidents happened in the same period and in the same geographical area. To achieve this we used the concept of *surprisal*. The *surprisal* (also called *Information content*) of an event E with probability $p(E)$ is defined as $log(1/p(E))$ (or equivalently $-log(p(E))$). Surprisal is inversly related to probability (hence the term $1/p(E)$): when $p(E)$ is close to 1, the surprisal of the event is low, whereas when $p(E)$ is close to 0, the surprisal of the event is high. The log gives 0 surprise when the probability of the event is 1. The surprisal is closely related to *entropy*, which is the expected value of the information content of a random variable, thus quantifying how surprising the random variable is "on average". Exploiting the concept of surprisal, we were able to compute indicators not only for categorical variables but also for a set of variables. Specifically, surprisals were computed w.r.t the incidents happened in the same semester of the same year and in the same congressional district of the same state, for the following sets of features: {`address_type`} (resulting in the variable `surprisal_address_type`), {`min_age`}

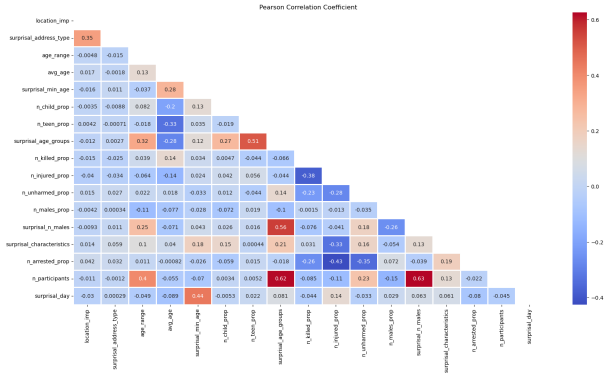


Figure 4: Pearson Correlation Coefficient between the features used for clustering.

(resulting in the variable `surprisal_min_age`), {`n_child`, `n_teen`, `n_adult`} (resulting in the variable `surprisal_age_groups`), `n_males` (resulting in the variable `surprisal_n_males`), {`firearm`, `air_gun`, `shots`, `aggression`, `suicide`, `injuries`, `death`, `road`, `illegal_holding`, `house`, `school`, `children`, `drugs`, `officers`, `organized`, `social_reasons`, `defensive`, `workplace`, `abduction`, `unintentional`} (resulting in the variable `surprisal_characteristics`) and {`month`, `day`} (resulting in the variable `surprisal_day`).

In addition to the indicators presented in the table, we also computed and examined the distribution and correlations of other indicators (such as the 'severity' of the incident, defined as a linear combination of the attributes `n_killed`, `n_injured` and `n_unharmed`). However, we ultimately excluded them from further consideration due to their high correlation with other attributes that we found more interesting or because of their highly skewed distributions.

2.1.1 Correlation between the features

Figure 4 displays the Pearson Correlation Coefficient between the newly defined features. The Pearson Correlation Coefficient measures the linear relationship between two continuous variables assuming that the variables are approximately normally distributed. In contrast, the Spearman Rank Correlation Coefficient measures the strength and direction of the monotonic relationship between two variables without assuming that the variables are normally distributed. It is based on the ranks of the data rather than the actual values, hence is less sensitive to outliers than the Pearson correlation. We compute the correlation matrix using both coefficients, and despite their distinct definitions, they yield similar results. The Pearson Correlation Coefficients ranges between [-0.43, 0.63], while the Spearman Rank Correlation Coefficients ranges between [-0.44, 0.7]. Higher positive or negative correlations are expected, considering the semantics of the indicators.

2.1.2 Distributions, statistics and quality of the features

Table 2 provides the statistics for the indicators. The indicator with the highest number of 'NaN' values is `surprisal_min_age`. Records containing at least one 'NaN' value constitute 60.92% of the dataset. The percentage of duplicated records among those without 'NaN' values amounts to 0.18%. It is important to note that the computed indicators should not exhibit errors, as potential inconsistencies were addressed during the data preparation phase. The distribution shapes of the indicators are illustrated in Figure 5. Many of them display a high degree of skewness and contain numerous outliers. We made the decision to retain all records without excluding outliers because we believe that such instances contain valuable information. It's worth acknowledging that outliers might impact clustering results.

2.2 Pre-processing

As illustrated in Figure 5, the indicators extracted for the clustering analysis have different scales and variances, necessitating normalization. The choice between Min-Max Scaling and Standard Scaling for data normalization entails both advantages and disadvantages. Min-Max scaling leaves variances unequal, but it preserves the shape of the original distribution. On the other hand, Standard Scaling makes the variances equal, but it does not preserve the shape of the original distribution. In certain clustering algorithms, such as K-Means, leaving variances unequal is like giving different weights to features based on their variance. At the same time, using Standard Scaling the feature `n_participants` – which resides in a different range from the other features – could potentially negatively affect the clustering results. Taking these factors into account, we opted for Min-Max Scaling because we wanted to keep the feature `n_participants` and because the features with higher inter-quartile range (e.g., `n_killed_prop`, `n_injured_prop`, `n_unharmed_prop`, `n_arrested_prop`), which could be weighted more in the clustering, actually capture the main characteristics of the incidents.

2.3 K-Means

2.3.1 Parameters of the algorithm

With the exception of the value for K, whose choice is later detailed, we used the default K-Means hyperparameters, i.e.: (1) euclidean distance as distance metric; (2) a maximum of 300 iterations; (3) initial centroids sampled based on an empirical probability distribution of the points' contribution to the overall SSE; (4) repeated execution of the algorithms for 10 times with different centroids initializations, choosing the best clustering according to the SSE.

2.3.2 Identification of the best value of K

To identify the best value of K we used the implementation of the elbow method from the library

Table 1: Features defined for clustering and classification.

Name	Description	Clustering	Classification	Rule-based	Distance-based
gun_law_rank	Gun law strength ranking in the state and year where the incident occurred	✗	✓	✓	✓
poverty_perc	Poverty percentage in the state and year where the incident occurred	✗	✓	✓	✓
democrat	Whether the Democratic Party won the last elections in the congressional district of the state and in the year where the incident occurred	✗	✓	✓	✓
location_imp	Location importance according to Geopy	✓	✓	✓	✓
x	Longitude projection using the Universal Transverse Mercator system (UTM)	✗	✓	✓	✓
y	Latitude projection using the Universal Transverse Mercator system (UTM)	✗	✓	✓	✓
surprisal_address_type	Surprisal of the address type retrieved via Geopy (e.g., "house", "street") w.r.t. the address types of incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
surprisal_day	Surprisal of the day of the month and of the month in which the incident happened w.r.t. the days of the month and the months of incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
day	Day of the month in which the incident occurred		✓	✓	
day_x	Projection on the x axis of the day of the month in which the incident occurred	✗	✓		✓
day_y	Projection on the y axis of the day of the month in which the incident occurred	✗	✓		✓
day_of_week	Day of the week in which the incident occurred	✗	✓	✓	
day_of_week_x	Projection on the x axis of the day of the week in which the incident occurred	✗	✓		✓
day_of_week_y	Projection on the y axis of the day of the week in which the incident occurred	✗	✓		✓
month	Month in which in the incident occurred	✗	✓	✓	
month_x	Projection on the x axis in which the incident occurred	✗	✓		✓
month_y	Projection on the y axis in which the incident occurred	✗	✓		✓
year	Year in which the incident occurred	✗	✓	✓	✓
days_from_first_incident	Days passed since the occurred of the first incident	✗	✓	✓	✓
aggression	Tells if the incident involved an aggression	✗	✓	✓	✓
accidental	Tells if the incident was accidental	✗	✓	✓	✓
defensive	Tells if the incident was defensive	✗	✓	✓	✓
suicide	Tells if the incident was a suicide	✗	✓	✓	✓
road	Tells if the incident occurred in a road	✗	✓	✓	✓
house	Tells if the incident occurred in a house	✗	✓	✓	✓
school	Tells if the incident occurred in a school	✗	✓	✓	✓
business	Tells if the incident occurred at a business	✗	✓	✓	✓
illegal_holding	Tells if the incident involved illegal holding	✗	✓	✓	✓
drug_alcohol	Tells if the incident involved drugs or alcohol	✗	✓	✓	✓
officers	Tells if the incident involved officers	✗	✓	✓	✓
organized	Tells if the incident was carried by an organization or a group	✗	✓	✓	✓
social_reasons	Tells if the incident involved social discriminations or terrorism	✗	✓	✓	✓
abduction	Tells if the incident involved any form of abduction	✗	✓	✓	✓
surprisal_characteristics	Surprisal of the values of the incident characteristic tags extracted from the incident characteristics w.r.t. the values of the tags for incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
age_range	Range of age of the participants involved in the incident	✓	✓	✓	✓
avg_age	Average age of the participants involved in the incident	✓	✓	✓	✓
n_child_prop	Proportion of children involved in the incident	✓	✓	✓	✓
n_teen_prop	Proportion of teen involved in the incident	✓	✓	✓	✓
surprisal_min_age	Surprisal of the minimum age of the participants involved in the incident w.r.t. the minimum age of the participants of incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
surprisal_age_groups	Surprisal of the number of child, teen and adult involved in the incident w.r.t. the number of child, teen and adult involved in incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
n_males_prop	Proportion of males involved in the incident	✓	✓	✓	✓
surprisal_n_males	Surprisal of the number of males involved in the incident w.r.t. the number of males involved in incidents happened in the same semester of the same year and in the same congressional district of the same state	✓	✗	–	–
n_killed_prop	Proportion of killed people in the incident	✓	✗	–	–
n_injured_prop	Proportion of injured people in the incident	✓	✗	–	–
n_unharmed_prop	Proportion of unharmed people in the incident	✓	✗	–	–
n_arrested_prop	Proportion of arrested people in the incident	✓	✗	–	–
n_participants	Number of participants involved in the incident	✓	✓	✓	✓
	Total number of features	17	–	31	34

Table 2: Statistics of the indicators defined for the clustering.

Indicator	Non-Nan	Mean	Std	Min	25%	50%	75%	Max
location_imp	204054	0.016	0.053	0.000	0.000	0.000	0.000	0.738
surprisal_address_type	204054	1.436	1.480	0.000	0.407	0.704	2.248	9.490
age_range	144731	3.505	8.203	0.000	0.000	0.000	3.000	82.000
avg_age	144731	30.179	12.383	0.000	21.000	27.000	36.000	101.000
surprisal_min_age	138954	4.899	1.100	0.000	4.170	4.807	5.555	9.615
n_child_prop	178814	0.011	0.092	0.000	0.000	0.000	0.000	1.000
n_teen_prop	178814	0.066	0.232	0.000	0.000	0.000	0.000	1.000
surprisal_age_groups	172004	2.235	1.530	0.000	1.177	1.700	2.720	9.615
n_killed_prop	191743	0.184	0.340	0.000	0.000	0.000	0.250	1.000
n_injured_prop	191743	0.333	0.413	0.000	0.000	0.000	0.667	1.000
n_unharmed_prop	188980	0.188	0.329	0.000	0.000	0.000	0.500	1.000
n_males_prop	176221	0.874	0.259	0.000	1.000	1.000	1.000	1.000
surprisal_n_males	169419	1.842	1.271	0.000	0.951	1.363	2.307	9.490
surprisal_characteristics	208710	3.677	1.681	0.000	2.354	3.555	4.907	9.615
n_arrested_prop	188980	0.286	0.401	0.000	0.000	0.000	0.500	1.000
n_participants	191743	1.822	1.192	1.000	1.000	2.000	2.000	103.000
surprisal_day	208710	5.715	1.134	0.000	5.000	5.755	6.484	9.615
Total number of records		216373						
Number of records without NaN values		131811						
Number of duplicated records without NaN Values		235						

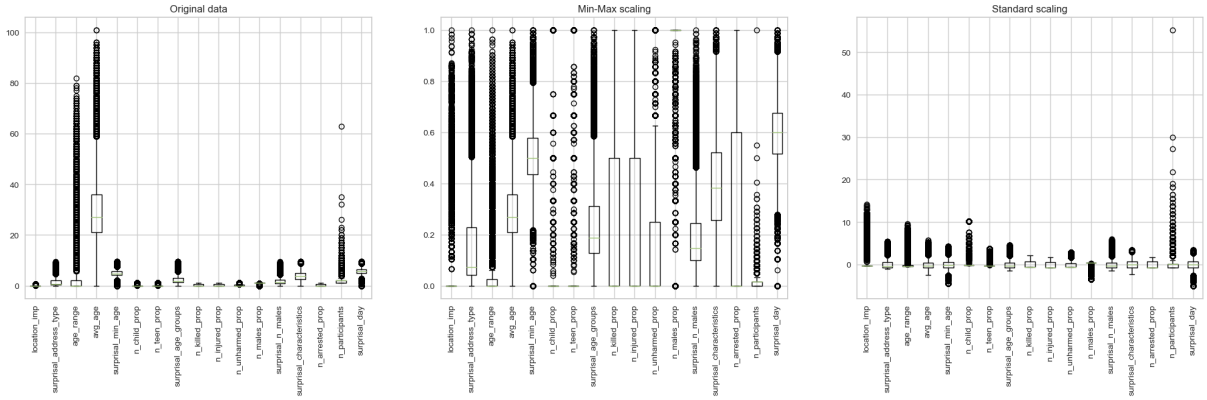


Figure 5: Indicators distributions.

yellowbrick[13]. This method consists in computing a metric to evaluate the quality of the clustering for each value of K, and then plotting the metric as a function of K. The best value of K is the one that corresponds to the point of inflection of the curve (the point where the metric starts to decrease more slowly). We first applied the elbow method evaluating the SSE and computing the elbow for each value of K between 1 and 10, 1 and 20 and 1 and 30 (higher values lead to results that are difficult to interpret). We considered also $K = 1$ (absence of clusters) and we repeated the analysis with K in different ranges because the inflection point of the curve could vary depending on the number of points the curve is made of. Figure 6 shows the resulting curves. The elbow method identifies $k = 4$, $k = 5$ and $k = 7$ as best possible values of K. At $K = 4$ the point of inflection is more evident. Then, we repeated the analysis with $K \in [2, 9]$ using the Silhouette score (whose computation requires higher running times) and the Calinski-Harabasz score (the ratio of the sum of between-cluster dispersion and of within-cluster dispersion). As evidenced in Figure 7 both scores reach their maximum values when K is set to 4.

