Data Mining Project Report

Group 6

A.Y. 2023/2024

Funaioli Francesca 563519
Karoui Hamza 543916
Mitola Francesco 655383
Vezzuto Samuele 656132

Introduction

This report goal is to describe all procedures and methodologies adopted to perform data analysis on a dataset divided in three csv files, *incidents.csv*, *povertyByStateYear.csv*, *year_state_district_house.csv*, which respectively contain information about gun incidents in the USA, about the poverty percentage for each USA state and year, about the winner of the congressional elections in the USA for each year, state and congressional district.

The report is organized as follows: section 1 is dedicated to Data Understanding, Data Integration and Distribution Analysis of each attribute of each dataset; section 2 described Data Preparation techniques used to fill missing values or deal with outliers identified during Data Understanding, and the definition of new indicators useful for Clustering Analysis; section 3,4 and 5 discuss respectively Clustering, Predictive and Explaination Analysis.

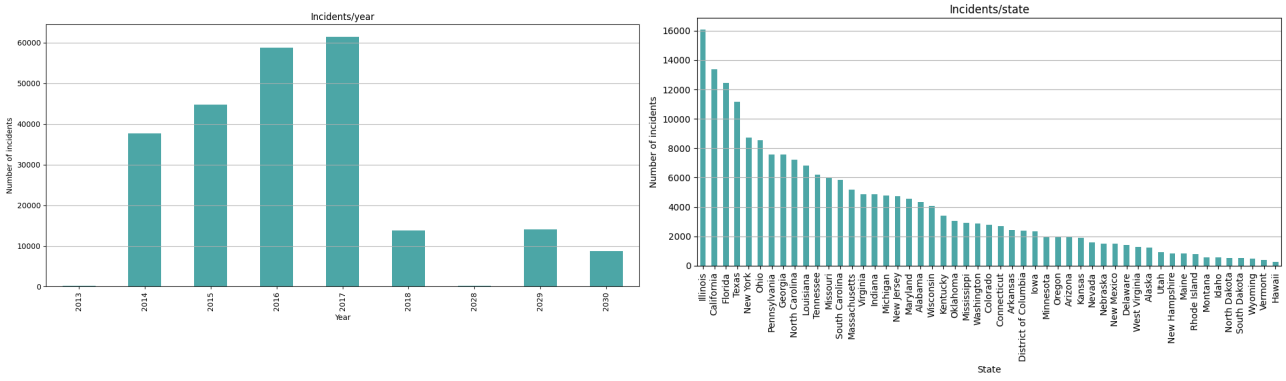
1 Data Understanding

1.1 Datasets Analysis

The data understanding phase aims to analyze the dataset and to gain information on the general properties of our data. This phase of data analysis is focused on identification of missing values, outliers, duplicates, data integration and distribution analysis. The three csv files, `incidents.csv`, `povertyByStateYear.csv` and `year_state_district_house.csv` were imported and analyzed separately. Firstly, some preliminary checks were performed, looking for missing values and checking the data types of the attributes, then a more in depth column-wise analysis of each dataset was performed. As preliminary checks, we computed the statistics of the numerical attributes of each dataset and we noticed a wrong value in attribute *participant_age1* (value of 311). We also found out that some of the columns, considered numeric, contained non-numerical values, as they were not displayed when the pandas `describe()` method was used. The value of 0 for attribute *participant_age1* occurs when there are infants involved and therefore it is not to be considered a wrong value. In a similar way, *n_participants* has value 0 if the incident had no casualties and no shooting, but also in records with most of the attributes set to null. In the following parts of this section, the more in depth column-wise analysis will be discussed.

1.1.1 Incidents

Date:
Each *date* value was converted to a datetime object, while checking that there were no null or NaT values. The plot in Figure 1a, representing the number of incidents per year, shows an increasing trend in the number of gun crimes over the years. It also shows a lack of records related to years 2013 and 2018: there are only 253 incidents happened in 2013, hence they were removed. As for the year 2018, there is only data relating to the first three months of the year in the dataset, so records from year 2018 were deleted from the dataset.
Geographical information:
The *state* attribute contains no null values and it has 51 unique values: the 50 states of the United States and the District of Columbia, which we considered as a special state. Figure 1b shows the number of incidents recorded in each state in decreasing order: the states with the highest number of incidents are, in order, Illinois, California and Florida.



(a) (b)

Figure 1: Number of incidents per year (a) and per state (b).

The *city_or_county* attribute has no null values, but it often contains additional informations about the suburb or the neighborhood in brackets, that provide a more precise location for the incident, e.g. "Minneapolis (Brooklyn Center)". Figure 2 shows the number of incidents recorded in each city/county in decreasing order: the top 3 cities per number of incidents are Chicago (Illinois), Baltimore (Maryland) and New Orleans (Louisiana).

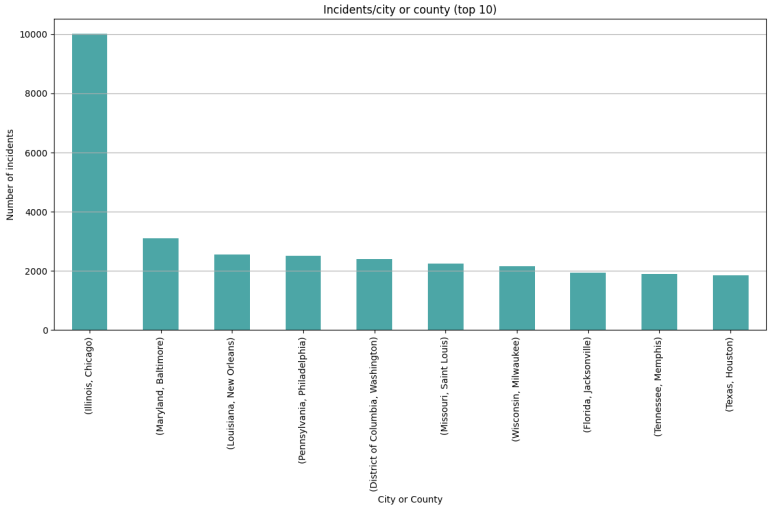


Figure 2: Number of incidents per city or county.

The *address* attribute contains some null values, but we believe it does not hold any statistical value, given that more specific information about the exact location of an incident is found by using the geographical coordinates. Furthermore, there are only 6020 records for which address information can not be inferred using *latitude* and *longitude* attributes, so we will remove this column during the Data Preparation.

The *latitude* and *longitude* attributes contain some null values. We also noticed some outliers by drawing empirical box boundaries of the United States: there are some incidents recorded outside of the U.S. that will be removed in the next phase.

Attributes *congressional_district*, *state_house_district* and *state_senate_district* contain some null values. By plotting the distribution for each of these attributes, we noticed that most of the incidents happened in congressional district 7 and 1 of Illinois state and in congressional district 2 of Lousiana state. We also observed that the most dangerous cities such as Chicago and its suburbs for Illinois and New Orleans for Louisiana fall in these districts.

Age and gender information:

The *participant_age1* attribute contains some outliers, mostly being values of type string and values that are too large to be the age of a person. There are also some wrong values if this attribute is compared to the corresponding value reported in the *participant_age_group1* field: we assume that the attributes *participant_age1*, *participant_age_group1* and *participant_gender1* all belong to the same randomly selected person, thus they have to coherently describe the age and gender of the participant. As shown in Figure 3, most of the participants are adult males. It can also be noticed that there is an outlier value "Male, female" for *participant_gender1*. Concerning age related features *min_age_participant*, *max_age_participants*, *avg_age_participants*, *participant_age1*, they all have similar distributions.

The attributes *n_participants_child*, *n_participants_teen* and *n_participants_adult* all present the same issues: they all contain outliers given by non-numerical strings, very large or negative numbers. These outliers will be handled during the data preparation step. **Number of involved people**:

The majority of the incidents only involve between 0 and 3 people. In particular most of the incidents report a very small number of killed, injured, unharmed or arrested people. We also noticed that in 62772 rows the following equality does not hold.

$$n\_participants = n\_killed + n\_injured + n\_unharmed = n\_males + n\_females$$

This mostly happens because the values sum up to a different number or due to null values.

Notes and incident characteristics:

More than 30% of records have null values for *notes* feature; the same holds for *incident_characteristics2*. The feature *incident_characteristics1* instead could be useful as they don't contain many missing values.

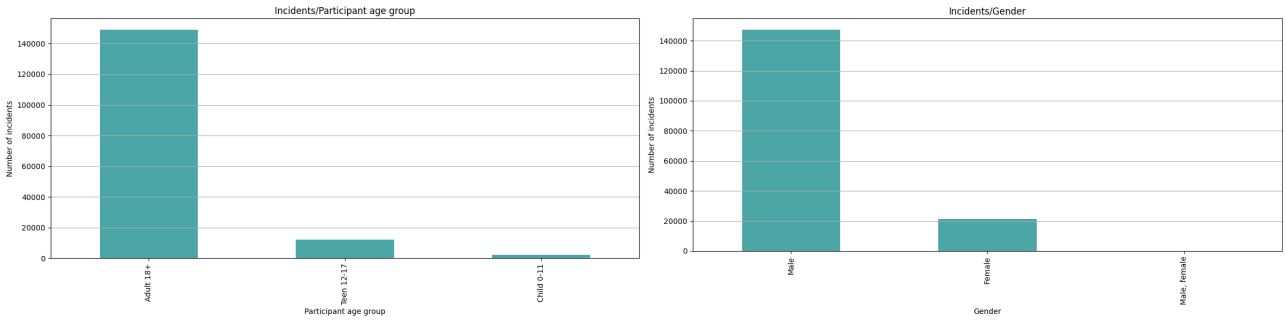


Figure 3: Number of incidents per age group (left) and gender (right) of the randomly chosen participant.

1.1.2 Poverty by state

This dataset contains three attributes: *state*, *year*, *povertyPercentage*. *povertyPercentage* is the only attribute with missing values. The *state* attribute contains 52 unique values: 51 of them are the same as the states in the incidents dataset, the remaining one is labeled "United States" and contains the average of the whole country.

The *povertyPercentage* attribute has no values for the year 2012, but we are only interested in relating this information to the incidents dataset, which only contains relevant incidents in the range of years 2014-2017.

1.1.3 Year state district house

This dataset is composed of the attributes *year*, *state*, *congressional_district*, *party*, *candidatevotes* and *totalvotes* and it contains no null values. We will only consider data in the range of years 2014-2017 for integrating this data with the incidents dataset.

1.2 Data Integration

We merged the incidents dataset with the poverty dataset using the attributes *state* and *year*. We then merged the resulting dataset with the remaining one using the attributes *state*, *year* and *congressional_district*. During the data integration process, records containing incidents set outside the U.S. were automatically deleted, resulting in a dataset that has no wrong values in attributes *latitude* and *longitude*. The dataset obtained by data integration will be used to perform further analysis and to have a more compact dataset.

1.3 Distribution Analysis

In order to analyze the features in the dataset, we displayed and examined the distribution of each column. This report only contains the most relevant distribution plots. Figure 4 shows the plots related to age attributes, *participant_age1*, *min_age_participants*, *max_age_participants* and *avg_age_participants*: as previously said, they all have very similar distributions.