To identify the best value of K we also used the X-means algorithm. X-means is a variant of the K-Means algorithm that should automatically find the best value of K. The algorithm starts with $K = 2$ and then it iteratively splits the clusters until no further improvement in the cluster structure is observed. To measure the quality of the cluster structure we used both the Bayesian Information Criterion (BIC) and the Minimum Noiseless Description Length (MDL) scores. Using both the metrics, X-means terminated with K equal to the maximum number of clusters allowed (30 in our case). This means that the score always improves when splitting the clusters. Consequently no value of K is optimal according to these criteria.

To choose the best value of K among the ones identified by the elbow method, we also computed the Between-cluster Sum of Squares (BSS) and the Davies-Bouldin score (i.e., the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances). Table 3 reports the values of the computed scores and the number of iterations performed by the algorithm. With $K = 4$ and $K = 5$ the algorithm stops after 6 iterations. With $K = 7$, 21 iterations are needed to converge. These results confirm that the value chosen to set the maximum number of iterations (300), was enough. We observe that SSE and BSS are both best for $K = 7$, but these metrics are expected to improve while increasing the number of clusters. Davies-Bouldin score, Calinski-Harabasz score and Silhouette score are all best for $K = 4$. We could have also used hierarchical clustering to identify the best value of K. However, because of the high computational cost of hierarchical clustering, we should have used a subset of the data, hence, risking to lose representativeness. Furthermore, since the methods used are concordant in identifying $K = 4$ as the best value of K, we decided to conclude the search for the best value of K at this point, considering $K = 4$ as the optimal number of clusters.

[13] https://www.scikit-yb.org/en/latest/api/cluster/elbow.html

2.3.3 Characterization of the clusters

Figure 8 shows the coordinate plot for the resulting centroids. We observe that some features do not vary much between the clusters, while others do. Specifically: (1) the centroid of cluster 0 has a higher proportion of injured people and teens, a lower proportion of arrested people and lower values for surprisal_characteristics; (2) the centroid of cluster 1 has a higher proportion of arrested people; (3) the centroid of cluster 2 has a higher proportion of killed people and a lower proportion of injured people; (4) the centroid of cluster 3 has a higher proportion of unharmed people.

Studying the distribution of variables within the clusters and comparing it to that in the whole dataset (not shown here), we had a confirmation of the observations made by looking only at the centroids. The fact that incidents are grouped according to the variables n_injured_prop, n_arrested_prop, n_killed_prop and n_unharmed_prop is probably due to the fact that these features have larger interquartile ranges compared to the others (see Figure 5).

2.3.4 Evaluation of the clustering results

Figure 9 shows the Silhouette plot for $K = 4$. Few points have a negative silhouette score. The majority of points with a negative silhouette score belong to cluster 3. Also, the majority of points of cluster 3 have a silhouette score below the average.

Figure 10 shows the distance matrix sorted by the cluster labels and the ideal distance matrix for a stratified subsample of 5000 points. The Pearson Correlation Coefficient between the two matrices is 0.62. Indeed, the matrix on the left has a sharp block diagonal structure, meaning that clusters are well separated.

Due to space constraints, we won't report the values of the additional metrics computed to validate the clustering. We will only provide a description of the key insights derived from the evaluation. We observed that the clusters are also well separated in the space of the first three principal components. Computing BSS and SSE for each cluster separately we observed that: (1) cluster 0 is the less cohesive but it is the most separated; (2) cluster 1 is both cohesive and well separated; (3) cluster 3 is the lowest separated (this was also evident from the Silhouette plot). Analyzing the contribution to the SSE for each feature separately we found that the features that contribute the most are n_males_prop and n_teen_prop, while the feature that contributes less is n_participants.

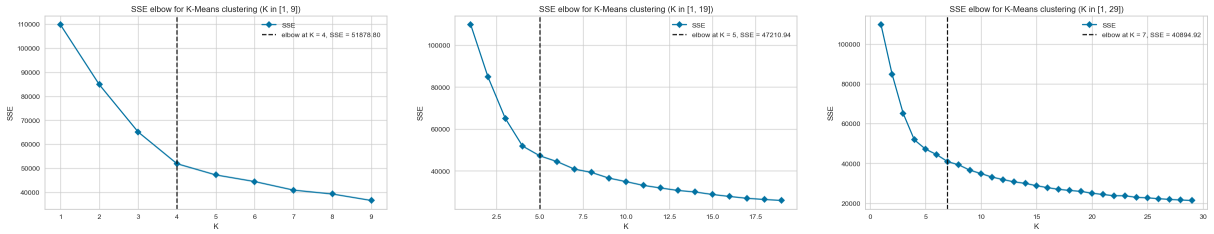


Figure 6: SSE varying the number of clusters identified by K-Means. The curve has inflection points for $K = 4$, $K = 5$ and $K = 7$.

Table 3: K-Means clustering scores for different values of K.

K	SSE	BSS	Calinski-Harabasz	Davies-Bouldin	Silhouette	# Iterations	Cluster sizes
4	5.188×10^4	5.786×10^4	4.900×10^4	1.213	0.327	6	1: 40711; 2: 31460; 3: 31389; 0: 28251
5	4.721×10^4	6.253×10^4	4.364×10^4	1.456	0.299	6	2: 30377; 1: 28640; 3: 27375;4: 25979; 0: 19440
7	4.089×10^4	6.885×10^4	3.698×10^4	1.397	0.303	21	3: 28513; 5: 26781; 6: 20842; 0: 20202; 1: 19382; 2: 10031; 4: 6060

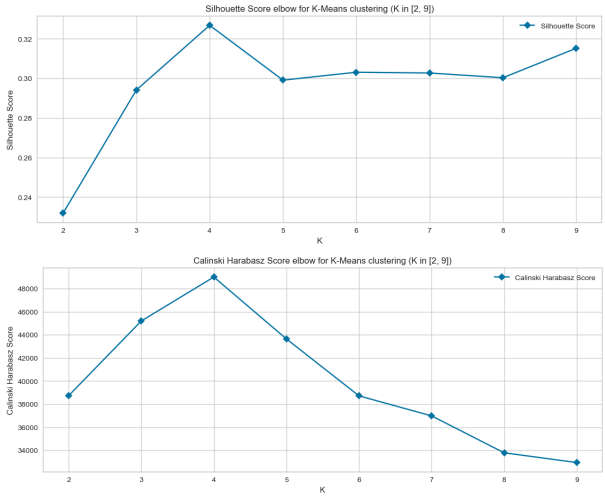


Figure 7: Silhouette and Calinski-Harabasz scores varying the number of clusters identified by K-Means.

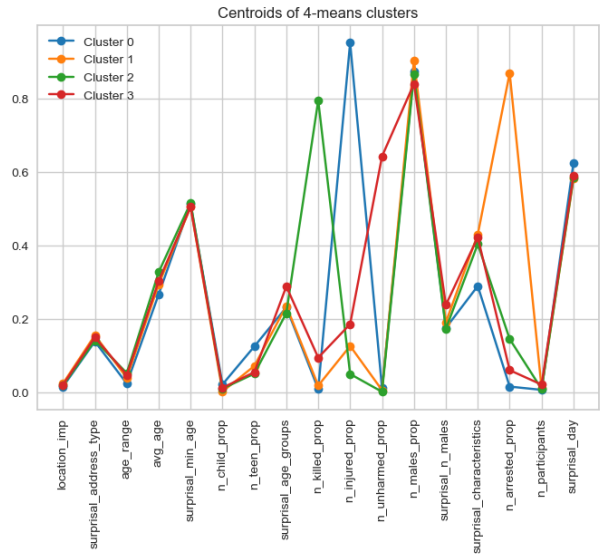


Figure 8: Coordinate plot of the cluster centroids obtained with 4-Means (standardized variables).

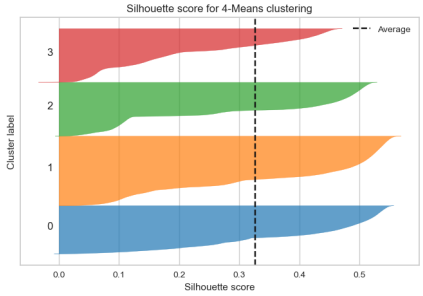


Figure 9: Silhouette plot of the clustering obtained with 4-Means.

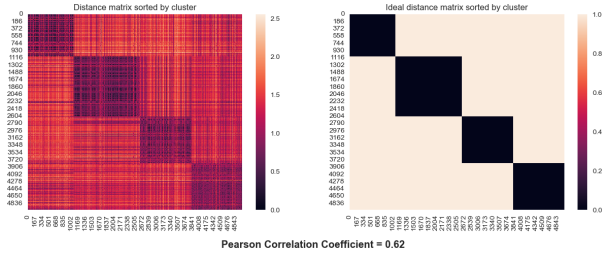


Figure 10: Distance matrix sorted by the clustering labels found by 4-Means and ideal distance matrix.

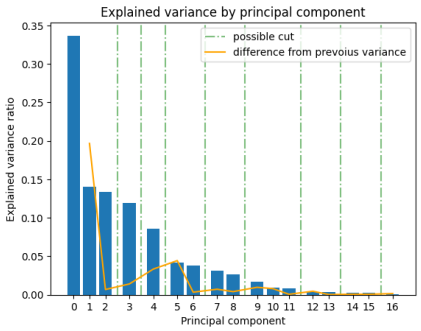


Figure 11: Explained variance of each principal component.

2.4 K-Means and Principal Component Analysis

We also attempted to apply K-Means after transforming the data using Principal Component Analysis (PCA). This pre-processing step should lead to better clustering results because it allows to select meaningful orthogonal variables, thereby reducing noise in the data. However, a drawback is that the transformed features are more difficult to interpret. We identified the number of components to use by plotting the reconstruction error and the explained variance varying the numbers of components (see Figure 11) and chose the to use the first 8 components because they capture most of the variability of the data. We observed that: (1) the first component is correlated with the variables `n_injured_prop` and `n_arrested_prop`; (2) the second component is correlated with the variable `n_killed_prop`; (3) the third component is mostly correlated with `n_unharmed_prop` and `n_arrested_prop` and slightly correlated with `n_killed_prop` and `n_injured_prop`; (4) the fourth component is correlated with `n_teen_prop` and `avg_age`. These variables were also those that mainly separate the clusters identified with K-Means on raw data. We performed the same analysis described in Section 2.3.2 to identify the best value of K and once again, the optimal value of K was 4. To evaluate the quality of clustering we applied the same metrics described in the previous sections. However, the clustering scores achieved in this setting are comparable to that obtained on raw data. Ultimately the results obtained differ little from those obtained previously, with the silhouette score decreasing slightly to 0.324.

2.5 Final considerations

One of the advantages of K-Means is its computational efficiency, which gives us the possibility to use it on all the dataset and to apply it several times to choose the best configuration. However, this clustering algorithm could be affected by outliers and is biased towards clusters with globular shapes. Furthermore, to apply the algorithm the number of clusters must be known a priori. If it is not – as in this setting – a careful validation of different choices has to be performed. Finally, the resulting clustering could vary depending on the initialization of the centroids. One approach to mitigate this problem is to initialize the centroids with the final centroids computed by Bisecting K-Means. We also explored this approach, observing a slight improvement in the Davies-Bouldin score (1.189) and in the Silhouette score (0.330). However, the other metrics did not improve. This means that the repeated execution with different initializations already addressed the problem of the instability of the results.

2.6 Hierarchical Clustering

We employed agglomerative hierarchical clustering algorithms to cluster incidents in the state of Illinois using four different linkage types: Single, Complete, Average, and Ward. We searched for the optimal cut by calculating the Silhouette score for each clustering obtained by cutting the tree at the 10 merging steps with the greatest distance between merged clusters. Table 4 summarizes the results, while Figure 12 displays the dendrograms with their corresponding optimal cuts. Applying single-linkage, the outcome resembles what we could achieve by incrementally adding points to a starting cluster. Complete-linkage produces a more balanced tree, but with one cluster significantly larger than others, containing 4553 points. Average-linkage yields a compromise between single and complete linkage results, as expected. Ward linkage provides the most balanced outcomes, both in terms of tree structure and cluster sizes. Also regarding the quality of the clustering, the algorithm using Ward linkage achieved the best Silhouette score.

Figure 13 shows the Silhouette plot for the different linkage methods. Complete linkage presents more points with a negative silhouette score compared to Ward linkage. In fact, Ward linkage achieves a higher value for the average silhouette score. To evaluate the clustering results we also computed the Cophenetic Correlation Coefficient (CCC), a metric that measures the correlation between the entries of the cophenetic distance matrix and the dissimilarity matrix. The cophenetic distance between two objects is the proximity at which an agglomerative hierarchical clustering technique puts the objects in the same cluster for the first time. The best CCC is achieved by the average linkage method. However, this metric assesses the clustering

Table 4: Agglomerative hierarchical clustering scores and characteristics for different linkage methods.

Method	Cut height	Merging difference	# Clusters	Cluster Sizes	CCC	Silhouette Score
single	0.921	0.032	3	0: 13229; 1: 1; 2: 1	0.790	*0.318*
complete	2.108	0.095	10	0: 5637; 2: 2554; 3: 1695; 4: 1190; 1: 833; 5: 715; 6: 470; 8: 59; 9: 57; 7: 21	0.721	0.326
average	1.746	0.106	3	0: 13223; 2: 5; 1: 3	**0.833**	0.331
ward	24.665	3.480	7	2: 4188; 0: 4145; 3: 1806; 5: 1119; 1: 915; 4: 675; 6: 383	*0.699*	**0.368**

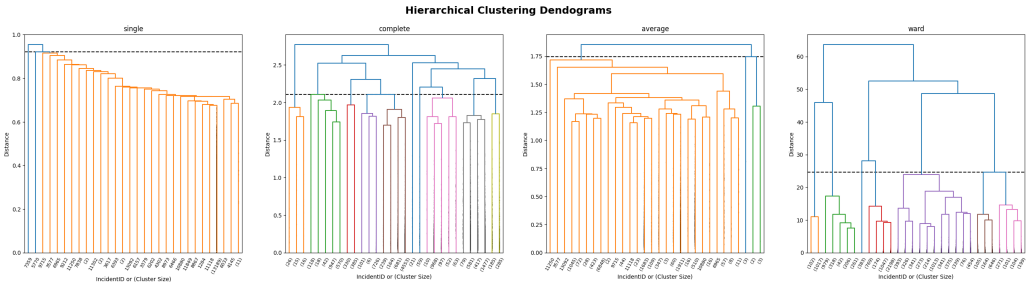


Figure 12: Dendrograms for each linkage method with the optimal cut based on the Silhouette score.

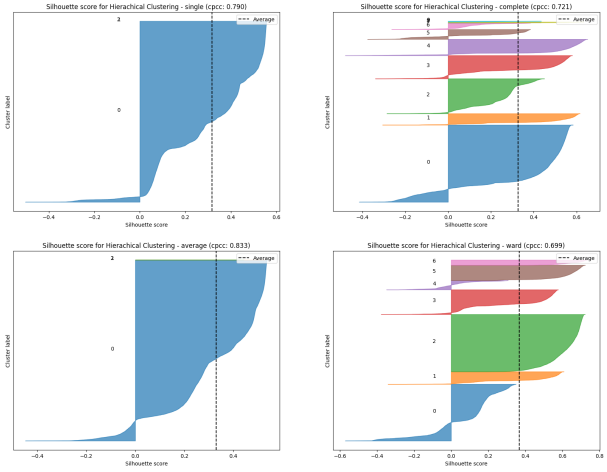


Figure 13: Silhouette plot for each method of hierarchical clustering.

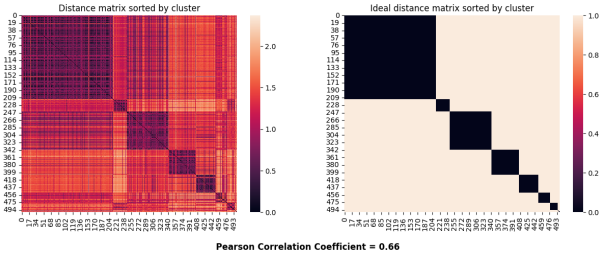


Figure 14: Distance matrix sorted by the clustering labels found by Complete-linkage Hierarchical clustering and ideal distance matrix.

process as a whole and does not offer an evaluation of the clustering obtained by cutting the tree at a certain heigh. Examining the sizes of the resulting clusters, it is evident that both the single and average linkage methods result in unbalanced clustering, with one large cluster containing almost all the points. In contrast, the complete and ward linkage methods produce more balanced clusterings. Finally we also visualized the distance matrix between the points, sorted by the cluster labels, for Complete and Ward linkage. Figure 14 and 15 show these matrices compared to the ideal distance matrix. However, it's worth noting that this type of visualization is more suitable for evaluating clustering algorithms that produce globular clusters.

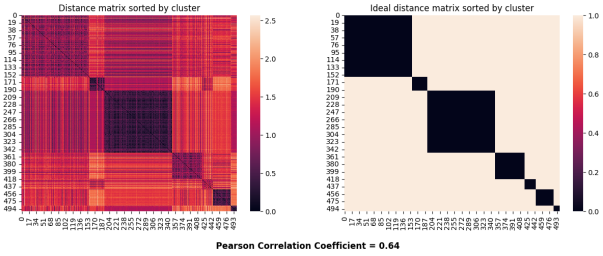


Figure 15: Distance matrix sorted by the clustering labels found by Ward Hierarchical clustering and ideal distance matrix.

2.6.1 Characterization of the clusters

We chose to analyze the clustering produced by the Ward linkage method with the identified cut threshold because it has the highest silhouette score. By comparing the distributions of the features in each cluster and in the whole dataset we found that the features that are distributed differently are `n_killed_prop`, `n_injured_prop`, `n_unharmed_prop`, `n_arrested_prop`, `n_participants`, `suprisal_n_males` and the attributes related the age of the participants.

2.6.2 Final considerations

The benefit of hierarchical clustering algorithms lies in their innate ability to reveal nested clustering structures; however, they come with significant computational and storage costs. Moreover, hierarchical clustering is not explicitly designed for producing partitioned clusters. Determining the optimal cut threshold presents a challenge, as it often involves a tradeoff between achieving balanced cluster sizes and ensuring cohesion and separation in the clustering.

2.7 Density-based clustering

Among density-based clustering algorithms, we chose to utilize DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Since DBSCAN encounters difficulties with high-dimensional data we carefully selected a subset of features, excluding those exhibiting the highest correlation with other features. The final set of chosen features includes: `surprisal_address_type`, `avg_age`, `n_killed_prop`, `n_arrested_prop`, `n_males_prop`, and `n_participants`. The Kendall Correlation Coefficient between these selected features ranges from -0.22 to 0.12.

2.7.1 Parameter selection

At first, to determine the range for exploring the parameter `eps`, we plotted the distance between each point and its k^{th} nearest neighbor for various values of k (3, 5, 9, 15, 20, 30), sorting points based on this distance. Figure 16 displays the resulting plots. Recognizing the knee of the curves within the range [1.25, 2], we opted to concentrate on exploring this interval for the epsilon parameter. Afterward, the selection of the best value for both the parameters `eps` and `min_samples` considered multiple factors, including the silhouette coefficient, the number of clusters, their size, and the percentage of detected noise. During these exploration we decided not to use `min_samples` greater than 20 since, in preliminary trials, we noticed that opting for higher values of `min_samples`, despite yielding higher silhouette coefficients, resulted in highly unbalanced clusters, with one cluster grouping the majority of points. The seven best outcomes from these trials are detailed in Table 5, arranged in descending order based on the silhouette coefficient. We opted to analyze the clustering produced by the algorithm configured with `min_samples=20` and `eps=1.5`, as, among these, it exhibited the highest silhouette score.

2.7.2 Evaluation of the clustering results

The chosen configuration of the algorithm grouped incidents into 6 clusters. About 1.27% of the points were labeled as noise, constituting 168 samples. The largest cluster encompasses 9941 samples, representing 75.12% of the dataset. Figure 17 presents the silhouette score for each point in every cluster. Notably, the most populated clusters (cluster 0 and 1) contain points with negative silhouette scores. The clusters with the highest silhouette scores are 5 and 6, although they are the least populated. The distance matrix, organized by cluster labels, is depicted in Figure 18, and its Pearson Correlation Coefficient with the ideal distance matrix amounts to 0.43.

2.7.3 Characterization of the clusters

Clusters appear to group the data primarily based on the variables `n_killed_prop`, `n_arrested_prop`, and `n_males_prop`, with other features exhibiting a relatively uniform distribution across the clusters. The features with the most homogeneous distribution within clusters and among points labeled as noise are `surprisal_address_type` and `avg_age`.

The variables `n_arrested_prop` and `n_males_prop` exhibit a uniform distribution in clusters 0 and 1. In clusters 2 and 3, these variables take values close to 0, while in cluster 4, they assume a value of 1.

2.7.4 Noise analysis

Among the incidents classified as noise by the algorithm: (1) 113 incidents involve at least one woman (in the overall dataset, 15.35% of incidents involve women); (2) for 104 incidents, the average age exceeds 27, which is the overall average age of incident participants in the dataset; (3) 102 incidents involve more than two participants, which represent the fourth quartile of the number of participants in incidents. Additionally, all data points with a number of participants greater than 6 are classified as noise. These points are identified as outliers in the distribution of the number of participants, both within the data used for clustering and in the entire dataset. These results suggest that the DBSCAN algorithm correctly identified noisy points.

2.7.5 Final considerations

DBSCAN offers the advantages of being resistant to noise and capable of handling clusters with arbitrary shapes and sizes. However, it faces challenges when clusters exhibit widely varying densities and when applied to high-dimensional data. Tuning its parameters can be more challenging compared to some other algorithms. Additionally, the computation of nearest neighbors involves calculating all pairwise proximities, making it computationally intensive.

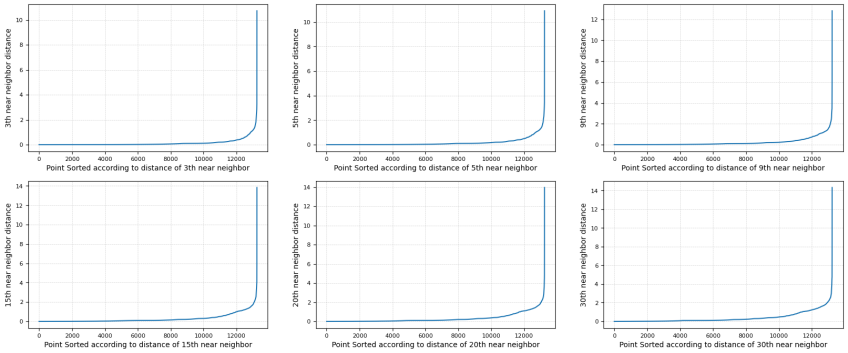


Figure 16: Distances between each point and its kth nearest neighbor for various values of k.

Table 5: DBSCAN clustering scores and characteristics for different parameters.

eps	min_samples	# Clusters	Noise	% Noise	Silhouette
1.50	**20**	**6**	**168**	**1.269**	**0.251**
2.00	12	3	54	0.408	0.243
2.00	7	4	38	0.287	0.242
2.00	20	3	69	0.521	0.241
1.75	7	4	61	0.461	0.232
1.50	7	5	83	0.627	0.221
1.50	12	5	111	0.839	0.219

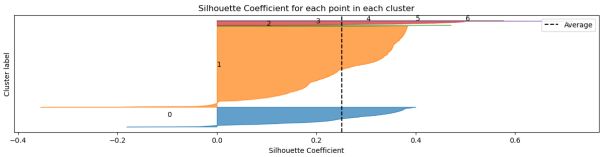


Figure 17: Silhouette plot of the clustering obtained with DBSCAN.

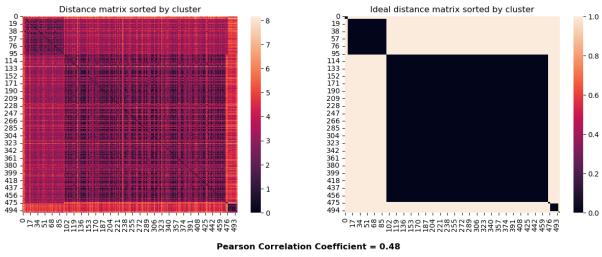


Figure 18: Distance matrix sorted by the clustering labels found by DBSCAN and ideal distance matrix.

2.8 Clustering with Self Organizing Maps

Self Organizing Maps (SOM) is an unsupervised neural network model that could be used both for dimensionality reduction, visualization of high-dimensional data and clustering. This model organizes the input data into a lower-dimensional grid of neurons (numerical vectors of 'weights'), where each neuron represents a cluster prototype. Neurons weights are randomly initialized and, during training, the network adjusts its weights to map similar input patterns to neighboring locations on the grid, preserving the topological relationships of the input data.

Determining the best configuration of the parameters of the algorithm was out of the scope of this analysis. We opted for the default parameters (i.e `init_learn_rate`=0.1, `autostop`=True, `adaptation_threshold`=0.001, `init_radius`=2, `init_type`='distributed in line with uniform grid') and we used a 3x3 grid (i.e., 9 clusters) with at most 4 neighbors per cell.

2.8.1 Characterization of the clusters

To explore the topological relationship between the clusters we colored each cell of the grid based on the mean for numerical variables or mode for categorical variables of the corresponding cluster's features. Figure 19 displays a selection of these plots. We observe that: (1) the up-left corner of the grid groups incidents with higher values of `n_injured_prop`; (2) the up-right corner groups incidents with higher values of the variable `n_arrested_prop`; (3) the down-left corner groups incidents with higher values of the variable `n_unharmed_prop`; (4) the down-right corner groups incidents with higher values of the variable `n_killed_prop`; (5) the center of

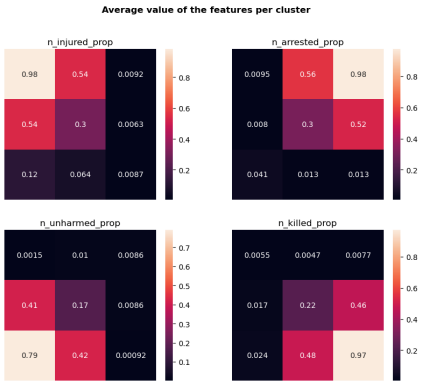


Figure 19: SOM map colored by the average values of the features in the clusters.

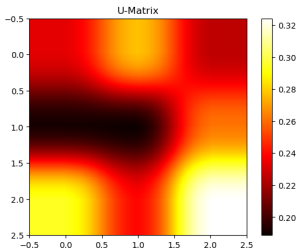


Figure 20: U-Matrix for Self Organizing Maps clustering.

the grid groups incidents with higher values of the variable `suprisal_age_groups`, `surprisal_n_males`, `suprisal_characteristics` and `n_participants`, while lower values of the variable `n_males_prop`. Similar considerations were also drawn by examining the coordinate plot of the cluster centroids and the distributions of the variables within the clusters.

Figure 20 displays the resulting U-Matrix (a visualization of the grid in which each cell is colored according to the average distance between the weights of the cell and its neighbors). We notice that the prototype of the cell in the middle of the grid and the one on the left of the central row are the most similar to their adjacent cells. Instead, the prototype of the cluster in the down-right corner is the most different from the adjacent prototypes.

2.8.2 Evaluation of the clustering results

Table 6 shows the clustering scores and the size of the clusters found. Cluster labels are assigned sequentially based on the order of the grid cells, from left to right and top to bottom. There are two big clusters, two clusters of medium size and five smaller clusters.

Figure 21 shows the Silhouette plot for the clustering. The average Silhouette score is 0.244, the lowest among all the clustering algorithms. Indeed, many points exhibit a negative silhouette score. In cluster 4 (represented by the prototype on the left of the central row of the grid) the majority of points have a negative silhouette score. The lowest performance of the algorithm

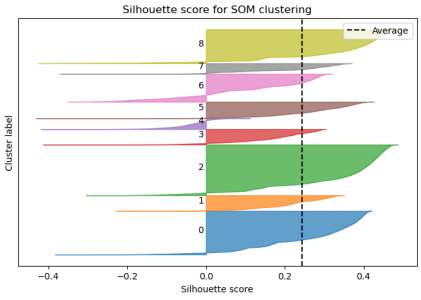


Figure 21: Silhouette plot of the clustering obtained with SOM.