From Figure 5 we can see that most of the incidents involve very few participants. Specifically, the majority of them only involve between 0 and 3 participants, with incidents having only one person involved being the most common ones.

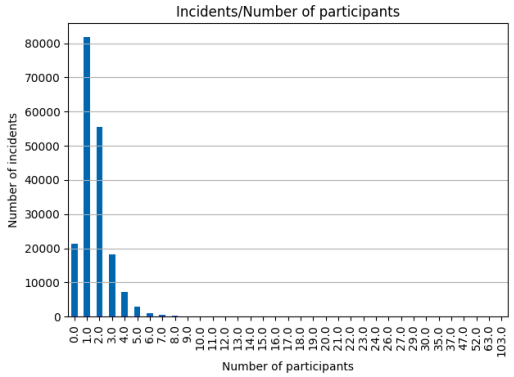


Figure 5: Number of incidents per number of people involved.

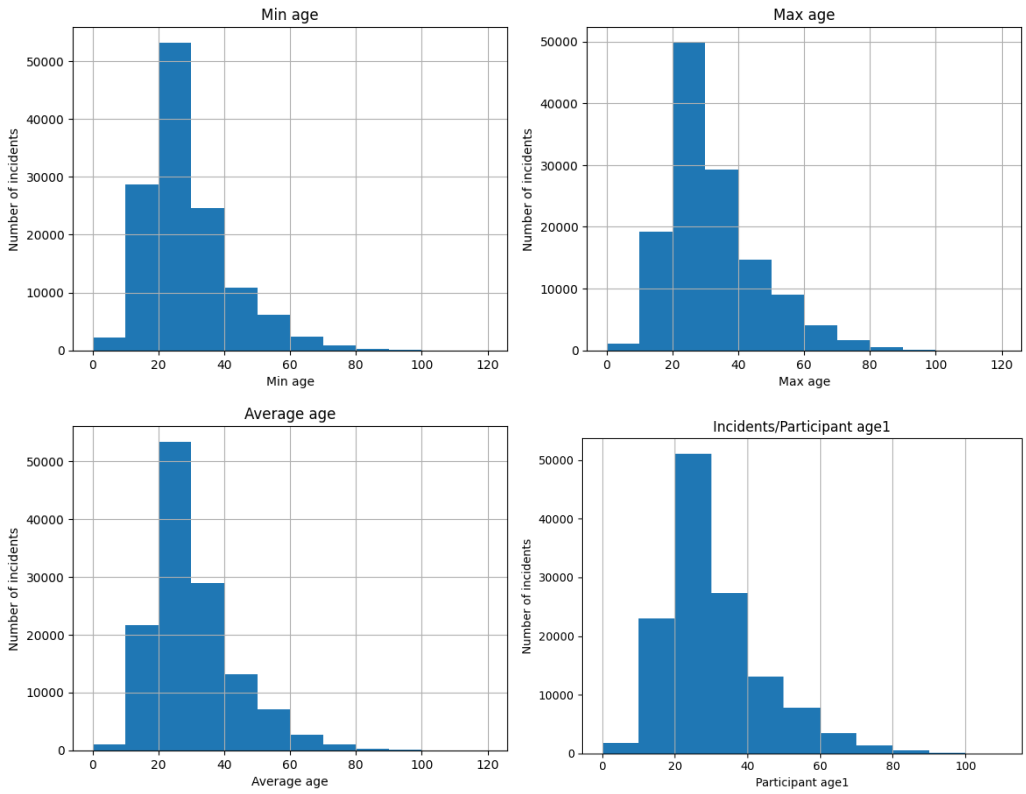


Figure 4: Plot of minimum (top left), maximum (top right), average (bottom left) and participant1 (bottom right) age distributions.

As previously shown in Figure 3, the majority of participants recorded in the incidents is comprised of adult (18 years old or older) and male people.

We also analyzed the geographical distribution of the incidents over the U.S. territory. Specifically we focused on the number of killed people in each incident in order to possibly understand the relation between killed people and geographical location. The plots in Figure 6 show the individual incidents recorded and the mass shooting events. The incidents that resulted in 0 to 3 killed people are the majority: this trend is confirmed by the first map, given that most of the dots are dark purple, that means few or no killed people. The second map instead highlights events that are to be considered mass shootings such as, for example, Orlando (Florida) mass shooting in 2016 (50 killed people) and Southerland Springs (Texas) mass shooting (27 killed people).

1.4 Correlation Analysis

The correlation matrix is shown in Figure 7. At first glance, the only noteworthy correlations are those between the *participant_age1* feature of the randomly taken person and the attributes *min_age_participants*, *max_age_participants* and *avg_age_participants*. This relation between features is probably due to the fact that the majority of the incidents only involve 1 or 2 participants. The correlation matrix will be computed again once outliers in the numerical attributes, which compromise the correlation calculation, have been eliminated.

We also observed that the number of males involved has a high correlation (0.83) with the number of participants because on average, as seen above, incidents tend to have a much more higher number of males participants than females participants.

2 Data Preparation

The data preparation phase uses the information gained in the previous phase to select records, manage outliers and missing values, improve data quality and prepare data by extracting new features interesting for describing the incidents. Each attribute was casted to the correct data type, in order to be coherent with the semantics of the feature, as shown in Table 1.

Duplicate records and records where all attributes were null were deleted. For all numerical attributes, negative values were set to null. We will describe in details in following parts the data preparation choices, grouped by different semantic topics.

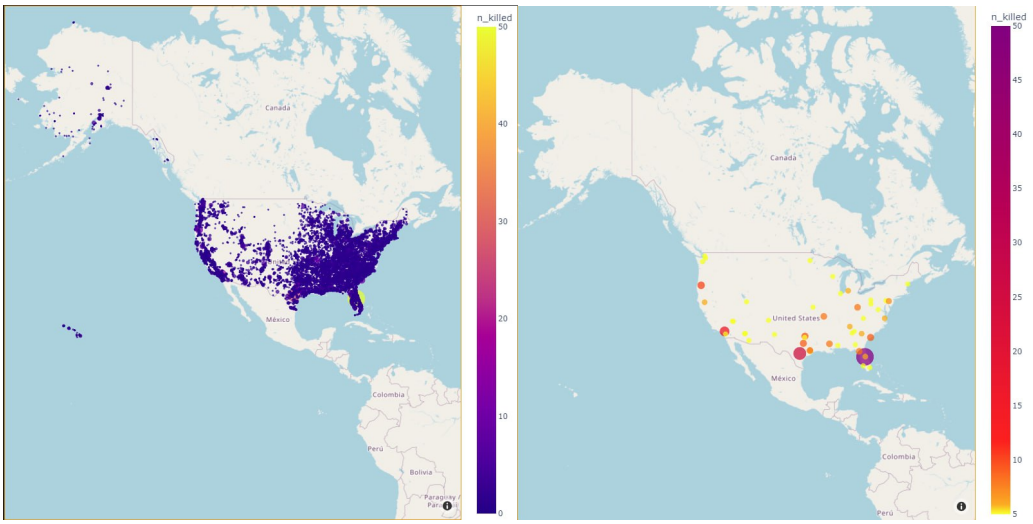


Figure 6: Map representing each incident (left) and mass-shooting incidents (right) recorded, the color used for the representation varies in relation to the number of people killed in the incident. The FBI has not set a minimum number of casualties to qualify an event as a mass shooting, but U.S. statute (the Investigative Assistance for Violent Crimes Act of 2012) defines a "mass killing" as "3 or more killings in a single incident." (source)

Date:
All records relating to events happened after 2023-10-01 (the date we received the dataset) were considered to be outliers, specifically to be errors in the data, and as such they were dropped. We also dropped all records related to year 2013, as the year was under-represented, and to year 2018, because there were only incidents for the first three months of the year. Finally, the feature *year* was dropped, as it was only a result of data integration.
Age attributes:
For the attributes *participant_age1*, *min_age_participants*, *max_age_participants* and *avg_age_participants*, values greater than 120 were considered to be outliers and set to null. After removing the outliers, these features still have similar distribution shapes (see Figure 4), mean and median. So, we delay the missing values fixing after the correlation analysis.
Geographical attributes:
The records with coordinates outside U.S. (other than null values) were automatically deleted after the data integration. We consider the triple $< date, latitude, longitude >$ to be a key identifying an incident in a unique way, because we assume that there is very difficult to have incidents happening on the same day in the exact same geographic coordinates. Hence, we decided to eliminate the records in which these 3 values are duplicates. For the rows in which *latitude* and *longitude* are null, the missing values were filled using the mean grouping by *state* and *city_or_county* to relocate correctly the incident records using their information about the state and city/county. This allowed us to replace all missing values for *latitude* and *longitude*.

The columns *state_house_district* and *state_senate_district* were dropped from the dataset, because they represent further subdivisions of the US territory that we considered not pertinent to our analysis. In fact, we are only interested in the *congressional_district* attribute, because the electoral information provided has the same granularity. The feature *address* was also deleted, because the exact location of the incident is already given by the geographic coordinates.
Number of participants attributes:
Outliers values found in *n_participants_adult*, *n_participants_teen* and *n_participants_child* features, namely those due to extremely large values, were set to null, using as threshold 103 i.e. the maximum value found in column *n_participants*, as it has no significant large values. Furthermore, in records that had *n_participants* equal to 0, all other attributes relating to the quantitative information of participants were also set to 0.

In order to keep the dataset coherent, the value in *n_participants* was replaced with *n_males* + *n_females*, if this is equal to *n_participants_adult* + *n_participants_teen* + *n_participants_child* and different from *n_participants*. Otherwise, if *n_males* + *n_females* is different from *n_participants_adult* + *n_participants_teen* + *n_participants_child* we set to null these attributes in the rows where these sums and/or *n_participants* are not equal. The null values were then substituted using the mean of each attribute, grouped by *n_participants*. This is done to preserve the original distribution of these attributes and also to avoid exceeding the number of participants by replacing the missing values of these attributes with a global mean; the few rows for which this replacement