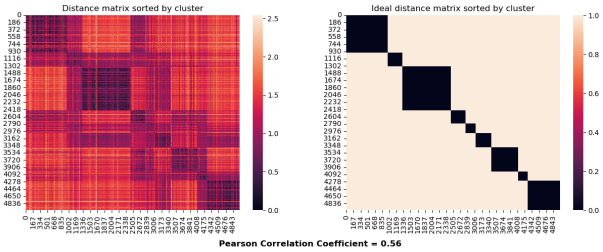


Figure 22: Distance matrix sorted by the clustering labels found by SOM and ideal distance matrix.

may be attributed to the fact that we did not validate its parameters, testing solely their default values.

The distance matrix sorted by the cluster labels for a stratified subsample of 5000 points is shown in Figure 22. The diagonal structure is not as discernible as that obtained with K-Means. Its Pearson Correlation Coefficient with the ideal distance matrix stands at 0.56.

The 9 clusters resulted well separated in the space of the first three principal components. As in K-Means, the features that contribute the most to the SSE are `n_males_prop` and `n_teen_prop`.

2.8.3 Final considerations

Similar to K-Means, SOM is computationally efficient but its applicability is limited to data where the concept of a centroid is meaningful. One of its notable strengths lies in its ability to capture the topological structure of the input space within the map, facilitating the visualization of cluster positions. However, its limitations include sensitivity to weight initialization and the constraint imposed on the number of clusters by the chosen grid structure.

2.9 Clustering Comparison

Although DBSCAN and Hierarchical clustering were exclusively applied to incidents that occurred in Illinois, while the remaining clustering algorithms were utilized on the entire dataset, the comparative analysis presented in this section remains meaningful. This is due to the similarity in the distribution of indicators for incidents in the state of Illinois and the distribution of variables in the entire dataset, as previously described in Section 2.

Table 6: SOM clustering scores and cluster sizes.

SSE	BSS	Calinski-Harabasz	Davies-Bouldin	Silhouette	Cluster sizes
4.929×10^4	4.425×10^4	2.602×10^4	2.040	0.244	0:25497; 1:9165; 2:29673; 3:9060; 4:6381; 5:9613; 7:16363; 8:6192; 9:19867

Table 7 summarizes the silhouette scores achieved by the different clustering algorithms. We exclusively rely on this internal metric for comparison purposes, as it is not computed based on cluster centroids, which can be irrelevant for some algorithms. Ward Hierarchical clustering achieved the highest Silhouette scores, while SOM and DBSCAN achieved the lowest scores. K-Means on raw data and on the first 8 principal components achieved similar results. Ward Hierarchical clustering exhibited the highest Silhouette scores, with DBSCAN yielding the lowest. K-Means applied to raw data and to the first 8 principal components produced comparable results. For comparison, Table 7 also presents the average Silhouette score and its standard deviation, obtained by randomly assigning labels to the records 10 times. In this setting the Silhouette score is around 0.

We also measured the extent to which the discovered clustering structure matched some categorical features of the dataset not employed in the clustering process, using the following external scores: Adjusted Rand Index, Normalized Mutual Information, Homogeneity, Completeness. All of them are permutation invariant. The values of each score for the different categories and algorithms is displayed in Figure 23. We observe that: (1) DBSCAN clusters best match the category `death` according to all the metrics; (2) the classes `unharmed` and `arrested` are well matched by the clusters found by K-Means, Hierarchical clustering and SOM, with K-Means achieving the highest scores; (3) the tags `drugs`, `illegal_holding` and `suicides` do not match with any clustering.

Finally, we used a Sankey diagram (Figure 24) to visualize the relationships between the clustering labels assigned to the points by the different algorithms. Since DBSCAN and Hierarchical clustering were applied only to the incidents happened in Illinois, we restrict the comparison to the incidents in this state. The clusters found by K-Means on raw data and on the first principal components are very similar. Cluster 2 of DBSCAN groups almost all the points from cluster 0, 1 and 3 of K-Means, while cluster 1 of DBSCAN groups almost all the points from cluster 2 of K-Means. Cluster 2 of K-Means on the first principal components groups points belonging mainly to cluster 0, 2 and 3 of the Hierarchical clustering algorithm. No relationship is observed between Hierarchical clusters and SOM clusters, despite the fact that the algorithms found the same number of clusters.

Given the differences in the labelings and the and comparable silhouette scores achieved by the algorithms, we can infer that incidents in the indicators space do not demonstrate inherent natural grouping. There are multiple ways to group the incidents achieving the same clustering scores.

3 Predictive Analysis

3.1 Classification models

We conducted experiments with 10 different classifiers, leveraging implementations from various libraries. Specifically, we utilized the implementations of Decision Trees (DT), Random Forest (RF), Ada Boost (AB), K Nearest Neighbors (KNN), Nearest Centroid (NC), and Support Vector Machine (SVM) from the `scikit-learn` library[14]. Extreme Gradient Boosting (XGB) was implemented using the `XGBoost` library[15]. For Neural Networks (NN), we employed the `Keras` library[16]. The Naive Bayes Classifier (NB) was implemented using the library `mixed-naive--bayes`[17], and the RIPPER algorithm was exploited from the library `wittgenstein`[18]. Initially, we explored various probabilistic classifiers (e.g., Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes). However, many of these classifiers make assumptions about the type of features and, consequently, their distribution. To ensure a fair comparison among classifiers, we have chosen to present here only the classification performance of Naive Bayes for mixed data. This classifier was applied to the same set of features used with the other classifiers.

3.2 Features for classification

The column 'Classification' in Table 1 reports the feature we used to train the classifiers. Depending on the classification algorithm, some features needed to be properly encoded. The column 'Rule-based' reports the features utilized with the following classifiers: Ripper, Decision Trees, Random Forest, Ada Boost, Extreme Gradient Boosting, Naive Bayes. It's noteworthy that these classifiers do not require feature encoding. The column 'Distance-based' details the features utilized with the following classifiers: K Nearest Neighbors, Nearest Centroid, Support Vector Machine, Neural Networks. We kept all the features used for the clustering analysis, excluding those

[14] https://scikit-learn.org/stable/

[15] https://xgboost.readthedocs.io/en/stable/python/index.html

[16] https://keras.io

[17] https://github.com/remykarem/mixed-naive-bayes

[18] https://github.com/imoscovitz/wittgenstein

Table 7: Silhouette scores achieved by the different clustering algorithms.

Algorithm	# Clusters	Silhouette score (std)
K-Means	4	0.327 (-)
K-Means PCA	4	0.324 (-)
Random labeling	4	-0.001 (0126)
DBSCAN	7	0.251 (-)
Random labeling	7	-0.003 (0317)
Ward Hierarchical Clustering	9	**0.368** (-)
Self Organizing Maps	9	*0.244* (-)
Random labeling	9	-0.003 (0420)

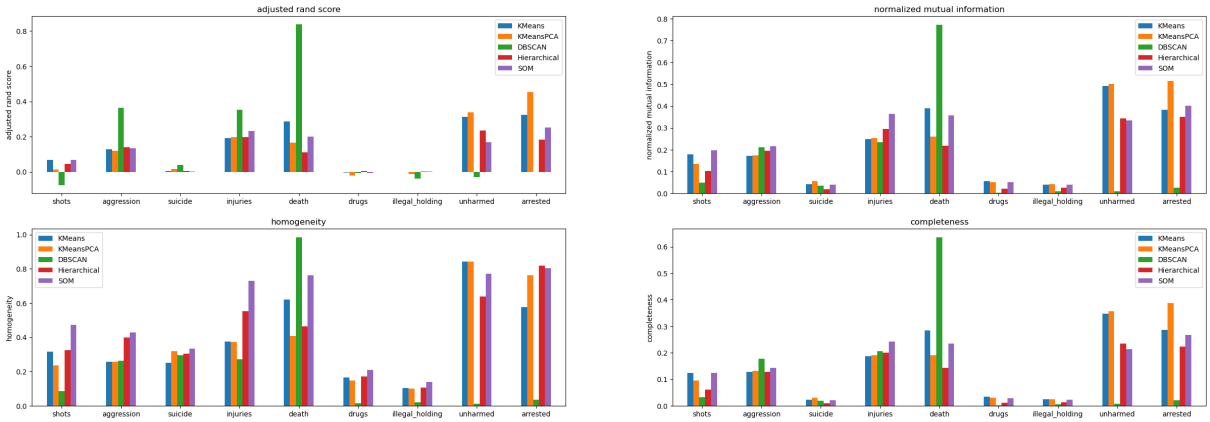


Figure 23: External scores for different categories and algorithms.

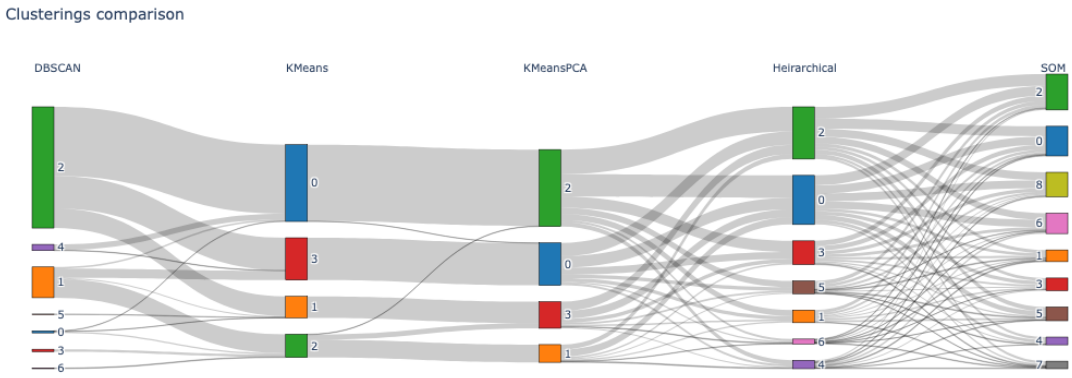


Figure 24: Sankey diagram of the labels assigned by the different clustering algorithms.

correlated to the target variable or that could be partially inferred from the feature n_participants (i.e., n_killed_prop, n_injured_prop, n_unharmed_prop, n_arrested_prop) and those involving the concept of surprisal. The exclusion of the latter was implemented to prevent any information leakage from the testing set into the training set. To enable the classification task we defined new features, the semantics of which are provided in the following sections.

3.2.1 Political-economic features

Every year, the Giffords Law Center to Prevent Gun Violence[19] assesses the gun regulations of all 50 states in the United States, assigning each state a letter grade reflective of the strength of its gun laws. The grades are determined by the efficacy and robustness of the state's firearm regulations, with the center asserting that states with stringent gun laws have fewer gun deaths. We gathered data spanning the period from 2013 to 2018. The Giffords Law Center to Prevent Gun Violence does not grade the District of Columbia. Based on a research of the gun laws in the District of Columbia,[20] we assigned it an 'A'. We then defined the feature gun_law_rank, mapping the letter grades to numerical values representing the rank of the state's gun laws (e.g., 1 for states ranked with 'A+', 2 for states ranked with 'A', etc.).

We also used the features poverty_perc and democrat, derived from the datasets 'poverty_by_state_year.csv' and 'year_state_district_house.csv' respectively.

3.2.2 Spatial features

To employ latitude and longitude with euclidean distance-based algorithms, we projected their values using the Universal Transverse Mercator system (UTM). This transformation was carried out through the pyproj library[21] using UTM zone 14, which corresponds to the center of the United States. We did not used the variable 'state' as its one-hot encoding, essential for some algorithm, would have significantly increased the number of variables, potentially introducing dimensionality issues. Additionally, this variable is already correlated to the variables gun_law_rank, poverty_perc and democrat.

3.2.3 Temporal features

We split the feature 'data' in the features 'day', 'month' and 'year' and computed the corresponding day of the week. Encoding these variables with progressive numbers, like assigning 1 to January, 2 to February, and so on for the feature 'month', leads to a loss of seasonality. This approach treats January and December as the most distant months, overlooking their chronological proximity in the yearly cycle. On the other hand, using one-hot encoding results in increasing the number of features by the number of values each feature can assume, introducing a potential dimensionality challenge. Furthermore, this encoding loses the similarities between nearby values. To overcome these limitations we opted for a 'cyclic encoding' of the variables, mapping the values on a circle centered in (0,0) with radius 1. As an instance, for the attribute month we used the following transformations: $month\_x = sin\left(\frac{2 \cdot \pi \cdot month}{12}\right)$; $month\_y = cos\left(\frac{2 \cdot \pi \cdot month}{12}\right)$.

To capture the sequentiality of the incidents, we introduced a feature that counts the days passed since the occurrence of the first incident. Consequently, the deployed classifiers can only be tested on incidents that occurred after the oldest incident present in the dataset we used to train the model.

3.2.4 Incidents characteristics

To describe the incidents characteristics we defined the following binary indicators: aggression, accidental, defensive, suicide, road, house, school, business, illegal_holding, drug_alcohol, officers, organized, social_reasons, abduction. These indicators were computed based on the features incidents_characteristics1 and incidents_characteristics2. We defined and used similar tags also for the exploratory analysis (Section 1.1.4); however, in this case, we adopted a less conservative approach. Previously, when faced with insufficient information to ascertain whether an incident exhibited a specific characteristic, we defaulted to setting the corresponding variables to 0 (indicating the absence of the characteristic). Here, however, we attempted assigning tags even when concrete information was lacking (e.g., for incidents with the characteristic 'School Incident' we set to 1 the variable aggression, since guns are not expected to be present in schools, hence shots could not be accidental and are more likely to be indicative of malicious intent). Nevertheless, in instances where information was lacking, we set the variables to 0. We did not employ another value (e.g., 'NaN'), as the one-hot encoding of the variables – needed to apply some classifiers – would have increased the number of variables. We also took into account alcohol consumption and redefined the previously used tag drug to drug_alcohol, setting it to 1 also for incidents presenting the characteristic 'House party' or 'Bar/club incident - in or around establishment'. Please note that the tags aggression and defensive are not mutually exclusive for the incidents in which a defensive act was undertaken in response to an aggression (e.g., 'Home Invasion'), as well as the variable suicide and aggression that are both 1 for Murder-suicides.

3.2.5 Correlation between the features

Figure 25 displays the Pearson Correlation Coefficient between the features used for the classification task and the target variable (the variable 'death'). No variable

[19] https://giffords.org/about/

[20] https://giffords.org/lawcenter/gun-laws/washington-dc/

[21] https://github.com/pyproj4/pyproj

Table 8: Class distribution in training and test folds.