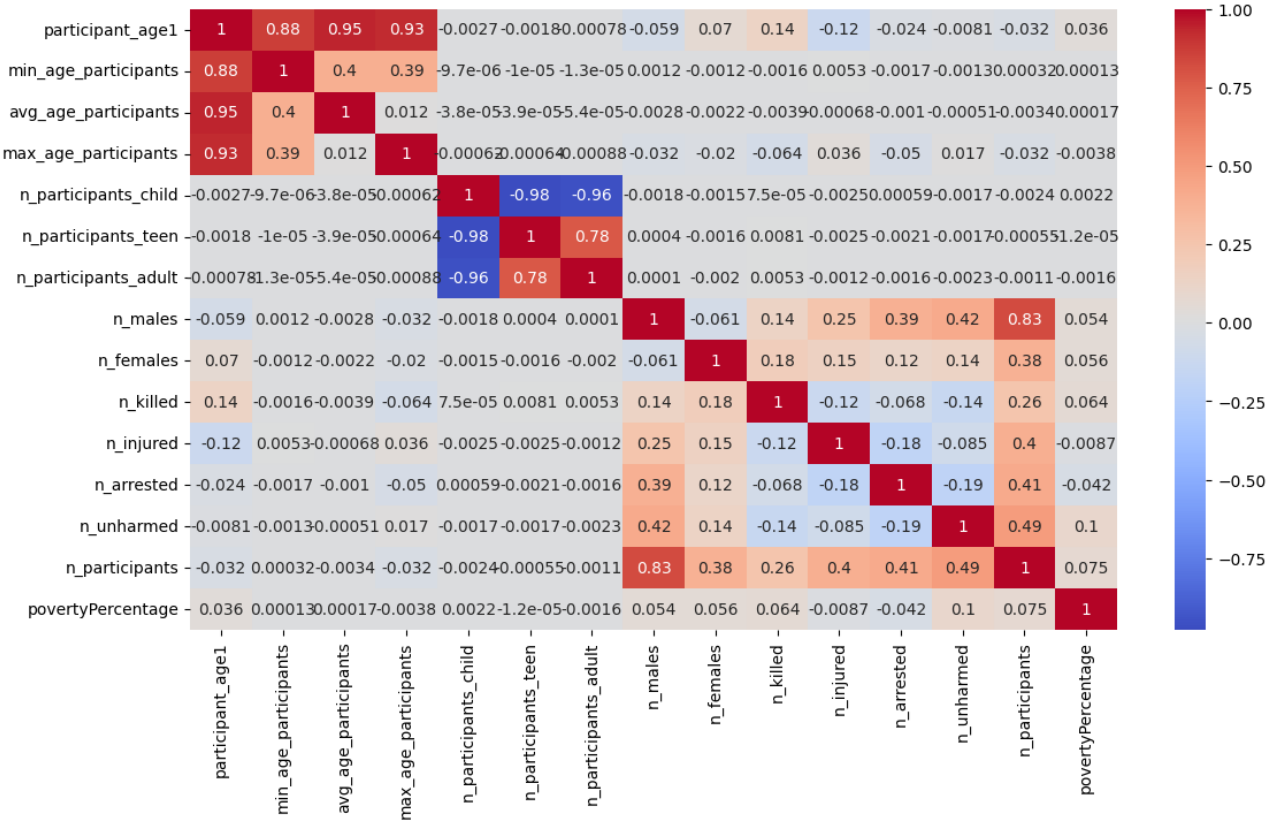


Figure 7: Correlation matrix of the most relevant numerical attributes.

was not possible (we were not able to reconstruct the mean) were dropped from the dataset. There were no relevant changes in the distribution of these attributes after the cleaning.

We checked whether the number of killed, injured, arrested and unharmed people exceeds the total number of participants in that incident. In case they do, we set them to null and then proceeded to fill null values using the mean. There was no major change in the distribution of these feature after this operation.

Incident characteristics:

In order to fill the missing values for feature *incident_characteristics1*, we analyzed records by looking at their values for *n_killed*, *n_injured* and *notes* attributes. This was done in order to create subgroups of records, so that incidents characteristics could be reconstructed if missing. For instance, if the incident record has at least one death and in its notes contains some expressions concerning gun violence, we fill the null values for *incident_characteristics1* with "Shot - Dead (murder, accidental, suicide)" category.

For records that did not fit in any of these subgroups, *incident_characteristics1* was filled using the string "Shots Fired - No Injuries", as they reported no injured nor killed people. We dropped the column *incident_characteristics2* given that it has 40% of null values and it adds only useless details to *incident_characteristics1* for our analysis. Similarly, the attribute *notes* was dropped because of the large number of missing values.

2.1 Correlation analysis

The correlation matrix, shown in Figure 8, has been computed only for numerical attributes. It can be seen that age attributes, namely *participant_age1*, *min_age_participants*, *max_age_participants* and *avg_age_participants*, are highly correlated with each other, confirming what we have observed during the distribution analysis of these features. For this reason, we decided to drop them all, except for *avg_age_participants*, which is the most correlated (95%) to the other attributes and gives us more general information about all the participants. Given the fact that *participant_age1* was dropped, the attributes *participant_gender1* and *participant_age_group1* become useless and were also dropped because there no longer is any information about the randomly selected person.

So the only remaining feature related to the participants age is *avg_age_participants*, which contains some missing values. We decide to fill them with the mean. In particular, we assume that if there is at least one adult, we fill the values with the general mean age; if no adult and at least one teen is present, we fill the values with the average age of the teens; if no adult and no teen are present, we fill the null value with the average age

Feature Name	Initial Type	Cast Type	Description
date	object	Datetime64	date of incident occurrence
state	object	String	state where incident took place
city_or_county	object	String	city or county where incident took place
address	object	String	address where incident took place
latitude	float64	float64	latitude of the incident
longitude	float64	float64	longitude of the incident
congressional_district	int64	Int64	congressional district where the incident took place
state_house_district	int64	Int64	state house district
state_senate_district	float64	Int64	state senate district where the incident took place
participant_age1	float64	Int64	exact age of one (randomly chosen) participant in the incident
participant_age_group1	object	String	exact age group of one (randomly chosen) participant in the incident
participant_gender1	object	String	exact gender of one (randomly chosen) participant in the incident
min_age_participants	object	Int64	minimum age of the participants in the incident
avg_age_participants	object	float64	average age of the participants in the incident
max_age_participants	object	Int64	maximum age of the participants in the incident
n_participants_child	object	Int64	number of child participants 0-11
n_participants_teen	object	Int64	number of teen participants 12-17
n_participants_adult	object	Int64	number of adult participants (18 +)
n_males	float64	Int64	number of males participants
n_females	float64	Int64	number of females participants
n_killed	int64	Int64	number of people killed
n_injured	int64	Int64	number of people injured
n_arrested	float64	Int64	number of arrested participants
n_unharmed	float64	Int64	number of unharmed participants
n_participants	float64	Int64	number of participants in the incident
notes	object	String	additional notes about the incident
incident_characteristics1	object	String	incident characteristics
incident_characteristics2	object	String	incident characteristics
year	int64	Int64	year of the incident occurrence
povertyPercentage	float64	float64	poverty percentage for the corresponding state and year
party	object	String	winning party for the corresponding congressional_district in the state, in the corresponding year
candidateVotes	int64	Int64	number of votes obtained by the winning party in the corresponding election
totalVotes	int64	Int64	total number of votes for the corresponding election

Table 1: Features types and semantics of the merged dataset

of the children. This is done because we observe that the mean of *n_participants* is around 1.5, so, if there is at least an adult or a teen or a child in the incidents, it is likely to be the only participant of that incident. The remaining records to fill have all participants information equals to 0, so we decided to drop them to not affect the distribution of the age. After these operations, we plotted, as usual, the distribution of *avg_age_participants* and we didn't note any particular change.

It can also be noticed that *n_participants_adult*, *n_males*, *n_participants* are all highly correlated. Furthermore, we observe that they all can be obtained summing the other attributes. In particular, n_participants can be obtained by $n\_participants\_adult + n\_participants\_teen + n\_participants\_child$, but also by $n\_males + n\_females$, and, *n_participants_adult* and *n_males* can be obtained by subtracting to *n_participants* the other columns. Hence the attributes *n_males* and *n_participants_adult* were dropped, to keep *n_participants* which holds the most general information.

2.2 Definition and analysis of new indicators

After analyzing the data, we defined the following indicators:

- *incident_gravity*, number of killed and injured people in an incident,

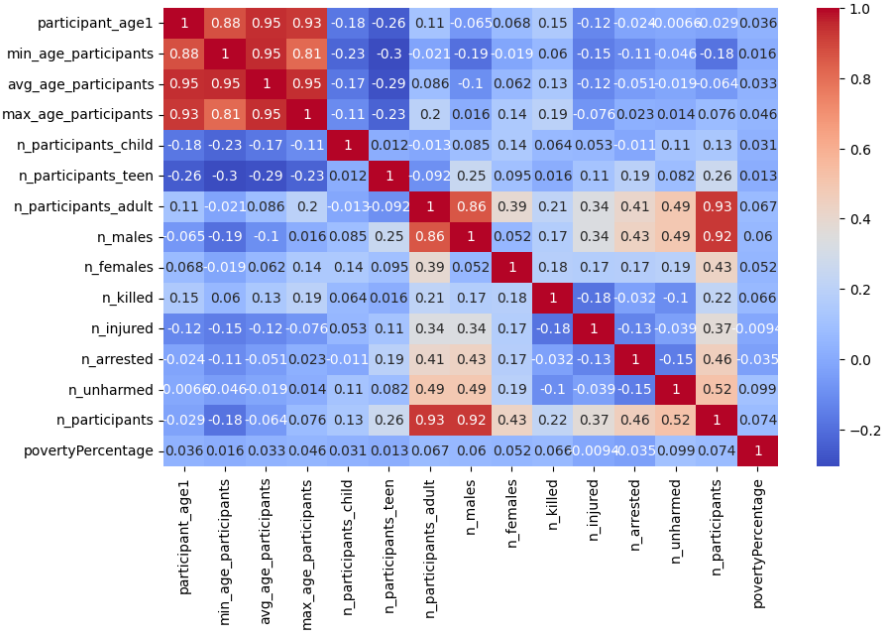


Figure 8: Correlation matrix of the most relevant numerical attributes.

- *females_rate*, number of female people involved in the incident over the total number of participants in that incident,
- *minor_rate*, number of minors (younger than 18 years old) involved in the incident over the total number of participants,
- *arrested_rate*, number of arrested people in the incident over the total number of participants,
- *survival_rate*, number of unharmed people in the incident over the total number of participants,
- *injured_rate*, number of injured people in the incident over the total number of participants,
- *killed_rate*, number of killed people in the incident over the total number of participants,
- *winning_party_percentage*, the number of votes for the winning candidate over the total number of votes in that election.
- *killed_disp_per_district*, number of killed people in the incident over the number of killed people in that same congressional district in the same year,
- *injured_disp_per_district*, number of injured people in the incident over the number of injured people in that same congressional district in the same year,
- *part_disp_per_district*, number of participants in the incident over the number of participants in incidents in that same congressional district in the same year.

After defining the indicators and describing their semantic, we perform a step of data understanding on these new variables, calculating their mean, median, standard deviation, plotting and analysing each distribution and the correlation matrix. We insert the most interesting distribution plot and the correlation matrix in the report (see respectively Figure 9 and 10).

It can be noticed that indicators *female_ratio* and *minor_ratio* both have similar behaviors and distributions: for the majority of the incidents there are no females nor minors. For most of the records, the rates of survival, arrested, injured and killed people are near to 0. For a better understanding, we look at the value of each ratio indicator against the attributes *n_killed*,*n_injured*,*n_arrested*,*n_unharmed*,*n_participants*. In fact, we discover that a lot of incidents involving injuries, arrests or survivors do not involve deaths. Similarly, incidents involving injuries, arrests or deaths do not involve any survivors. The same tendency can also be observed for *arrested_rate* and *injured_rate*. This analysis highlights that these indicators are uncorrelated and this is confirmed by looking also at the correlation matrix.

The last three indicators defined per district often have value 0, which shows that the number of killed or injured people or simply the participants to the particular incident are equal or below the mean of killed or

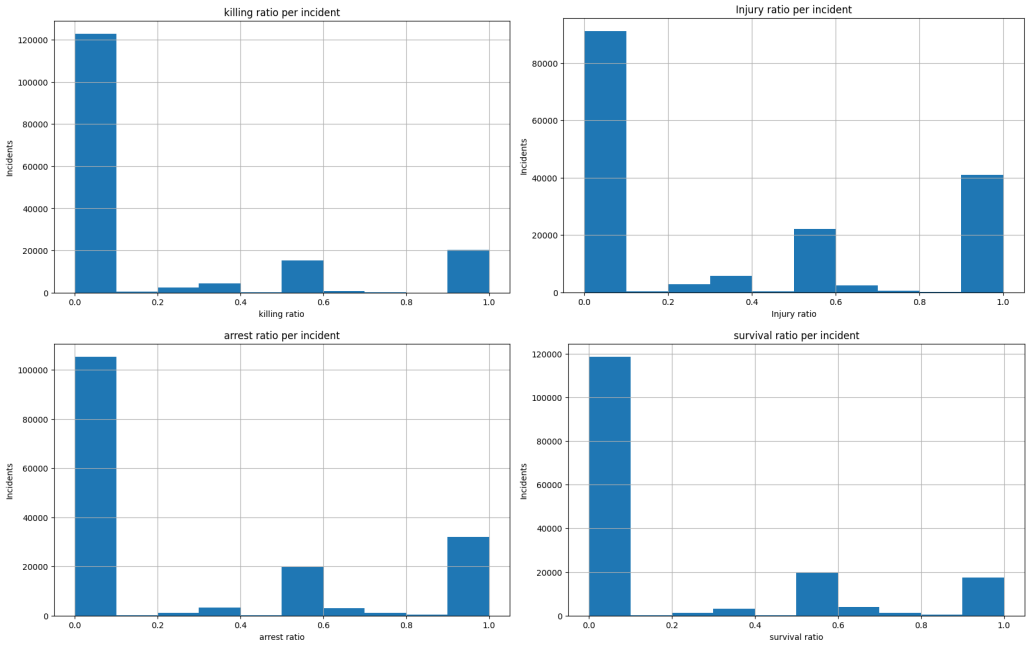


Figure 9: Plot of killing (top left), injury (top right), arrest (bottom left) and survival (bottom right) ratio distributions.

injured people or participants in each district in a specific year. This can be explained also by the fact that we substitute the division by 0 with 0, while calculating the values of these indicators. Overall, the indicators we identified have no missing values; they have some outliers that are coherent with the one in the original numerical attributes, so we decided not to clean them. All these indicators provide a good characterization of the incidents and are at most moderately correlated.

3 Clustering analysis

3.1 Preprocessing

In order to apply clustering methods, data needs to be prepared through preprocessing steps.

For clustering, we will use distance metrics that only work on numerical attributes, so we decided to remove any categorical attribute. However, also the numerical attributes that are highly correlated with new indicators defined (we will plot the correlation matrix in classification task) were deleted, focusing mainly on the indicator attributes, listed in the previous section. Normalization of numerical features, using data scaling, has been applied, a valuable technique in clustering algorithms to prevent any attribute from disproportionately interfering with the analysis due to its scale. In particular, the Z-Score Normalization has been applied. Before applying any clustering techniques, we ensured that our data has a sort of clustering structure, plotting data distribution through PCA. We defined some new binary columns only to label the PCA results (not used for the clustering):

- *isKilled*, its value is 0 if nobody was killed in the incident, it is 1 if at least a person was killed;
- *involve_injury*, its value is 0 if nobody was injured in the incident, it is 1 if at least a person was injured;

The new feature *isKilled* will also be used in the following classification task. As shown in Figure 11, we can see emerging about four clusters of incidents, two of them clearly involving a kill, and the other two not. So, we applied clustering techniques to the data and we analyzed the results in the following subsections.

3.2 Characterization of Indicators

Two extra indicators, *involve_arrest* and *involve_unharmed*, have been created only for characterizing the clusters. These, in conjunction with the previously mentioned *isKilled* and *involve_injury* indicators, have significantly contributed to comprehending the structure of the clusters.

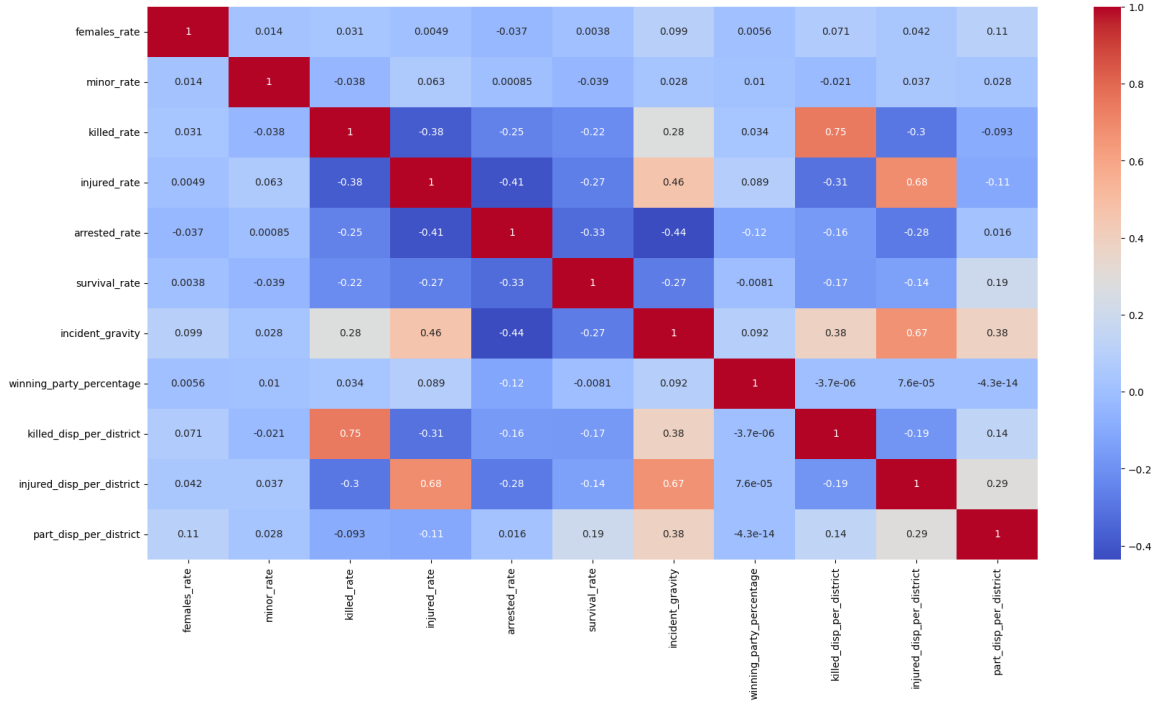


Figure 10: Correlation matrix of the newly created indicators

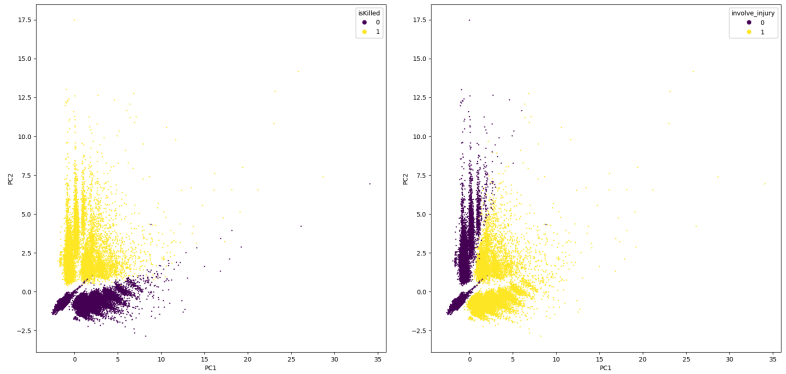


Figure 11: Plot of PCA colored with isKilled (left) and PCA colored with involve incident (right).

3.3 K-Means

Best K value

To estimate the best value of k to apply the K-means clustering algorithm, the elbow method was used. The grid search on values of $k \in \{2, 3, 4, 10, 20, 30, 50, 100\}$ gave the results which can seen on Table 2 in terms of the metrics SSE, separation (Davies-Bouldin Index) and silhouette. A plot of the SSE and silhouette scores is shown in Figure 12.

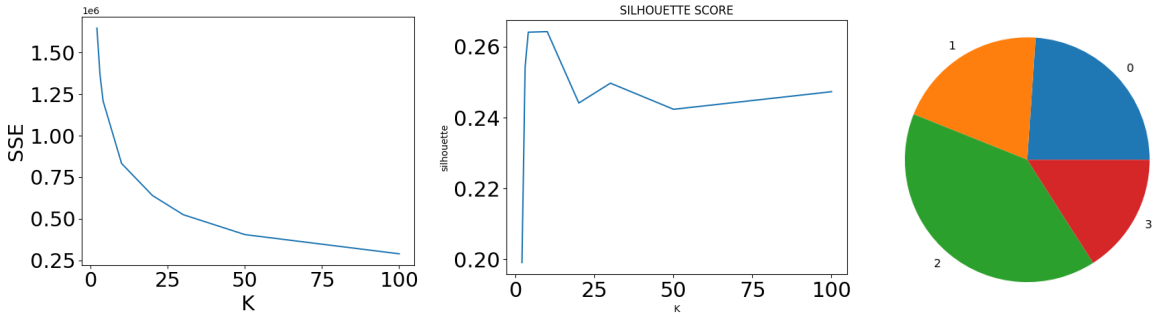
Both the elbow method and the silhouette score point out that the best value for k is around four. The cluster distribution colored by cluster labels can be seen in Pie Plot in Figure 12c.

Clusters Analysis

First of all, we colored each cluster using the binary indicators defined above in sections 3.1 and 3.2. The Pie Plots in Figure 13 show an interesting trend: each cluster is characterized, for the majority of records, only by a specific binary indicator set to true(1), while the others (not shown in figure) are set mostly to false(0). In particular, for the first cluster *involve_injury* is mostly true, while for the second *involve_arrest*, for the third *isKilled* and for the last *involve_unharmed* are the only binary indicators holding mostly true values. This behavior is confirmed by the radar plots and the line plots in Figure 15, in which we can see that for the first cluster the most important features are the indicators *injured_rate* and *injured_rate_per_district*, for the second,

K	SSE	separation	silhouette
2	1646704.87	1.98	0.20
3	1366709.62	1.58	0.25
4	1208515.04	1.37	0.26
10	832875.83	1.49	0.26
20	640117.77	1.47	0.21
30	524562.02	1.38	0.23
50	404817.22	1.30	0.24
100	289769.5750204325	1.32	0.24

Table 2: Values of SSE, separation and silhouette score computed for increasing values of k. (K-means)



(a) (b) (c)

Figure 12: Plots of SSE (a), the silhouette (b) scores and of the distribution of the cluster labels (c). (K-means)

arrested_rate, for the third *killed_rate* and *killed_rate_per_district* and for the last one *survival_rate*. It's also interesting to notice from these plots that all the other indicators such as *povertyPercentage*, *females_rate* and *minor_rate* are not particularly determining for the clustering. Moreover, the clusters distribution behavior became even clearer, when we have colored each cluster by *incident_characteristics1*. In particular, from Figure 14, it emerged that the cluster 0, associated with incidents that mostly involve injured people, is linked with events characterized by non-fatal gun violence. The cluster 1, the "arrests cluster", is related to minor offenses, such as non-use of firearms or firearms shots that resulted in no injuries in which the police probably managed to prevent the gun incident by promptly arresting the perpetrators. The cluster 2, linked to the fatality attribute, predominantly exhibits incidents categorized as "Shot - Dead (including murder, accidental deaths, or suicides)". Finally, the cluster 3, the cluster with majority of incidents with unharmed people, shows a dominance of incidents not involving guns nor injuries.