Set	Fatal (%)	Non-Fatal (%)
Training	28175 (32.01)	59835 (67.99)
Training random oversampling	39890 (40.00)	59835 (60.00)
Training SMOTE	39890 (40.00)	59835 (60.00)
Test	14087 (32.01)	29918 (67.90)

exhibits a high correlation with the target. The feature `suicide` demonstrates the strongest correlation with the target, with a Pearson Correlation coefficient of +0.24. The relatively low value of the coefficient stems from the fact that 6.263% of the suicides in the incidents were attempted, resulting in a count of zero for the number of killed individuals. Other slightly correlated features are `illegal_holding` (-0.14), `avg_age` (+0.11), and `aggression` (-0.11). As expected, the variables `year` and `days_from_first_incident` are highly correlated (+0.97). However, no correlation is observed among the other variables.

3.3 Validation schema and pre-processing

To ensure a fair comparison, we excluded records containing missing values in the features specified for classification, as some implementations of the classifiers we utilized do not support handling such values. Then, we split the data stratifying on the target variable into a development set (comprising 2/3 of the data) and a test set (consisting of the remaining one-third). Given the class imbalance, we also created alternative development sets by either randomly oversampling the development set or employing SMOTE[22] to generate synthetic samples, excluding the categorical features from the generation process. In both instances, we rebalanced classes so that the minority class in the development set represented 40% of the overall data. These alternative development sets were employed exclusively for training the classifiers with the best hyperparameter configurations. Table 8 summarizes the number of samples belonging to each class in each set. To address class imbalance, we also experimented by assigning class weights to the models that support this parameter, such as Random Forest or Neural Networks.

To fine-tune the hyperparameters of each classifier, we conducted grid searches on the development set. The list of hyperparameters optimized for each classifier is shown in Table 15 in the Appendix. For models with low computational requirements, such as Decision Trees, we employed a stratified 5-fold cross-validation. For other models, we conducted two rounds of stratified splits on the development set, creating a training set (comprising 2/3 of the development set) and a separate validation set in each round. Such a validation schema has been chosen in order to to mitigate the impact of the randomness inherent in certain models and to prevent the estimation of performances becoming dependent on the training-validation splitting. With Neural Networks, we further divided each training set into an internal training set (80%) and an internal validation set (20%) to implement early stopping. For algorithms requiring it, we pre-process the data using Min-Max Scaling. This involved calculating the minimum and maximum values for each feature within each training fold and applying the transformation to the features in the training and in the corresponding test fold, thereby preventing any potential data leakage. For each classifier, we selected the optimal hyperparameter configuration based on the mean of the macro-average F1 scores achieved across the validation folds. We chose not to rely on the accuracy to choose the best model because it is sensitive to unbalanced distribution of classes. Ultimately, we re-trained the classifiers using the best hyperparameter configuration on the entire development set and evaluated their performance on the test set. If required, prior to training, we applied Min-Max scaling using the statistics derived from the training set.

[22] https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

3.4 Evaluating Classifier Performance

During the model selection phase, to determine the best hyperparameters for the models, we generated heatmaps to visualize the performance of the classifiers with different combinations of hyperparameters. This facilitated the understanding of the interactions between the parameters. We also plotted the performance of the model increasing the number of samples in the training set, and we explored the performance of the models by increasing their complexity (e.g., increasing the number of nodes for a Decision Tree). Additionally, we attempted to display the decision boundaries of the classifiers. However, since in none of the feature spaces formed by pairing the features nor in the space of the first principal components, points belonging to the two classes are well separated, we could not visualize them effectively. Finally, we generated the confusion matrix for each classifier's predictions on the test set.

3.5 Classification Results

Table 9 presents the performance on the test set of the classifiers with the best configuration of hyperparameters obtained from the grid search. The corresponding hyperparameters are detailed in Table 16 in the Appendix. The highest Macro-average F1 score (0.716) is achieved by Extreme Gradient Boosting. The model performing worse is Ripper, with an F1 Score Macro Average of 0.598. We observed that even when training the models on randomly oversampled data or augmented data with SMOTE, there is not a significant improvement in performance (see Table 17 in the Appendix).

All the models exhibit a low standard deviation in the Macro-average F1 score on the test set of the cross-validation (Table 18 in the Appendix). The maximum standard deviation is 0.008 for Neural Networks.

Figure 26 compares the macro average of the f1 score

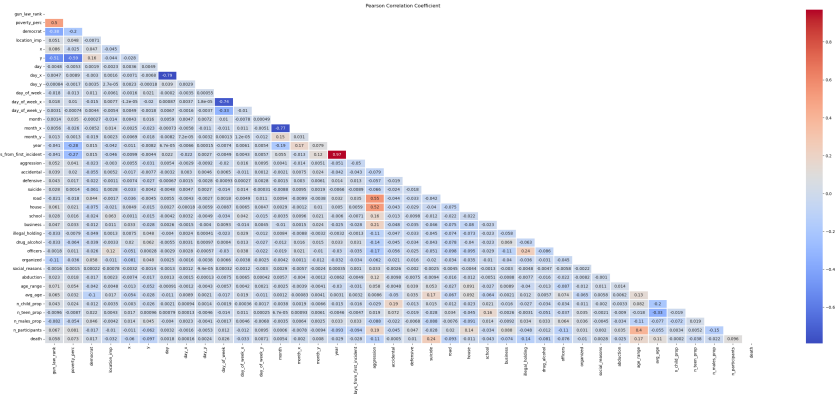


Figure 25: Pearson Correlation Coefficient between the features used for the classification and the target (the variable 'death', located in the last row of the matrix).

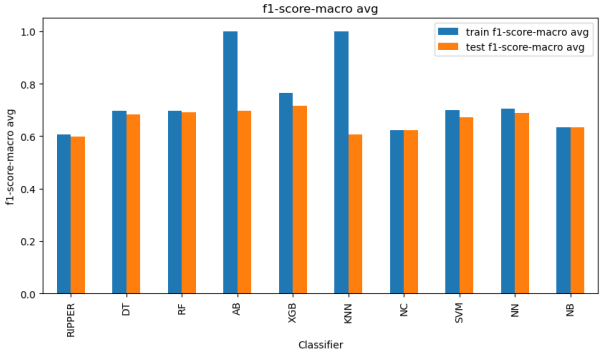


Figure 26: Macro-average F1 scores achieved by the classifiers on both the training and test sets.

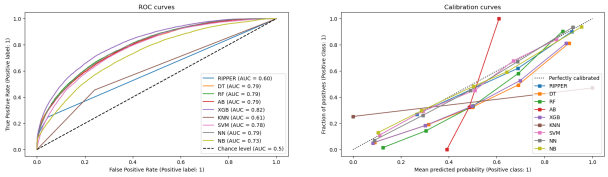


Figure 27: AUROC curves and Calibration curves for the predictions of the classifiers on the test set.

achieved by the classifiers on the training and on the test set. K Nearest Neighbors and Ada Boost exhibit a notable decrease in performance, while the other models demonstrate a more moderate decline.

Figure 27 displays the ROC curves and Calibration curves for the predictions of the classifiers on the test set. The figure includes only classifiers that provide confidence to each prediction, i.e., those implementing the method `predict_proba`. The highest AUROC score is achieved by Extreme Gradient Boosting (which also reports the highest macro average F1 score). All the classifiers demonstrate performance superior to random guessing. The calibration curve allows evaluating the confidence with which a classifier predicts the membership of instances to classes. The predicted probabilities are binned, and for each bin, a point is plotted with the mean of the probabilities in the bin as the x-coordinate and the fraction of instances that were correctly predicted with a probability belonging to that bin as the y-coordinate. A classifier is well-calibrated if its calibration curve does not deviate from the diagonal bisector. The better calibrated models are Support Vector Machine, Neural Networks, Naive Bayes and RIPPER. The worst calibrated models are Ada Boost and K Nearest Neighbors. Ada Boost tends to predict a probability around 0.5 for all instances.

4 Explainability

We have chosen to explain the predictions for two specific incidents in the test set: one with the highest number of individuals killed (9 people) and another where the variable suicide is True but is labeled as Non-Fatal (indicating an attempted suicide). Intuitively, in the first instance models should predict fatality with higher confidence, while in the second case they should be less confident, given that the majority (93.46%) of samples with suicide=1 are labeled as Fatal. The main attributes of these instances are outlined in Table 19 in the Appendix. Additionally, for each model, we selected instances predicted as Fatal with the highest probability and those predicted as Non-Fatal with the highest probability.

4.1 Transparent models

We experimented with two transparent models: Explainable Boosting Machine (EBM) and TabNet (TN). For Explainable Boosting Machine, we employed features listed under the 'Rule-based' column in Table 1, specifying for each feature its type (continuous, nominal or ordinal). For TabNet, we utilized features listed under the 'Distance-based' column. Both the models were trained with their default parameters on the training set. In the case of TabNet the training set was scaled using Min-Max scaling.

The model's performance on both the training and test sets is presented in Table 10. Remarkably, the performance on the test set is comparable to that of opaque models.

Table 9: Performance of classifiers on the test set. Table cells are colored on each column independently. Due to the absence of the predict_proba method in the Nearest Centroid implementation, its AUROC score was not computed.

Classifier	Precision Non-Fatal	Recall Non-Fatal	Precision Fatal	Recall Fatal	F1 Score Macro Avg	Accuracy	Auroc
Ripper	0.729	0.954	0.716	0.249	0.598	0.728	0.603
Decision Tree	0.84	0.693	0.524	0.72	0.683	0.701	0.788
Random Forest	0.842	0.707	0.535	0.718	0.691	0.710	0.794
Ada Boost	0.805	0.807	0.588	0.584	0.696	0.736	0.786
Extreme Gradient Boosting	0.853	0.74	0.569	0.729	0.716	0.736	0.818
K Nearest Neighbors	0.747	0.759	0.471	0.455	0.608	0.662	0.607
Nearest Centroid	0.799	0.629	0.457	0.663	0.623	0.640	nan
Support Vector Machine	0.846	0.655	0.505	0.747	0.671	0.685	0.776
Neural Network	0.782	0.873	0.642	0.483	0.689	0.749	0.792
Naive Bayes	0.749	0.879	0.593	0.375	0.634	0.718	0.734

Table 10: Performance of transparent models on the training and on the test set. Table cells are colored on each column independently.

Classifier	Set	Precision Non-Fatal	Recall Non-Fatal	Precision Fatal	Recall Fatal	F1 score Macro Avg	Accuracy	Auroc
Explainable Boosting Machine	Train	0.779	0.904	0.690	0.455	0.692	0.760	0.804
Explainable Boosting Machine	Test	0.774	0.903	0.681	0.440	0.684	0.755	0.799
TabNet	Train	0.764472	0.920	0.702	0.398	0.671	0.753	0.800
TabNet	Test	0.763325	0.921	0.700	0.394	0.669	0.752	0.795

4.1.1 EBM

Figure 28a illustrates the local explanation of EBM for the attempted suicide incident. The model accurately predicts the incident as Non-Fatal. Notably, the variable `suicide` does not contribute to the prediction, indicating that the model correctly disregarded the information that the incident was a suicide. Less straightforward to interpret is the fact that the feature with the highest positive contribution is `y` (the latitude projection) and that the feature primarily negatively contributing to the prediction is `age_range`. Figure 28b shows the local explanation of EBM for the mass shooting incident. Once again, the model accurately classifies the incident as Fatal. Notably, the feature `suicide` exhibits a significantly higher positive contribution compared to other features. The model seems to capture the pattern that when the number of participants is high and the variable `suicide` is true, it is likely that the incident is Fatal, indicating that the perpetrator killed people and then either killed themselves or attempted to do so. Among other important features are `age_range` and a combination of `n_participants` and `aggression`. Please note that the variable `aggression` is set to 0 because the features `incident_characteristics1` and `incident_characterics_2` lacked sufficient information to assess whether the incident was aggressive. Therefore, it may seem counterintuitive that its value positively contributes to the prediction. However, this may be attributed the distribution of the variable in the training set: the majority of Fatal incident have the variable `aggression` set to 0. Conversely, a feature negatively contributing to the prediction is the location importance, that has a value of 0.33 (in the training set the majority of incidents have location importance equal to 0). Again, this seems counterintuitive, as we might assume that mass shootings frequently occur in squares or public places, implying that when this feature assumes higher values, it should be considered important.

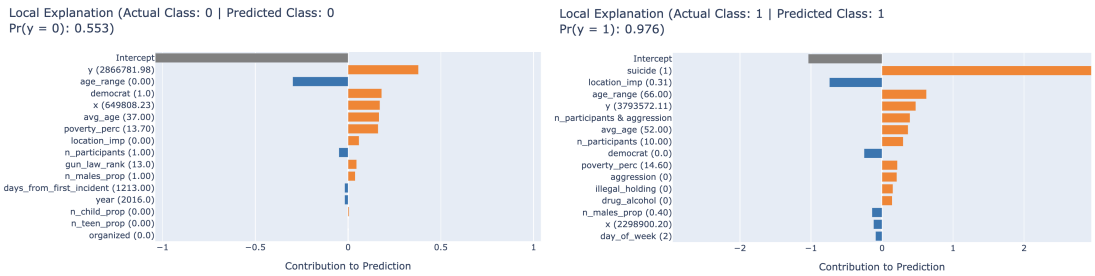
4.1.2 TabNet

Figure 29a and 29b show the local explanation of TabNet for the attempted suicide incident (wrongly classified) and for the mass shooting incident (correctly classified). In both cases the only features that are assigned importances different from 0 are `democrat`, `age_range` and `illegal_holding`. Unfortunately the provided explanation lacks interpretability.

From the analysis of the attention masks of TabNet for the first 50 samples in the test set (Figure 30) we observed that the first mask focuses on the feature `democrat`, `location_imp` and `officers`, the second on `age_range` and the third on `gun_law_rank`, `y` and again `illegal_holding`. Some of these features are expected to be considered important for the prediction task.