We also mention that each cluster was also colored according to states but no clear predominance emerged. The states with the most incidents which as seen in the data understanding are Illinois, California, Florida, Texas, New York are similarly distributed across the clusters occupying a good percentage of the records. The only notable clear gap is present in cluster 0, the cluster containing crimes with the most injuries, where Illinois dominates with nearly nine thousand records.

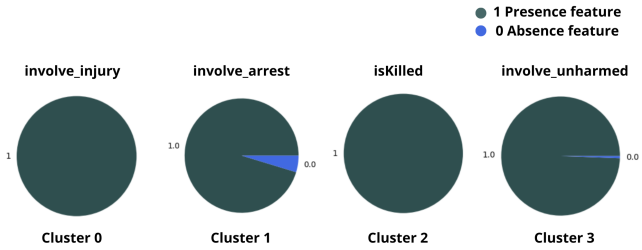


Figure 13: Pie Plots of each cluster distribution colored by the most important binary indicator (K-Means)

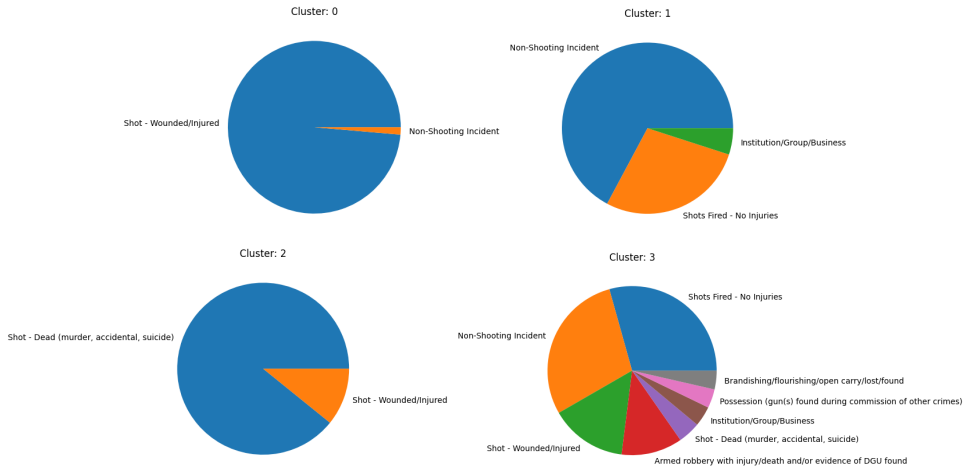


Figure 14: Pie Plots of the distribution of each cluster colored by most relevant incidents characteristics(K-Means).

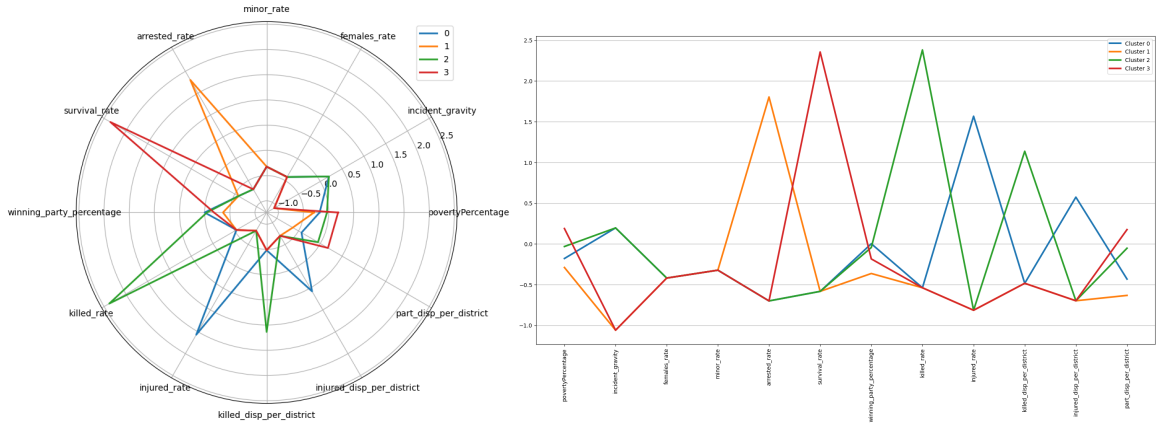


Figure 15: Plots of the most representative features for each cluster, obtained using K-Means clustering, Radar plot(left) and Line Plot(right).

3.4 Dbscan clustering

As requested by the clustering sub task in the project description, only one state was chosen for this clustering task: the state of California, which is in the top three states per number of gun incidents.

To estimate the best value for the epsilon (ε) parameter of Dbscan, the elbow method was used; in particular, we computed the distance to the k nearest neighbors (with k set to 10) and sorted them in decreasing order. The results of the approach is shown in Figure 16a. The best estimation found for the value of ε is around 1.8. So, different values of ε around 1.8 were then explored along with varying values of *min_samples*. The results in decresing order of silhouette score are shown in Table 3. Therefore, the best values of ε and *min_samples* found are 1.6 and 160 respectively.

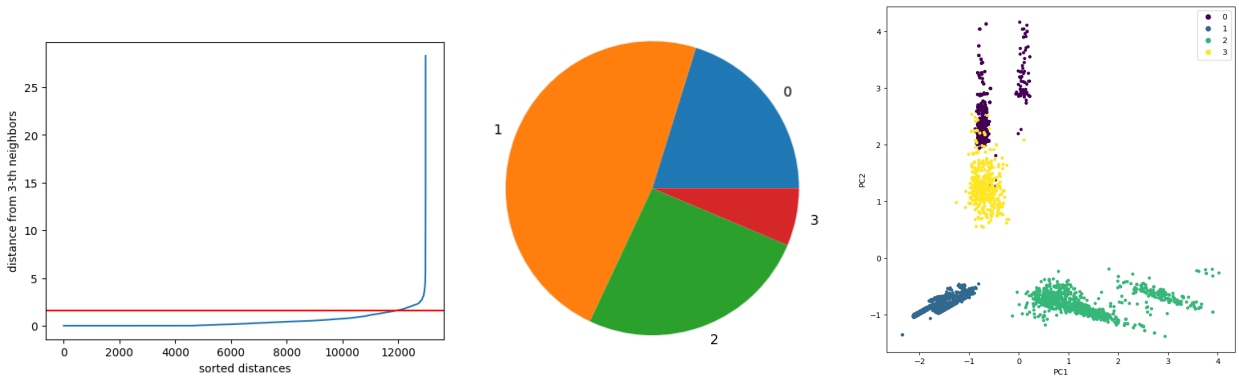
The application of Dbscan to the dataset using the best values for ε and *min_samples* led to the finding of five clusters. However, in the cluster analysis, we didn't consider important the cluster containing only noise points labeled as -1 (see cluster label distribution on Figure 16b) and we excluded it.

As with K-means, we colored each cluster distribution with the binary indicators defined in sections 3.1 and 3.2. Then, we compared the results with radar and line plots and with the same cluster distribution colored by *incident_characteristics1*. The results obtained are similar to those of K-means with some important differences: not every cluster is characterized by a distinct indicator. In particular, the first and the last clusters are discriminated by true values of the same *isKilled* variable, the first more and the last less (see Figure 17). The second and the third clusters, instead, similarly to K-means are distinguished respectively mostly by *involve_arrest* and *involve_injury*. We decided to investigate this behavior further. We observed that plotting the radar and line plots (see Figure 18), the first and last clusters are distinguished by high values of death ratio and death ratio by district. Moreover, coloring the distributions with *incident_characteristics1*, we noticed that the same two clusters contain incidents resulting in deaths by firearms. The exact same K-means considerations

can be made, however, for the second and third clusters characterized by arrests and incidents involving injuries, respectively. The features that do not affect clustering are also the same as those observed for K-means (see Figure **??** before and 18 then). Another interesting coloring of the clusters was by *city_or_county*. With this coloring, we found that both clusters concerning crimes resulting in deaths are mostly related to Los Angeles which is the city with the largest population in California[1]. This suggested us that the two clusters are probably close in space and in fact the trend is confirmed by running a PCA (see cluster 0 and 3 in Figure 16c) and labeling the examples with cluster labels. In the remaining clusters, however, no particular city emerges, but a good percentage of incidents involving arrests or injuries are located in Oakland, Fresno, and Bakersfield.

ε	**silhouette**	**separation(DB score)**	**n_clusters**	**min_samples**
1.6	0.2030	2.0584	5	160
1.6	0.2027	2.0530	5	170
1.6	0.2004	2.0469	5	180
1.7	0.1829	2.6988	3	180
1.8	0.0813	2.6123	4	180
1.7	0.0703	2.6684	4	160

Table 3: Table showing the grid search results for the best values of ε and *min_samples* parameters. (Dbscan)



(a) (b) (c)

Figure 16: Plots of the elbow method result (a), the distribution of the cluster labels (b) and the PCA components colored by cluster labels (c) (Dbscan).

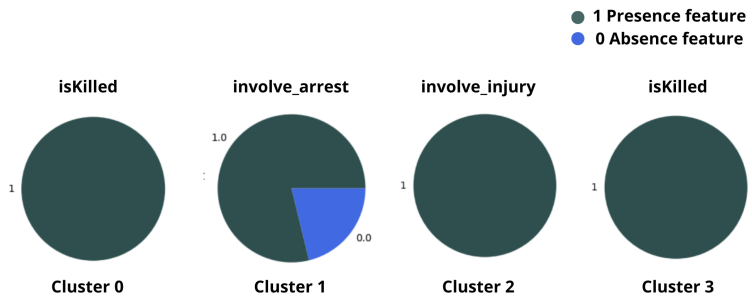


Figure 17: Pie Plots of each cluster distribution colored by the most important binary indicator (Dbscan).

3.5 Hierarchical clustering

As described in the clustering sub task in the project description, only one state was chosen for this clustering task: the state of California. To identify the best distance metric and the best method, we performed an

[1]source: Annual Estimates of the Resident Population for Incorporated Places in California: April 1, 2020 to July 1, 2022. United States Census Bureau. Retrieved May 18, 2023.

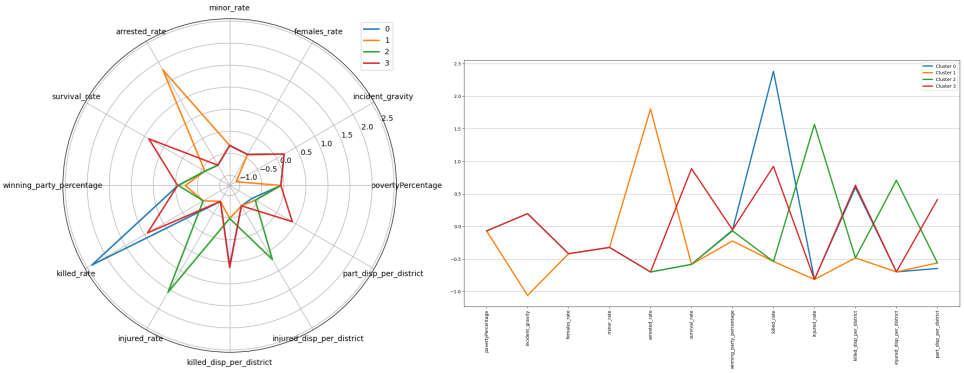
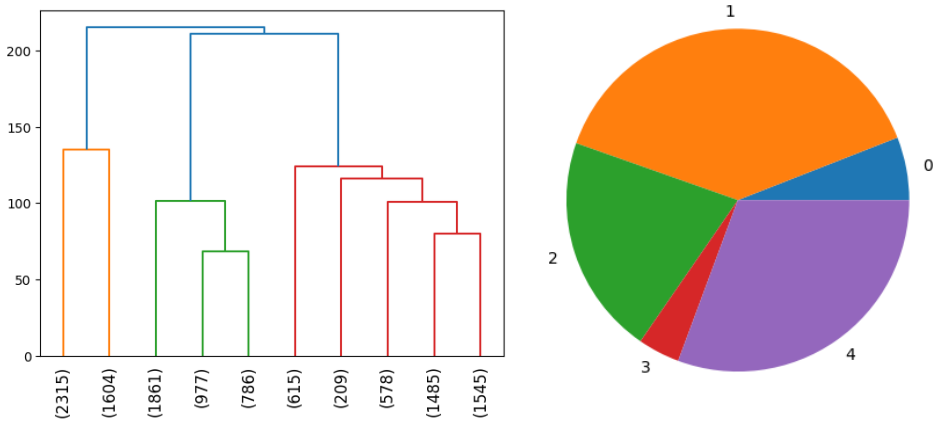


Figure 18: Plots of the more representative features for each cluster, obtained with DBScan, Radar plot(left) and Line Plot(right).

extensive evaluation on different combinations of metrics and linkage criteria concerning mostly agglomerative type of clustering. The best results were obtained by using the euclidean distance metric, ward linkage method and "lastp" as truncation mode.

By using this final model, the silhouette score is about 0.31 and the separation (Davis-Bouldin Index) 1.29. The dendrogram displayed in Figure 19a shows five different clusters, one more with respect to K-means and Dbscan, which are represented from the distributions in Figure 19b. We performed the exact same analysis and distribution coloring done for K-means and Dbscan. We can make much the same observation as for K-means and Dbscan, because the cluster behaviors are very similar. The only difference, that makes the cluster distribution less clear to understand with respect to K-means or Dbscan, came with the fourth cluster, the smallest one. In this cluster, no binary indicator do sharply characterize the distribution and it can be noticed that it is also the only one characterized by high value of female ratio and minor ratio (see radar and line plots in Figure 20). It's reasonable to think that it probably represents a cluster with only outliers.



(a) (b)

Figure 19: Plots of the dendrogram obtained by applying hierarchical clustering (a) and the distribution of the cluster labels (b).

3.6 X-Means

We also analyzed and tested a further clustering algorithm, X-means. The X-means implementation used is *pyclustering*[2]. The starting parameter for X-means was the maximum number of clusters, set to 10, repeating 10 times the execution of K-means, starting from four centroids. The starting centroids positions where chosen using the initializer of the *pyclustering* implementation. After computing the clusters with the X-means algorithm, we obtained exactly 10 clusters. This outcome suggested that the algorithm tended to select the maximum number of clusters. Consequently, to prove this point, we attempted to use a larger number of clusters and our tests consistently converged to that specific number. Then, for the final evaluation, we continued to use

[2]Pyclustering: `https://github.com/annoviko/pyclustering/`

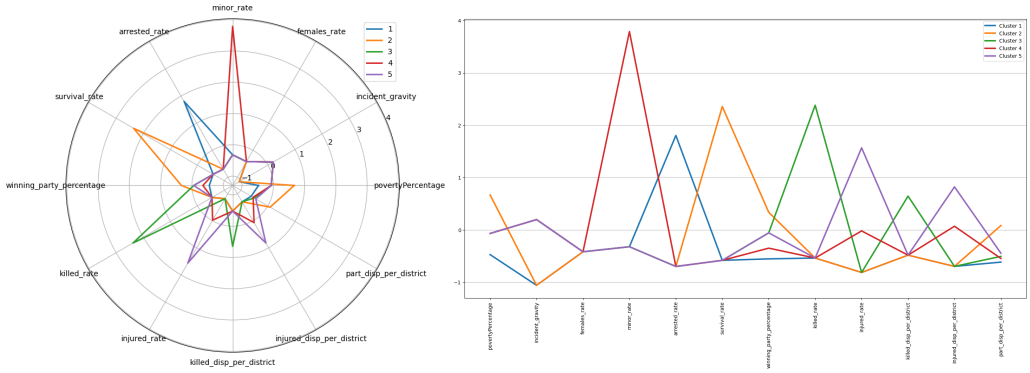


Figure 20: Plots of the features more representative for each cluster, obtained using Hierarchical clustering, Radar plot(left) and Line Plot(right).

10 as maximum number of clusters and the metrics of silhouette and separation obtained from these clusters were 1.43 and 0.22 respectively. Several clusters obtained are very small as we can see in the Pie Plot colored with cluster labels in Figure 21. Moreover, the distribution of values of the clusters, as we can see in radar and line plots Figure 21 is less defined and much more difficult to interpret than the K-Means implementation. Considering this, we concluded that X-means doesn't give us an interesting clusterization.

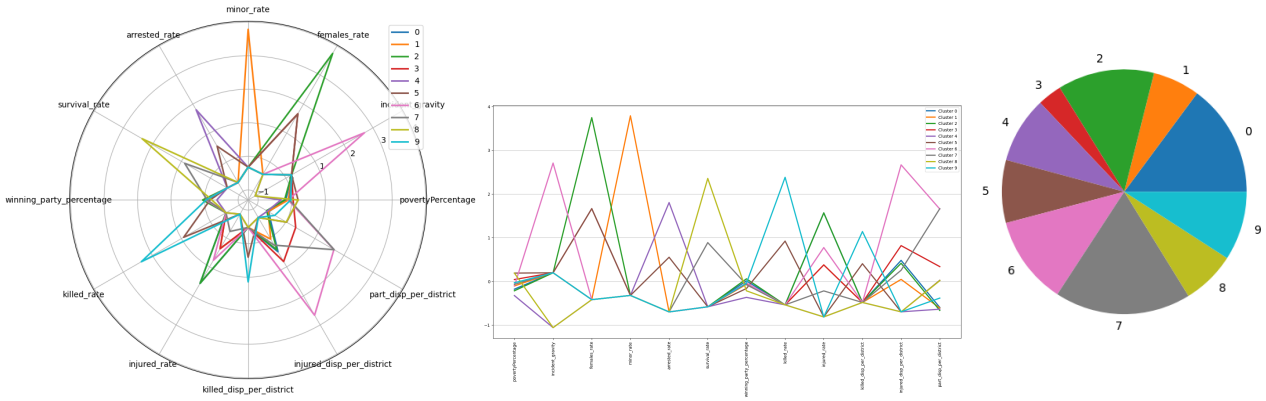


Figure 21: Plots of the most representative features in the XMeans Clusters (left Radar Plot, middle Line Plot) and the plot of the distribution of the 10 cluster labels (right).

3.7 Clusters comparison and best result

From the various clustering methods used, different structures emerged in the clusters. Among the better-performing methods K-Means, Dbscan, and Hierarchical—clusters fell within the range of four to five clusters. Concerning evaluation metrics, K-Means and Hierarchical methods showed similar values of separation metrics, while DBScan had poorer silhouette and Davis-Bouldin Index measures as shown in Table 4.

For what concern the quality of the cluster structures, as stated before, the clusters identified by Dbscan appeared less clearly defined compared to the distinct clusters found by K-Means. In fact, Dbscan has two clusters, first and last clusters, that refer to the same city and same incident types, and are distinguished by the same killing binary indicator. Moreover, the last cluster contains much less incidents than the others, as we can see in Figure 16b. This suggest us that these two clusters can be merged in a single bigger cluster. Finally, differently from K-Means, Dbscan was not able to find a cluster characterized by unharmed participants. When comparing Hierarchical clusters to K-Means, although Hierarchical showed slightly better metrics, the five clusters it produced didn't display the clear separation seen in K-Means. In fact, similarly to DbScan and differently from K-Means, Hierarchical clusters also contained mixed incidents within certain clusters.

Considering these observations, the K-Means method seemed the most effective. It achieved good scores in term of Davis-Bouldin Index and Silhouette and it resulted in four well-separated and easy to understand clusters, each representing a specific type of incident, as elaborated in the corresponding section.

Method	separation(DB Index)	Silhouette
K-Means	1.37	0.26
Hierarchical	1.29	0.31
DBScan	2.06	0.20

Table 4: Values of DB Index and silhouette computed for the three best-performing clustering methods.

4 Predictive Analysis

4.1 Preprocessing

The goal of the classification task is to predict whether an incident resulted in at least one kill or not. For this purpose, we defined the binary attribute *isKilled* and used it as label to predict. The categorical attributes *date*, *state*, *city_or_county* and *party* were discretized so that they could be used by classification models.

After plotting the correlation matrix of the numerical attributes, shown in Figure 22, all the indicators used

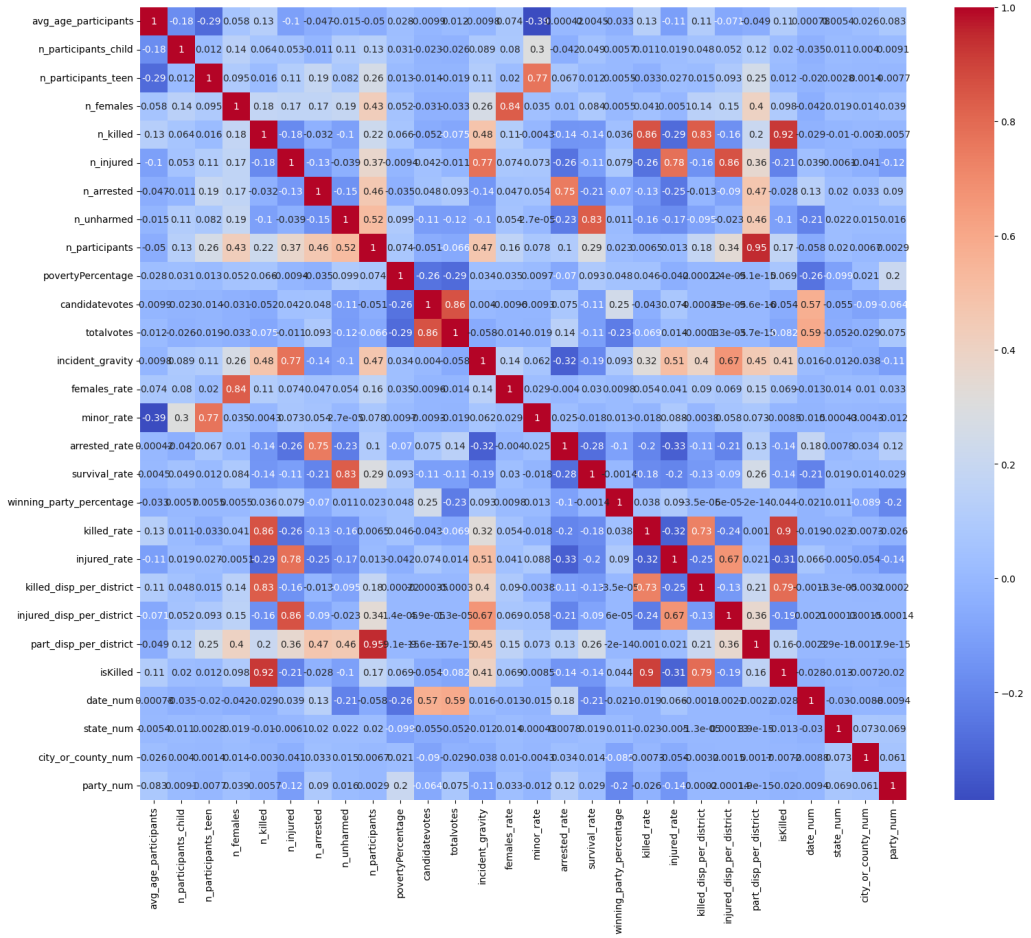


Figure 22: Correlation matrix of numerical attributes after Data Preparation and Clustering Analysis.