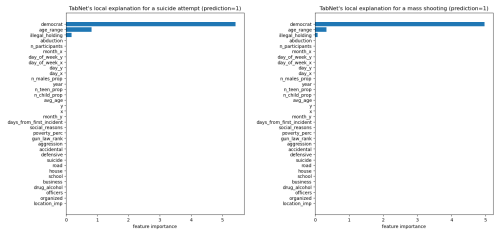
4.2 Global feature importance

For a comprehensive understanding of feature significance, we compared the global feature importance assigned by the classifiers providing these values, namely Decision Trees, Random Forest, Extreme Gradient Boosting, Ada Boost, TabNet and Explainable Boosting Machine. We utilized feature importance values from classifiers trained on the training set with optimal hyperparameters. Figure 43 in the Appendix displays for each feature the feature importance assigned by each classifiers. The features are arranged based on their mean importance values across all classifiers. Please note that not all the models where trained using the same set of features (refer to Table 1) and that Explainable Boosting Machine feature importances do not sum to 1 as for other classifiers. There is limited agreement



(a) (b)

Figure 28: (a) EBM explanation for an attempt suicide. (b) EBM explanation for a mass shooting.



(a) (b)

Figure 29: (a) TabNet's explanation for an attempt suicide. (b) TabNet's explanation for a mass shooting.

among feature importance values. For instance, the feature `age_range` is the most important for Explainable Boosting Machine, TabNet and Decision Tree and it is also among the most important features for Random Forest Classifier. However it is less important for Ada Boost and Extreme Gradient Boosting.

Upon analyzing the centroids computed by the Nearest Centroid classifier, we observed that the features primarily differing in the centroids include `aggression`, `gun_law_rank`, `drug_alcohol`, `illegal_holding`, `suicide`, `business`, `road`, and `officers`. Features such as `age_range`, `avg_age`, `n_teen_prop`, `day_of_week_x`, and `poverty_perc` exhibit slight variations, while the remaining features almost coincide.

4.3 Post-hoc explainers

4.3.1 LIME

To explain predictions, we employed the class `LimeTabularExplainer` with default parameters. This includes an exponential kernel with a width equal to $\sqrt{\text{n\_features}} \times 0.75$, 'auto' mode for feature selection, feature quantile discretization for continuous features, and points sampled from a normal distribution centered on the mean of each continuous feature. For each instance to explain we generated 5000 samples, using the euclidean distance as distance metric to weight points and fitting a Ridge Regressor. We used LIME to explain the predictions of different classifiers, as a sample we report the explanation for the prediction of Extreme Gradient Boosting (XGB), the classifier achieving the best performance. Figure 31a shows the LIME explanation of the XGB prediction for the attempted suicide incident. XGB wrongly classifies the incident as 'Fatal'. As expected, the feature with the highest contribution is `suicide`. Interestingly, in contrast to other explanations, LIME identifies `illegal_holding`, `drug_alcohol`, `aggression` and `abduction` among the features with the highest importance. While for the first three features, we expect that they are likely to be False in the case of a suicide, the importance assigned to the feature `abduction` is less interpretable. If the model assigns importance to it even if the number of participants in the incident is 0, it means that it may not have captured the fact that abduction must involve more than one person. Figure 31b illustrates the LIME explanation of the XGB prediction for the mass shooting incident. This time XGB correctly classifies the incident as 'Fatal'. The five most important features are the same as for the explanation of the prediction for the attempted suicide. As one may expect, `age_range` appears among the features that positively contribute, however the other positively contributing features (`abduction`, `officers`, `business`, `school`, `accidental`, `organized`) are less meaningful. Despite the variable `aggrssion` is set to 0, it is still included among the ones positively contributing to the prediction.

4.3.2 SHAP

We explained predictions of different models using the classes `TreeExplainer` and `KernelExplainer`, depending on the classifier. Figure 32a shows the SHAP explanation for the XGB prediction of the attempted suicide incident. We remind that the model wrongly classifies the instance as Fatal. The feature that mostly positively contributes to the prediction is `suicide`, followed by `y` and `democrat` (equal to 1). Conversely, the `age_range` feature has a negative impact on the prediction. The contribution of the features `suicide` and `age_range` computed through SHAP, align with the contributions provided by LIME. Figure 32b illustrates the SHAP explanation for the XGB prediction for the mass shooting incident. This instance is correctly classified by the model as 'Fatal'. The features with the highest positive contribution are `suicide` (equal to 1), `n_participants` (equal to 10), and `age_range` (equal to 66). SHAP indicates that XGB, like EBM, probably captured the pattern that when the number of partici-

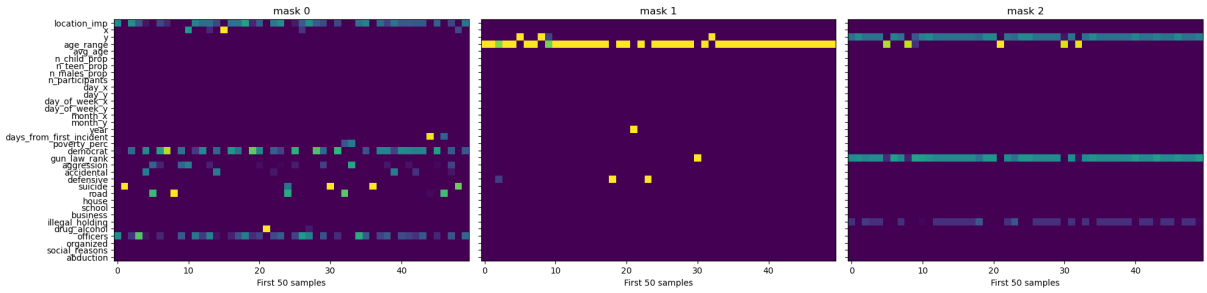
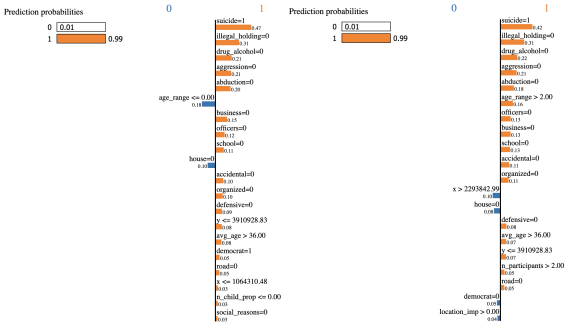
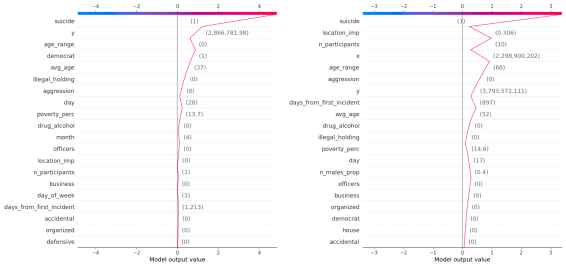


Figure 30: TabNet masks for 50 samples in the test set.



(a) (b)

Figure 31: (a) LIME's explanation for the prediction of the Extreme Gradient Boosting classifier on an incident depicting a suicide attempt. (b) LIME's explanation for the prediction of the Extreme Gradient Boosting classifier on an incident depicting a mass shooting.



(a) (b)

Figure 32: (a) SHAP's explanation for the prediction of the Extreme Gradient Boosting classifier on an incident depicting a suicide attempt. (b) SHAP's explanation for the prediction of the Extreme Gradient Boosting classifier on an incident depicting a mass shooting.

pants is high and the variable `suicide` is true, it is likely that the incident is Fatal, suggesting that the perpetrator killed people and then either killed themselves or attempted to do so. Despite the `aggression` feature being equal to 0, it positively contributes to the prediction, similar to what LIME showed. Another shared observation with LIME is that the `location_imp` feature negatively contributes to the prediction.

Figure 33 illustrates the global feature contributions for the XGB model. Among the most significant features are `age_range`, `aggression`, `y`, and `illegal_holding`. An interesting observation is that the values of several binary features effectively distinguish the samples based on the model's predictions. For instance, when `illegal_holding` is False, the incidents are predicted as Non-Fatal.

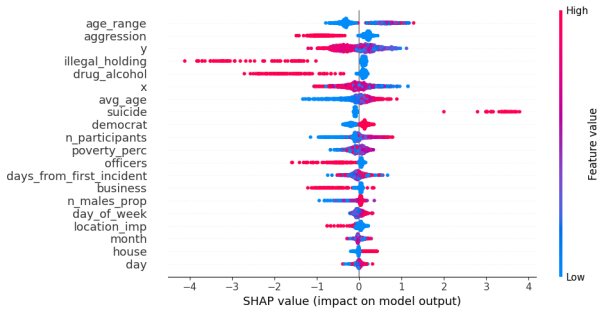


Figure 33: Global feature contributions to the predictions of the XGB model according to SHAP.

4.4 Explainability metrics

To assess the quality of explanations, we employed two metrics, faithfulness and monotonicity, utilizing the implementation from the `aix360` library[23]. The computation of both metrics entails "removing" each attribute considered important by the model by setting its value to its 'base' value. We computed base values as the median value for each feature in the training set. Faithfulness

[23] https://github.com/Trusted-AI/AIX360

scores are displayed in Table 11. The SHAP faithfulness for the explanations of XGB predictions on both the attempted suicide incident and the mass shooting incident is significantly higher than the faithfulness of LIME explanations (0.979 vs. 0.545 for the attempted suicide and 0.918 vs. 0.571 for the mass shooting). This trend of SHAP overperforming LIME on the two instances is consistent across various classifiers. Interestingly, for instances predicted as 'Fatal' with the highest confidence, SHAP's explanations achieve higher scores, whereas for instances predicted as 'Non-Fatal' with the highest confidence, LIME's explanations perform better.

5 Time Series Analysis

5.1 Data preparation

Time series were generated by modeling each city as a sequence of incidents. Specifically, we computed a score for each week over the four years, representing the mean number of participants per incident per week in each city. The weeks were numbered from 0, with the first week having 2 days less than the others.

Prior to creating the time series, incidents with 'NaN' values for `n_participants` (i.e., incidents where the number of participants was unknown) were removed. Additionally, we considered only cities with a number of weeks containing incidents greater than 15% of the total number of weeks over the four years. This led to the creation of a dataset comprising 588 time series, each comprising 209 data points.

To facilitate a more in-depth exploration of the cluster analysis on the upcoming time series, we constructed a comprehensive dataset that includes various features extracted from the available data for each city. The features extracted for each city include demographic and incident-related information, such as population, the number of incidents occurrences in the city across all the years in which the analysis was conducted, and the number of weeks in which at least one incident occur. Additionally, we discretized the features into quantiles based on their distribution across all time series. This approach facilitates easier comparison among the clusters. A more detailed description of the features can be found in Table 20 in the Appendix.

5.2 Clustering analysis

5.3 Shape-based

We employed the K-Means clustering algorithm on time series data, utilizing the TimeSeriesScalerMeanVariance object from the tslearn library for scaling purposes.

The clustering procedure was executed using the TimeSeriesKMeans method from the tslearn library, employing the Dynamic Time Warping (DTW) distance as the metric. We aimed to create 10 clusters (k = 10) and set the maximum number of iterations to 100.

It's worth noting that the use of Dynamic Time Warping as a metric involves longer computations compared to the Euclidean distance, given their distinct time complexities. The Euclidean distance exhibits linear complexity, while DTW does not. Despite the increased computational cost, our preliminary analysis suggests that the use of DTW yields superior results. This rationale justifies our preference for DTW over the Euclidean distance for the clustering.

The population of the states to which the cities belong appears to be uniformly distributed across the clusters. Cluster 5 is characterized by elevated values in the time series, indicating a higher frequency of incidents and involving a larger number of people. This cluster includes only time series related to cities where the total number of participants in incidents, along with all other features in which at least one incident occurred, fall into the third quantile of the dataset.

Cluster 1 appears to encompass time series with higher values, although the distinction is somewhat less pronounced compared to those in Cluster 5.

In contrast, Cluster 9, the most populated cluster (137 out of 588), exhibits lower values, suggesting that these cities likely experience numerous weeks with zero incidents, resulting in an overall lower incident count. Within Cluster 9, the majority of cities are characterized by the number of participants in incidents and the number of weeks in which at least one incident occurred falling into the first quantile.

Cluster 6 stands out for its time series displaying considerable variability and numerous fluctuations. Notably, the majority of time series in Cluster 6 are associated with cities where both the average number of participants and fatalities exceed the overall dataset average.

Cluster 4 exhibits a uniform distribution for all the city features considered.

Clusters 1, 3 and 8 predominantly have features with values above the threshold set by the second quantile.

We also performed the same analysis using time series generated in the same manner as the previous ones, but this time using the `n_killed` variable. From the Sankey plot, we observe that there is no apparent correlation between the cluster analyses performed on the two distinct time series. We also observed that clusters 3 and 9 in this analysis contain features with values that surpass the dataset's average, corresponding to the cities in Cluster 5 from the previous analysis.

5.4 Compression-based

We apply clustering to the compressed representations of time series data using both Agglomerative Clustering and KMeans.

5.4.1 Agglomerative Clustering

We conduct hierarchical clustering on compressed time series data using the AgglomerativeClustering algorithm from scikit-learn.

Data compression is achieved by calculating pairwise distances between rows of the matrix, representing the dissimilarity measure on data encoded using `UTF-8` and then compressed using `zlib` compression.

Table 11: Faithfulness for the LIME and SHAP explanations on different samples.