for clustering have been deleted, as they are all highly correlated with the original features from which they were computed. Additionally, other columns that were discarded are:

- *latitude* and *longitude*, as they represent a sort of incident unique identifier (each incident has a unique pair of *latitude* and *longitude* values after the data preparation), so they were considered irrelevant;
- *n_killed*, because it has a high correlation with *isKilled*;
- *candidatevotes* and *totalvotes*, as they add only further details to the majority winning party represented already by the attribute *party*, so they are irrelevant;
- *incident_characteristics1*, because it could simplify too much the classification task as it provides a specific label to each incident record.

The Python library `scikit-learn` provides every classification model and dataset splitting methodology used in this task. Before applying any classification methods, both in preliminary analysis (section 4.2) and the exhaustive analysis (section 4.3), the data set was partitioned such that a training set including 70% of the records and a test set including the remaining data (30%) were created. To keep the same class distribution as the original data set a stratified partitioning approach was employed, based on *isKilled* class attribute.

Moreover, to perform a robust model selection and once again keep in each partition the class distribution of the original dataset, a Stratified 5-Fold Cross Validation technique was applied on the training set. The metrics used for the model selection and model evaluation are accuracy, precision, recall, f1-measure. As a 5-Fold Cross Validation was employed, the performance metric values were averaged on training and on validation set folds and also the standard deviation was computed. To ensure the reproducibility of the results, the seed of each random operation involved in the partitioning of the data set into training and test sets and in cross-validation was fixed. Finally, in order to get a better performance from the classification methods, data was standardized.

More details about the predictive task and the employed classifiers will be explained in the following two sections.

4.2 Preliminary analysis and further preprocessing

We noticed that:

1. almost all the incident records in the dataset (157302 rows out of 166664) fit the linear relationship $n\_killed = n\_participants - (n\_injured + n\_unharmed + n\_arrested)$ and, remembering that *n_killed* is highly correlated with *isKilled*, this relationship would make the classification task much easier.

2. the dataset is unbalanced toward the negative class value (0) of the class attribute *isKilled*; in fact, only about 26% of the incident records involve at least a kill.

To address these two problems `Classification_all_features.ipynb` notebook was coded. In particular, for the first problem, some simple baseline methods, including Logistic Regression and Decision Tree, were tested; for the second problem, the analysis is extended also to a non-baseline method, the Neural Network. To compare the classifiers explored in this notebook, the Stratified 5-Fold Cross Validation discussed in section 4.1 was performed and mean accuracy, precision, recall and f1-measure on validation folds were calculated.

Linear relationship problem analysis

For Logistic Regression, a brief model selection with the Stratified 5-Fold Cross Validation approach discussed in section 4.1 was performed to choose the best solver between 'lbfgs', 'newton-cg', 'newton-cholesky', 'sag', 'saga'. The other hyperparameters were set to their default values.

The Decision Tree classifier was cross-validated with approximately its default values i.e. gini as splitting criterion, maximum depth equal to 10, minimum number of samples to make a split equal to 3, minimum number of samples required for a node to be considered a leaf node equal to 1.

We can observe in Table 5 how Logistic Regression and Decision Tree can easily reach more than 95% on accuracy and over 91-93% in precision, recall and f1 on validation set. These very good results are obtained even if these are baseline methods with approximately their default hyperparameters, and they were tested on unbalanced dataset without even using class weights to enhance the predictions of the minority class.

model	**accuracy**	**precision**	**recall**	**f1-measure**
Logistic Regression	0.97 ± 0.001	0.91 ± 0.003	0.98 ± 0.001	0.94 ± 0.002
Decision Tree	0.97 ± 0.001	0.93 ± 0.002	0.95 ± 0.005	0.95 ± 0.002

Table 5: Mean accuracy, precision, recall, f1-measure and their standard deviation calculated on validation folds according to 5-Fold Cross-Validation technique described in 4.1

For this reason, to test the classifiers on a more challenging task, we have decided to drop also *n_injured* and *n_unharmed*, assuming this information harder to find by police at scene of the incident with respect to *n_arrested*, as they can grow in time and be uncertain. Moreover, we preferred to leave a more general information represented by *n_participants*.

Unbalancing problem analysis

Once we solved the linear relationship problem, the unbalancing problem was explored, testing Logistic Regression and Decision Tree with the same hyperparameter setting used for the previous problem. Moreover,

we expand the analysis including also at least one non-trivial classification method, in particular, the Neural Network.

This model was cross-validated with the same technique employed for Logistic Regression and Decision Tree. Also for Neural Network, we used mostly the default hyperparameters, only fixing hidden layers to one single hidden layer with 16 hidden units, activation function to 'tanh', optimizer to 'Adam' and number of epochs to 500.

A first experiment was conducted by including also class weights for Logistic Regression and Decision Tree to enhance prediction for the minority class. The class weights combinations tested were the following:

- 0.7 for negative class and 0.3 for the positive one;
- 2 for negative class and 1 for the positive one;
- balanced importance for each class based on class frequency on the available dataset.

The best combination for both classifiers resulted the one that balance the importance of each class based on class frequencies.

Then, random undersampling and oversampling techniques from `Imbalanced-learn` library were applied to the dataset, fixing a random seed for reproducibility. The resulting class proportion changed from 73% for negative class (0) and 26% for the positive class (1) to 50% for both classes.

From these experiments, using, also in this case, approximately default hyperparameters for each classification method, we can observe that without adopting oversampling or undersampling techniques or weighting the classes, the recall reached at most 30% even with non-trivial methods like Neural Network (see its recall in Table 6). Instead, adopting the weighting technique, the recall improved a lot (more than +20%), but precision was very low (between 35% and 45%) (see Logistic Regression and Decision Tree precision and recall in Table 6).

model	**accuracy**	**precision**	**recall**	**f1-measure**
Neural Network	0.76 ± 0.001	0.58 ± 0.006	0.31 ± 0.02	0.40 ± 0.02
Neural Network - oversampling	0.72 ± 0.001	0.69 ± 0.002	0.79 ± 0.008	0.74 ± 0.003
Neural Network - undersampling	0.72 ± 0.003	0.69 ± 0.003	0.79 ± 0.004	0.74 ± 0.003
Logistic Regression - class weights	0.62 ± 0.002	0.35 ± 0.002	0.54 ± 0.005	0.43 ± 0.003
Logistic Regression - oversampling	0.59 ± 0.001	0.60 ± 0.002	0.54 ± 0.002	0.57 ± 0.002
Logistic Regression - undersampling	0.59 ± 0.003	0.60 ± 0.003	0.54 ± 0.006	0.57 ± 0.005
Decision Tree - class weights	0.67 ± 0.004	0.43 ± 0.003	0.80 ± 0.01	0.56 ± 0.002
Decision Tree - oversampling	0.72 ± 0.001	0.69 ± 0.004	0.80 ± 0.01	0.75 ± 0.003
Decision Tree - undersampling	0.71 ± 0.005	0.68 ± 0.006	0.78 ± 0.005	0.73 ± 0.005

Table 6: Mean accuracy, precision, recall, f1-measure and their standard deviation of Logistic Regression, Decision Tree, Neural Network calculated on validation folds weighting classes, oversampling or undersampling

In conclusion, the most appropriate techniques to use are sampling ones, where the recall improved without affecting too much the precision. The difference between the two sampling methods did not seem to be sharp (in some cases, performance differs by no more than 1% or 2%), so undersampling technique was selected. This method was chosen because it reduces the majority class records and it makes it computationally less expensive to do model selection and evaluation on the data set, allowing to test a broader number of models and to explore more advanced validation techniques such as the Stratified 5-Fold Cross-Validation.

4.3 Preprocessed and balanced data set analysis

After exploring linear relationship and unbalancing problems, an extensive analysis, which can be found in `Classification_sel_feature_undersamp.ipynb` notebook, was conducted on the resulting data set preprocessed and balanced with undersampling.

Training and test set split and Stratified 5-Fold Cross-Validation on training set were applied as described on section 4.1. To perform model selection, all the models were cross-validated on training set, using a grid search from Python library `scikit-learn` to determine the best hyperparameter configuration. Once the model selection phase was terminated, for each model the *best hyperparameter configuration* together with an average performance on validation set were obtained (see Table 7). Then, each model with their best hyperparameter configurations was retrained on the whole training set and also tested on the test set.

A large number of models was explored during this predictive task. They are the following: Logistic Regression, K-nearest neighbors, Naive Bayes, Decision Tree, Support Vector Machine, Neural Network, Rule-based Classifier built using RIPPER method, and ensemble models such as Random Forest and AdaBoost.

For the **Logistic Regression** model, the hyperparameter configurations explored were 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga', 'newton-cholesky' as optimizers; C regularization parameter with values 1e-1, 1e1, 1e4; 'l1', 'l2', 'elastic net' as norms of the penalty term or no penalty term; in case elastic net norm was selected, l1 ratio values in the range [0,1] were tested. The **best hyperparameter configuration** is the one including C equals to 0.1, 'l1' as norm of the penalty term and 'liblinear' as optimizer.

For the **K-nearest neighbors** model, the hyperparmeter values analyzed were 5,7,9,15,30 as k; 'auto' and 'ball_tree' as algorithms; Euclidean, Chebyshev, Cosine distance metrics and also Manhattan distance metric, setting Minkowski distance power parameter to 1; finally, uniform or distance-based neighbor weighting functions were tested. The **best hyperparameter configuration** includes k equals to 30, Manhattan as distance metric and distance-based neighbor weighting function.

For the **Naive Bayes** model, the only relevant hyperparameter was the type of model to use, so Gaussian, Bernoulli, Multinomial and Complement Naive Bayes were explored. The **best** type of model results Gaussian Naive Bayes.