Sample	DT		RF		XGB		NN		SVM	
	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
Attempted Suicide	0.638216	0.963508	0.604192	0.964190	0.545138	0.979389	-0.049704	0.130451	-0.049129	-0.235384
Mass shooting	0.734065	0.977253	0.682913	0.964476	0.571016	0.918252	-0.212902	-0.037200	-0.037180	0.298565
Fatal with highest confidence for DT	-0.143784	0.762704	0.249907	-0.801124	0.189934	-0.558125	0.050865	-0.442505	0.062525	-0.023103
Fatal with highest confidence for RF	0.715922	0.980199	0.676820	0.971303	0.590276	0.964750	-0.175269	0.126164	-0.133346	0.169740
Fatal with highest confidence for XGB	0.680731	0.966861	0.664125	0.963553	0.599138	0.953075	-0.149152	0.047637	-0.151222	-0.230260
Fatal with highest confidence for NN	0.661191	0.977253	0.695583	0.964476	0.560756	0.918252	-0.219692	-0.033473	-0.019645	0.304121
Fatal with highest confidence for SVM	0.048304	0.349155	0.017421	0.128338	0.032596	0.779718	-0.059077	0.449277	-0.078551	-0.409537
Non-Fatal with highest confidence for DT	0.380304	-0.929084	0.483481	-0.957459	0.412477	-0.965275	0.157702	-0.226454	0.150209	-0.081196
Non-Fatal with highest confidence for RF	0.368766	-0.794060	0.495323	-0.951173	0.388899	-0.955820	0.144798	-0.181697	0.210072	-0.143239
Non-Fatal with highest confidence for XGB	-0.391588	0.697630	0.541675	-0.877779	0.431065	-0.887615	-0.016043	0.074321	0.168056	-0.111436
Non-Fatal with highest confidence for NN	0.304621	-0.711646	0.432901	-0.812596	0.436162	-0.908356	0.072044	-0.018091	-0.089336	0.051093
Non-Fatal with highest confidence for SVM	0.113793	-0.840387	0.139607	-0.773380	0.139134	-0.853630	0.051220	-0.052365	0.097364	-0.186853

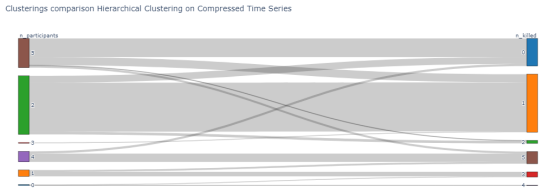


Figure 34: Sankey diagram of the labels assigned by Hierarchical Clustering algorithm on compressed representations of time series data reppresenting the mean number of participants and killed person per incident per week in each city

We perform a preliminary analysis by evaluating different linkage methods, and the dendrogram inspection was employed to guide the selection of the most suitable number of clusters. We choose average linkage with 4 clusters.

The hierarchical clustering algorithm provided varied outcomes when grouping similar time series derived from the `n_killed` variable. Using 4 clusters resulted in an imbalanced distribution, with the majority of cities assigned to Cluster 1, indicating a lack of diversity. However, with 6 clusters, the distribution improved, revealing specific patterns.

Clusters 0, 1, and 4 consist of cities where features fall within the third quartile concerning the total weeks of incidents in the dataset. Cluster 2 emerges as the most populated cluster, with features distributed across the first three quantiles. Cluster 5 encompasses cities with features predominantly falling into the second and third quantiles. Cluster 3, which includes only 4 cities, exhibits features exclusively in the first quantile.

This is the only clustering algorithm in which the results reveal a pattern between the clusters formed using time series generated based on the number of participants and those formed using the number of killed people, as Figure 34 illustrates.

5.4.2 TimeSeriesKMeans on Piecewise Aggregate Approximation data

In our analysis, we applied the TimeSeriesKMeans clustering algorithm to time series data after compressing it using the PiecewiseAggregateApproximation method from the tslearn library. The results from this clustering do not revealed a discernible pattern relative to the population.

Clusters 1, 2, 3, and 6 predominantly exhibit features that with values falling between the second quantile and the third quantile. Notably, Clusters 2 and 3 show a high degree of similarity, each containing 10 cities. On the other hand, Clusters 4 and 0 exhibit features distributed across the first three quantiles. Notably, Cluster 4, being the most populous, is characterized by predominantly first quantile features.

5.5 Feature-based

In this approach, we employed the KMeans method from the sklearn library to train a model on a set of features containing statistical values for each time series. These features include the average of the values, standard deviation, percentiles, median, interquartile range, coefficient of variation, skewness, and kurtosis.

The default hyperparameters for K-Means, as previously defined in section 2.3.1, were utilized, with k = 11.

This clustering algorithm outperforms all other tested time series clustering algorithms. Table 12 presents the scores and cluster sizes for further evaluation.

From the clustering analysis, cities appear to be grouped into 11 clusters of non-uniform dimensions. The largest cluster is Cluster 4, which contains 171 elements, while two clusters, namely Cluster 8 and Cluster 9, each contain only one element.

Studying the clustering distribution, we can conclude that there is no discernible pattern in cluster division based on population size or geographical location of the cities.

Clusters 0, 6, 8, 9, and 10 are primarily composed of cities exhibiting features (total number of incidents, number of weeks in which at least one incidents is present, sum of the number of killed person, females, males, individuals injured and arrested for all the incidents occurred in a city) predominantly situated in the third quantile. The mean values of their respective time series consistently surpass others across all time points, signifying time series characterized by fewer zero values. In essence, these clusters encapsulate cities where incidents occur in most weeks.

Clusters 8 and 9 each consist of a single city: Chicago,

Table 12: KMeans Feature-Based clustering scores and cluster sizes.

SSE	BSS	Calinski-Harabasz	Davies-Bouldin	Silhouette	Cluster sizes
120.493	2032.009	973.058	0.669	0.581	0:25; 1:113; 2:37; 3:45; 4:171; 5:117; 6:17; 7:26 ; 8:1; 9:1; 10:35;

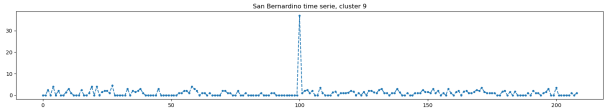


Figure 35: San Bernardino (California) time serie.

Illinoise, and San Bernardino, California, respectively. Despite attempts to reduce the number of clusters to 9 and eliminate the cluster with only one city, this adjustment proved unsuccessful, as the two clusters with a singular city persisted.

San Bernardino time series displays a peak assuming a value of 37, the highest reached in all the time series, representing the maximum number of participants in incidents within a specific week in any city throughout the entire dataset. Excluding this value, the time series exhibits a significant number of null values (94 out of 209). The mean of its values is 1.125. If we disregard the value 37, the series would share characteristics similar to those belonging to Cluster 4. The peak value in the time series of San Bernardino, observed in the 100th week of the dataset (from November 30th to December 6th, 2015), is associated with a significant incident known as the San Bernardino attack. This was a terrorist attack that occurred on December 2nd, 2015, involving a mass shooting and an attempted bombing. A total of 37 people participated in the incident, and excluding the two perpetrators, all participants were either injured or killed[24].

Chicago stands out with a time series entirely devoid of null values. The only other cities sharing this characteristic and their respective clusters are as follows: City of New York (New York), Columbus (Ohio), Milwaukee (Wisconsin) and New Orleans (Lousiana), all this cities are clustered into Cluster 6. The time series of these cities are represented in Figure 36.

Chicago's time series is more regular, with respect to the series belonging to Cluster 6, with values ranging between 1.2 and 2.2, presenting a relatively constant trend over time. In contrast, time series in Cluster 6 exhibit a more irregular pattern, covering a broader range of values between 1 and 4.5.

Cluster 6 encompasses 17 time series, characterized by a range in the cardinality of non-zero value ranging from 201 and 209 (out of 209). Cluster 0 contains time series with characteristics similar to those in Cluster 6. However, the number of non-zero values is slightly lower, ranging from 201 to 209 (out of 209).

[24] https://en.wikipedia.org/wiki/2015_San_Bernardino_attack

Figure 36: Chicago (Illinoise), City of New York (New York), Columbus (Ohio), Milwaukee (Wisconsin) and New Orleans (Lousiana) time series.

Clusters 2 and 7 consist of cities associated with features primarily falling above the second quantile. With the exception of the number of fatal incidents, these cities experience a higher-than-average number of incidents throughout the years, with a higher number of participants per incident. However, the average number of incidents with at least one fatality is lower on average compared to cities clustered in Clusters 0, 6, 8, and 10. The main distinction between the two clusters lies in the number of non-zero values. For Cluster 2, this number ranges between 89 to 104, with a mean of 94.054. On the other hand, for Cluster 7, it ranges between 107 to 134 (out of 209), with a mean of 118.731.

Clusters 1 and 4, the second and first most populated clusters, predominantly include cities with features mostly belonging to the first two quantiles. Specifically, in Cluster 4, cities mostly have features in the first quantile, indicating time series associated with many null values.

Cluster 3 contains features predominantly in the third quantile, and it is the third-largest cluster.

Cluster 5 features predominantly fall within the second and third quantiles, making it the third-largest cluster.

5.6 Motifs and anomalies extraction

We used the `matrixprofile-ts` library[25] for evaluating time series data using the Matrix Profile algorithm. We find optimal window size using the Highest Autocorrelation method from `claspy` library[26].

The optimal window size was determined for all the series, yielding a mean value of 39.15 with a standard deviation of 24.62. The minimum optimal window size observed was 10, while the maximum reached 103.

The matrix profile computation was utilized for tasks involving the extraction of motifs and anomalies. Specifically, the motifs method and discords method were employed, both of which are available in the `matrixprofile-ts` library.

We focused our analysis on the two series belonging to clusters 8 and 9.

The optimal window size for San Bernardino corresponds to 79 weeks. Extracting motifs with this parameter resulted in a single motif being detected twice, covering the time periods from May 19, 2014, to November 29, 2015, and from October 12, 2015, to April 23, 2017. In the weeks common to both motifs, only null values are associated, indicating no incidents occurred. Overlaying the motifs onto the San Bernardino attack, which happened in week 100, corresponds to week 27 when an incident took place on July 7, 2014. In this incident, 7 participants were involved, exceeding the average number of participants per incident for San Bernardino, which is 4.5. The value assumed by the time series at that point is the maximum, excluding the value 37 recorded in week 100. The time series associated with the San Bernardino city, along with the highlighted motifs in red, are illustrated in Figure 37. Additionally, we

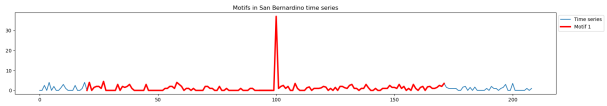


Figure 37: Motifs extraction for San Bernardino time serie, window size = 79.

observe that by reducing the window size, the detected motifs no longer include the value corresponding to the 100th week. However, recurring patterns still emerge in the earlier part of the time series, in the first 90 values. Figure 38 represents the time series associated with the San Bernardino city (in light blue) along with the highlighted motifs. The algorithm identifies four distinct motifs, each repeated twice. Chicago time series have an optimal window size of 11 weeks. The motif extraction reveals 5 different motifs distributed across all time series. The anomalies are detected throughout the entire time series, with some anomalies occurring at points where motifs were previously identified. These anomalies do not seem to be tied to specific incidents or weeks with a notable frequency of events, as the series

[25] https://github.com/target/matrixprofile-ts
[26] https://github.com/ermshaua/claspy

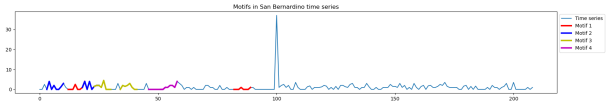


Figure 38: Motifs extraction for San Bernardino time serie, window size = 8.

Figure 39: Motifs extraction for Chicago time serie, window size = 11.

takes values within a limited range, suggesting a more subtle pattern in the time series data. Anomalies and motifs detected in the Chicago time series are reported in Figures 39 and 40.

5.7 Shapelets extraction and classification

We defined the binary variable `is_killed` for each city time series. A time series related to a city is labeled as True if the ratio of fatal incidents in the city between 2014 and 2017 is above the second quantile of the distribution of `fatal_incidents_ratio` related to all time series; otherwise, it is labeled as False. This variable is independent of the number of null values present in the time series, posing a challenging classification task.

For the shaplet classification task, we divided the dataset into training and test sets, with the test set comprising 20% of the total data. Stratification was applied based on the clustering memberships obtained using the feature-based K-Means clustering algorithm. This ensures a representative distribution of clusters in both the training and test sets and also leads to a balanced distribution with respect to the `is_killed` variable in both the training (49.79% of True labels) and test sets (50% of True labels).

In order to classify the extracted shaplets into the defined class based on the `is_killed` variable, we employed three different models: ShapletModel, Decision Tree and K Nearest Neighbors.

ShapletModel Classifier, from the tslearn library, both extracts shaplets from time series and also performs the classification task. In Figure 41, the extracted shaplets are depicted, with 4 shaplets of length 30. These shaplets exhibit relatively linear patterns, except for one that displays a significant peak, resembling the pattern observed in the time series of San Bernardino.

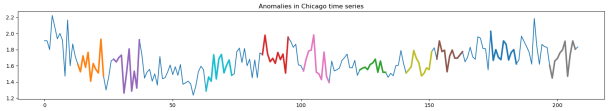


Figure 40: Anomalies extraction for Chicago time serie, window size = 11.

Table 13: Parameters and performance of shaplets classifiers and time series RocketClassifier on test set.

Classifier	Parameters	Precision False	Recall False	Precision True	Recall True	F1 score macro avg	Accuracy
ShapletModel	`batch_size=16, max_iter=200, weight_regularizer=0.01`	0.786	0.373	0.589	0.898	**0.609**	**0.636**
DecisionTreeClassifier	`criterion='entropy', max_features='log2', min_samples_split=0.02`	0.607	0.627	0.614	0.593	0.610	0.610
KNeighborsClassifier	`algorithm='brute', weights='distance'`	0.582	0.542	0.571	0.610	0.576	0.576
RocketClassifier	`num_kernels=500`	0.564	0.525	0.556	0.593	0.559	0.559

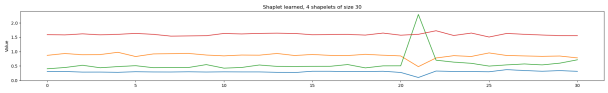


Figure 41: Extracted shaplets from the time series data using ShapletModel.

Additionally, we attempted the classification task using two classifiers from the sklearn library: Decision Tree and K Nearest Neighbors. These classifiers were employed for the classification task on the previously extracted shaplets. For shaplet extraction, we employed the LearningShapelets module from the tslearn library. The extraction process resulted in four shaplets of size 31 and four of size 62. The model for shaplet extraction was trained using a batch size of 64, incorporating L2 regularization with a coefficient of 0.001. The optimization process utilized the Adam optimizer from the Keras library with a learning rate set to 0.01. This approach allowed us to explore different classification models and assess their effectiveness in handling the task.

Figure 42 represents the extracted shaplets from the time series data. The shaplets exhibit a irregular but generally linear trend, suggesting the presence of recurrent patterns in the time series. The lack of a clear growth trend in the shaplets and the presence of local minima and maxima within the same range indicates a sort of stability in the salient features of the time series.

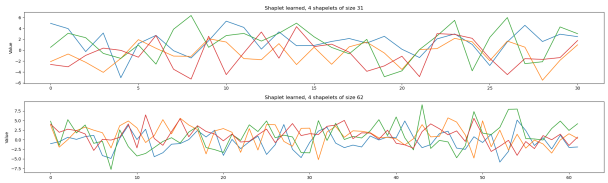


Figure 42: Extracted shaplets from the time series data using the LearningShapelets model.

Table 13 provides details on the parameters and classification performance of the classifier. For parameters that are not specified, default values were utilized.

The ShapletModel exhibited the highest F1 score (macro avg) and accuracy among the three models, with a value of 0.609 and 0.636, respectively. However, all models demonstrated a relatively balanced performance, with trade-offs between precision and recall for both classes. Importantly, it is observed that the Decision Tree Classifier assigns higher feature importance to the shaplets of larger size. This could suggest that longer and potentially more complex patterns increased discriminatory power in the classification process.

Table 14: Comparison of Macro F1 Averages Between Labels obtained by Different Classifiers.

	ShapeletModel	DecisionTree	KNN
DecisionTree	0.667		
KNN	0.693	0.717	
Rocket	0.719	0.617	0.635

We also trained a RocketClassifier model, a kernel-based time series classifier from sktime library, on all time series related to the shaplets in the training set. Subsequently, we tested the model on the time series of shaplets from the test set. This analysis aims to compare the classification results obtained using shaplets with those derived from the same task using the entire time series.

It is noticeable that the classification on the entire time series achieves slightly lower performance compared to all the shaplet classifiers (results are reported in Table 13).

Comparing the labels assigned by various models, we observe that RocketClassifier and ShapletModel classified a higher number of data points similarly, this highlights consistency in predictions between a model based on random convolutions and ShapletModel.

Also Decision Trees and K Nearest Neighbors classifiers show consistency in predictions. This result was expected, as both models utilized the same set of shaplets. The scores of Macro F1 Averages Between Labels obtained by different classifiers for the time series-shaplet classification task are presented in Table 14.

Appendix

Table 15: Hyperparameter values explored in the grid search. The unreported hyperparameters were set to their default values.

Classifier	Hyperparameter	Values
DT	criterion	entropy, gini
	class_weight	balanced, None
	min_samples_split	2, 0.01, 0.025, 0.025, 0.05, 0.1, 0.25, 0.5
	min_samples_leaf	1, 0.01, 0.025, 0.05, 0.1
RF	n_estimators	100, 200
	max_features	sqrt, None
	max_samples	0.5, None
	min_samples_split	2, 0.01, 0.025, 0.05, 0.1
	min_samples_leaf	1, 0.01, 0.025, 0.05, 0.1
	bootstrap	True
	class_weight	balanced
	criterion	entropy
Ada Boost	estimator	None, DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=0.01, class_weight='balanced')|
	n_estimators	200, 500, 1000
	learning_rate	1, 0.9, 0.8
	algorithm	SAMME, SAMME.R
XGB	eta	0.3
	min_child_weight	1, 0.01, 0.025, 0.05, 0.1
	subsample	0.5, 1
	scale_pos_weight	1, n_neg / n_pos
	colsample_bytree	1
	colsample_bylevel	1
	colsample_bynode	1, sqrt(n_features)/n_features
	max_depth	4, 6, 8
KNN	n_neighbors	1, 3, 5, 9, 15
	weights	uniform, distance
	algorithm	brute
	metric	minkowski
	p	1, 2
NC	metric	euclidean, manhattan
SVM	kernel	poly, rbf
	C	0.1, 1, 2, 3.5, 5
	gamma	scale, auto
	degree	3, 4
	class_weight	balanced
NN	hidden_layer_sizes	[256, 256, 256], [128, 512, 128], [128, 128]
	dropout_rate	0.25, 0.325, 0.5
	output_bias	log(n_pos/n_neg), 0
	loss	BinaryCrossEntropy
	layer	Dense
	hidden_activation_fun	relu
	final_activation_fun	sigmoid
	optimizer	adamax
	learning_rate	0.01, 0.001, 0.0005
	epochs	125
	batch_size	1024, 2048
NB	alpha	0.5
	var_smoothing	1e-09

Table 16: Optimal hyperparameters for classifiers.

Classifier	Params
Ripper	{'k': 2, 'prune_size': 0.5}
Decision Tree	{'class_weight': 'balanced', 'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 0.01, 'random_state': 42}
Random Forest	{'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'sqrt', 'max_samples': None, 'min_samples_leaf': 1, 'min_samples_split': 0.01, 'n_estimators': 200, 'n_jobs': -1, 'random_state': 42}
Ada Boost	{'algorithm': 'SAMME', 'estimator': DecisionTreeClassifier(class_weight='balanced', min_samples_split=0.01), 'learning_rate': 0.8, 'n_estimators': 200, 'random_state': 42}
Extreme Gradient Boosting	{'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 1, 'eta': 0.3, 'max_depth': 6, 'min_child_weight': 1, 'scale_pos_weight': 2.1236912156166814, 'subsample': 1}
K Nearest Neighbors	{'algorithm': 'brute', 'metric': 'minkowski', 'n_neighbors': 1, 'p': 2, 'weights': 'uniform'}
Nearest Centroid	{'metric': 'euclidean'
Support Vector Machine	{'C': 3.5, 'class_weight': 'balanced', 'degree': 4, 'gamma': 'scale', 'kernel': 'poly', 'probability': True}
Neural Network	{'batch_size': 1024, 'epochs': 125, 'model_dropout': 0.5, 'model_final_activation_fun': 'sigmoid', 'model_hidden_layer_sizes': (256, 256, 256), 'model_output_bias': 0, 'optimizer': 'adamax', 'optimizer_learning_rate': 0.001}
Naive Bayes Mixed	{'alpha'=0.5, 'var_smoothing'=1e-09}

Table 17: Classifier performance on the test set when trained on the original development set, a randomly over-sampled development set, and a SMOTE-augmented development set. Table cells are colored on each column independently. Due to the absence of the predict_proba method in the Nearest Centroid implementation, its AUROC score was not computed.

Classifier	Precision Non-Fatal	Recall Non-Fatal	Precision Fatal	Recall Fatal	F1 Score Macro Avg
Ripper (Original)	0.729000	0.954000	0.716000	0.249000	0.598000
Ripper (Oversampled)	0.774000	0.847000	0.592000	0.474000	0.667000
Ripper (SMOTE)	0.737000	0.942000	0.698000	0.285000	0.616000
Decision Tree (Original)	0.840000	0.693000	0.524000	0.720000	0.683000
Decision Tree (Oversampled)	0.836000	0.701000	0.527000	0.707000	0.683000
Decision Tree (SMOTE)	0.815000	0.764000	0.557000	0.631000	0.690000
Random Forest (Original)	0.842000	0.707000	0.535000	0.718000	0.691000
Random Forest (Oversampled)	0.842000	0.707000	0.536000	0.718000	0.691000
Random Forest (SMOTE)	0.819000	0.775000	0.571000	0.637000	0.699000
Ada Boost (Original)	0.805000	0.807000	0.588000	0.584000	0.696000
Ada Boost (Oversampled)	0.805000	0.809000	0.591000	0.584000	0.697000
Ada Boost (SMOTE)	0.802000	0.821000	0.599000	0.569000	0.698000
Extreme Gradient Boosting (Original)	0.852000	0.735000	0.564000	0.728000	0.712000
Extreme Gradient Boosting (Oversampled)	0.879000	0.645000	0.51900	0.811000	0.688000
Extreme Gradient Boosting (SMOTE)	0.858000	0.719000	0.556000	0.747000	0.710000
K Nearest Neighbors (Original)	0.747000	0.759000	0.471000	0.455000	0.608000
K Nearest Neighbors (Oversampled)	0.747000	0.759000	0.471000	0.455000	0.608000
K Nearest Neighbors (SMOTE)	0.754000	0.741000	0.470000	0.487000	0.613000
Nearest Centroid (Original)	0.799000	0.629000	0.457000	0.663000	0.623000
Nearest Centroid (Oversampled)	0.797000	0.633000	0.458000	0.658000	0.623000
Nearest Centroid (SMOTE)	0.809000	0.601000	0.452000	0.698000	0.619000
Support Vector Machine (Original)	0.846000	0.655000	0.505000	0.747000	0.671000
Support Vector Machine (Oversampled)	0.847000	0.654000	0.505000	0.749000	0.670000
Support Vector Machine (SMOTE)	0.827000	0.706000	0.524000	0.687000	0.678000
Neural Network (Original)	0.782000	0.873000	0.642000	0.483000	0.689000
Neural Network (Oversampled)	0.782000	0.874000	0.644000	0.483000	0.689000
Neural Network (SMOTE)	0.775000	0.895000	0.668000	0.447000	0.683000
Naive Bayes Mixed (Original)	0.749000	0.879000	0.593000	0.375000	0.634000
Naive Bayes Mixed (Oversampled)	0.778000	0.799000	0.547000	0.515000	0.659000
Naive Bayes Mixed (SMOTE)	0.781000	0.79700	0.549000	0.524000	0.663000

Table 18: Standard deviation in the Macro-average F1 score on the validation set of the cross-validation.

Classifier	Std
Ripper	0.002218
Decision Tree	0.004611
Random Forest	0.004239
Ada Boost	0.003758
Extreme Gradient Boosting	0.005298
K Nearest Neighbors	0.003557
Nearest Centroid	0.003802
Support Vector Machine	0.001087
Neural Network	**0.008270**

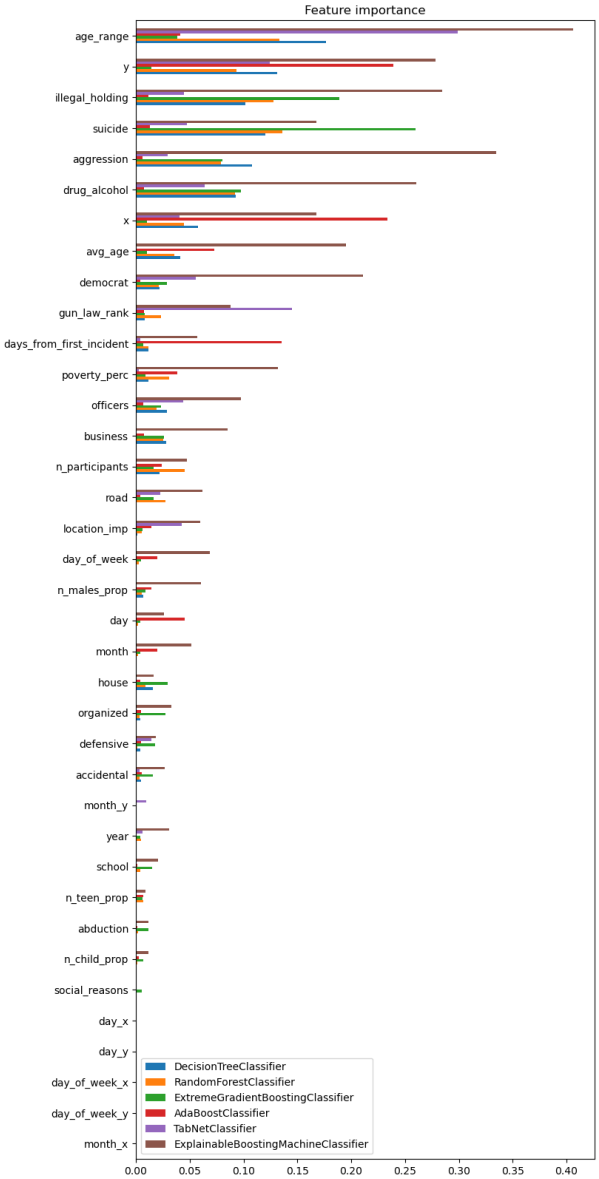


Figure 43: Global feature importance across classifiers.

Table 19: Features of the explained records (a mass shooting and an attempted suicide).

Feature	Attempted suicide	Mass shooting
date	2016-04-28	2015-06-17
state	Texas	South Carolina
city	Brownsville	Charleston
min_age	37	21
max_age	37	87
n_child	0	0
n_teen	0	0
n_adult	1	10
n_males	1	4
n_females	0	6
n_killed	0	9
notes	Perp escapes from prison and steals a shotgun from his former boss and later shoots himself.	at least 8 dead, more inj Black church, white shooter. 12:50am Mayor says 9 killed, 1 injured Rev. Pickney was a SC senator, tried to kill self, out of ammo
incident_characteristics1	Shot - Wounded/Injured	Shot - Dead (murder, accidental, suicide)
incident_characteristics2	Suicide - Attempt	Suicide - Attempt
poverty_percentage	13.7	14.6
democrat	1	0
gun_law_rank	13	13

Table 20: Definition of the features defined for each city for the time series task. These features cover demographic details, incident-related statistics, and quantiles derived from the distribution across all time series. The quantiles enable a categorical representation, facilitating easier comparison and interpretation of the cluster analyses.

Feature	Description	Average	Std Dev
population_state	The mean population of the state to which the city belongs.	1.182544e+07	1.031046e+07
n_incidents	Total number of incidents occurrences in the city across all years.	200.415	525.432
n_participants	Total number of participants in all incidents across all years.	360.573129	900.945
n_weeks_with_incidents	Total number of weeks with incidents across all years.	78.629	53.963
n_killed	Total number of killed across all years.	53.602	130.667
n_females	Total number of females across all years.	35.694	81.854
n_males	Total number of males across all years.	280.204	752.659
n_injured	Total number of injured across all years.	124.760	471.883
n_arrested	Total number of arrested across all years.	84.398	123.678
fatal_incidents	Total number of fatal, incidents in which at least una person died, incidents across all years.	49.214	121.574
fatal_incidents_ratio	Ratio of fatal incidents to n_incidents.	0.244	0.140
population_quantile	Quantile of population_state derived from the distribution across all time series.		
n_incidents_quantile	Quantile of n_incidents derived from the distribution across all time series.		
n_weeks_quantile	Quantile of n_weeks_with_incidents derived from the distribution across all time series.		
n_participants_quantile	Quantile of n_participants derived from the distribution across all time series.		
n_killed_quantile	Quantile of n_killed derived from the distribution across all time series.		
n_females_quantile	Quantile of n_females derived from the distribution across all time series.		
n_males_quantile	Quantile of n_males derived from the distribution across all time series.		
n_injured_quantile	Quantile of n_injured derived from the distribution across all time series.		
n_arrested_quantile	Quantile of n_arrested derived from the distribution across all time series.		
fatal_incidents_quantile	Quantile of fatal_incidents derived from the distribution across all time series.		