For the **Decision Tree** model, the hyperparameter configurations explored were Gini, Entropy and Log loss criteria to measure the quality of the split; as maximum depth of the tree, 5, 10 or expansion until all leaves are pure or contain less than minimum samples were analyzed; square root, or log to the second base of the number of features, or directly the number of features were considered as maximum number of features to consider when looking for the best split; then, 2, 4, 8, 16 as minimum number of samples required to split an internal node, and 1,2,4,8 as minimum number of samples required to be at a leaf node were used. The **best hyperparameter configuration** includes Log loss as criterion to measure quality, 10 as maximum depth of the tree, the number of features itself as maximum number of features to consider for the split, 16 as minimum number of samples to split an internal node, and 4 as minimum number of samples in a leaf node.

For the **Support Vector Machine** model, the hyperparameter values considered were linear, RBF, sigmoidal, polynomial as kernels; 1e-2, 1e1, 1e2 as values for the C regularization parameter; 'scale' or 'auto' technique to calculate the gamma kernel coefficient; 2,3,5 as degrees of polynomial kernel function, if polynomial kernel was selected. The **best hyperparameter configuration** is the one including RBF kernel, C equals to 10 and scale strategy to calculate the gamma kernel coefficient.

For the **Rule-based classifier** built using RIPPER algorithm, the hyperparameter configurations searched were 1, 3, 5 as number of RIPPER optimization iterations; 0.5 and 0.6 as proportion of training set used for pruning. The **best hyperparameter configuration** includes 1 optimization iteration and 0.5 as proportion for pruning.

For **Neural Network**, an extensive search of hyperparameter configuration was performed. In particular, a single hidden layer with 4, 16, 32, 64 hidden units was tested. Moreover, two hidden layers with respectively 32 and 16 hidden units and 16 and 32 hidden units were tested, and finally three hidden layers with 64, 32, 8 hidden units were explored. We also tested *Tanh* and *Sigmoid* as activation functions, Adam and Stochastic Gradient Descent as solvers and 1e-5 and 1e-8 as values for alpha regularization term (L2). The learning rate was fixed to constant and 0.02 and 0.005 were explored as its values. Finally, also 8,16,32 as batch size and 200 and 500 as maximum number of epochs were considered. The **best hyperparameter configuration** consists in one hidden layer of 64 hidden units with *Tanh* as activation function, Adam solver, constant learning rate of 0.005, batch size of 32 and 500 as maximum number of epochs.

For **Random Forest**, we considered the same hyperparameter values tested for Decision Tree, and also 32, 64 and 128 trees in the forest and whether to use or not bootstrap to build the trees. The **best hyperparameter configuration** includes 128 trees trained with no boostrap, Gini as criterion to measure quality, 15 as maximum depth of the trees, the log to base two of number of features itself as maximum number of features to consider for the split, 4 as minimum number of samples in a leaf node.

For **AdaBoost**, 10, 30 and 100 base estimators, consisting in Logistic Regression, Decision Tree or Support Vector Machine models set to their best hyperparmeter configurations, were tested. Moreover, 0.1 and 0.05 learning rate values were explored. The **best hyperparameter configuration** is the one including Decision Tree models as base classifiers and learning rate of 0.1 .

4.4 Conclusions

As we can see from Table 7, the model achieving the highest values on validation metrics is **AdaBoost**. Not far from it, other remarkable results were obtained by Random Forest and Neural Network.

Then, all models have also been tested on the test set, confirming AdaBoost, Random Forest and Neural Network as the top three models (see Table 8).

Analyzing both the final validation and test evaluations, we can observe that, deleting *n_injured* and *n_unharmed* attributes, the task became particularly challenging for our classifiers, even if we balanced the

Best configuration model	accuracy	precision	recall	f1-measure
Logistic Regression	0.59 ± 0.003	0.60 ± 0.003	0.54 ± 0.006	0.57 ± 0.005
KNN	0.69 ± 0.005	0.68 ± 0.005	0.70 ± 0.007	0.69 ± 0.005
Naive Bayes	0.57 ± 0.002	0.61 ± 0.003	0.41 ± 0.008	0.49 ± 0.006
Decision Tree	0.71 ± 0.005	0.69 ± 0.008	0.78 ± 0.008	0.73 ± 0.003
SVM	0.68 ± 0.004	0.66 ± 0.003	0.74 ± 0.007	0.70 ± 0.003
Rule-based classifier (Ripper)	0.55 ± 0.007	0.69 ± 0.01	0.19 ± 0.03	0.30 ± 0.02
Neural Network	0.72 ± 0.004	0.70 ± 0.01	0.79 ± 0.02	0.74 ± 0.007
Random Forest	0.73 ± 0.004	0.70 ± 0.004	0.80 ± 0.006	0.75 ± 0.003
AdaBoost	$\mathbf{0.73 \pm 0.003}$	$\mathbf{0.70 \pm 0.004}$	$\mathbf{0.80 \pm 0.002}$	$\mathbf{0.75 \pm 0.003}$

Table 7: Mean ***validation*** accuracy, precision, recall, f1-measure and their standard deviation for the best hyperparmeter configuration of all method explored in preprocessed and balanced data set analysis.

Best configuration model	accuracy	precision	recall	f1-measure
Logistic Regression	0.62	0.58	0.60	0.57
KNN	0.68	0.65	0.69	0.65
Naive Bayes	0.66	0.57	0.58	0.57
Decision Tree	0.66	0.66	0.71	0.64
SVM	0.65	0.64	0.69	0.63
Rule-based classifier (Ripper)	0.74	0.64	0.54	0.52
Neural Network	0.69	0.68	0.72	0.66
Random Forest	0.69	0.68	0.74	0.67
AdaBoost	**0.70**	**0.69**	**0.74**	**0.68**

Table 8: ***Test*** accuracy, precision, recall, f1-measure for the best hyperparmeter configuration of all method explored in preprocessed and balanced data set analysis.

dataset. In fact, the best model, AdaBoost, scores less than 75% of accuracy on the validation set and 70% on the test set, even with the best hyperparameters. However, we can observe that balancing the dataset, the recall is around 80% on the validation set and 74% on the test set for the best models, which is a not bad score. In our opinion, to obtain better results on this dataset, it could be useful to collect a larger number of records for each class and continuing testing more and more configuration of hyperparameters, even with more powerful and efficient machines than our laptops.

5 Explanation Analysis

The explainability phase aims to provide both global and local explanations for the decisions made by the models. During this phase, LIME and SHAP methodologies are used in order to explain black box models. Instead, EBM is employed as interpretable model by design, namely a model specifically designed to explain the reasoning behind each prediction it generates. The difference between global and local interpretability is that the former seeks to comprehend the overall logic of a model, while the latter gives explanations of a specific decision of a model.

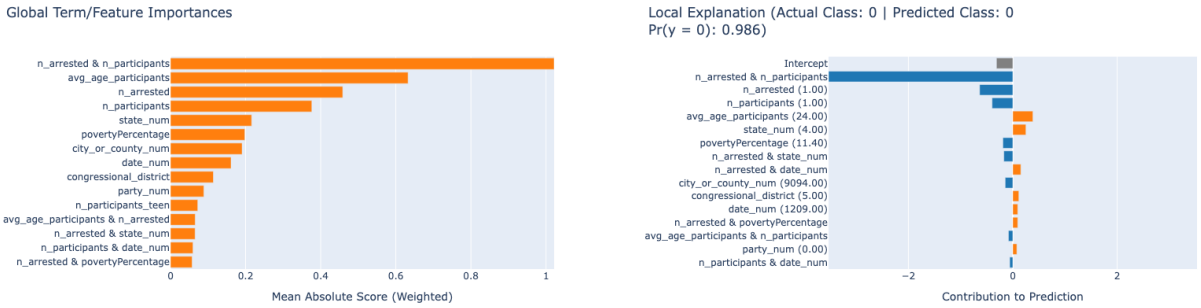
5.1 Interpretable by design models

Explainable Boosting Machine

The explainable by design method used is the **Explainable Boosting Machine (EBM)** from the `interpretml` library. The model is trained on the under-sampled version of the dataset split as seen in the predictive analysis in Section 4.1. The first step is to compute a grid search in order to find the best hyperparameters based on accuracy, precision, recall, f1-measure metrics.
The performances of the model are reported in Table 9, which are in line with the performances that we obtain for the models trained in the predictive task.

dataset	Accuracy	Precision	Recall	f1
Train set	0.73	0.71	0.80	0.75
Test set	0.68	0.68	0.73	0.68

Table 9: Mean validation performances of EBM in term of accuracy, precision, recall, f1



(a) EBM global explanation

(b) EBM local explanation

Figure 23: EBM explanations

A global explanation is then run on the training set (see Figure 23a), from which we can observe that the most important attributes are *avg_age_participants*, *n_arrested* and *n_participants*. It can also be visualized how much each feature contributes to local predictions on test set instances, as shown for the seventh test sample in Figure 23b. In particular, in the same Figure, it can be seen that the features contributing more for the prediction of the record class are also the most important ones globally, even if they might change according to the selected instance, since we are talking about local explanations.

5.2 Black box explanations

In order to explain the black box models, two different methods have been used, namely **Shapley Additive Explanations (SHAP)** and **Local Interpretable Model-Agnostic Explanations (LIME)**. They are widely used for model interpretability, as they can provide human-understandable explanations for the predictions of complex machine learning models. SHAP and LIME use different approaches to generate explanations: the former is based on the concepts of Shapley values from game theory, while the latter uses local regression on the neighbors of a certain instance.

The black box models that we tried to explain using LIME and SHAP techniques are Random Forest, Neural Network, Support Vector Machine. In particular, we trained each of these black box model with the best hyperparameter configuration obtained after the grid searches performed in the predictive analysis.

5.3 Shap

We used the SHAP `TreeExplainer` function in order to generate the Shap values for the Random Forest, while, for the other models, the `KernelExplainer` function has been used. These functions were tested on the first 200 instances of the test set, split according to the predictive analysis preprocessing in Section 4.1. We tried only 200 test instances due to the high computational cost requested by these functions.

The most relevant results that we will discuss below refer to explanations related to the Random Forest model.

Global explanation

The first plot that we were interested in is the summary plot on the Figure 24. It represents a global explanation, illustrating the average influence of different features on the predictions of the Random Forest model. The most important features are *avg_age_participants*, *n_arrested*, *n_participants*, *state_num* (the discretized version of the state name) and *povertyPercentage*. Instead, some features such as the number of child and teen are not helpful to classify the majority of instances. From this plot, we can see that SHAP is in line with the explainable by design model we have tested, because these feature have similar importance also in the EBM global explanation.

The plot in Figure 25a is another global plot. It concatenates various local explanations to return a global one. The plot is filtered by the feature *n_arrested*, showing its combined effect with all other features on the model prediction. It reveals that the combination of *n_participants* and *n_arrested* has the most significant impact on the model prediction. In particular, as *n_arrested* increases, it tends to have a negative impact on the positive label; while an increase in *n_participants* has a positive impact on the positive label. We consider this observation reasonable, because we observed that for 77% of instances where *n_arrested* is greater than 3, there are no deaths.

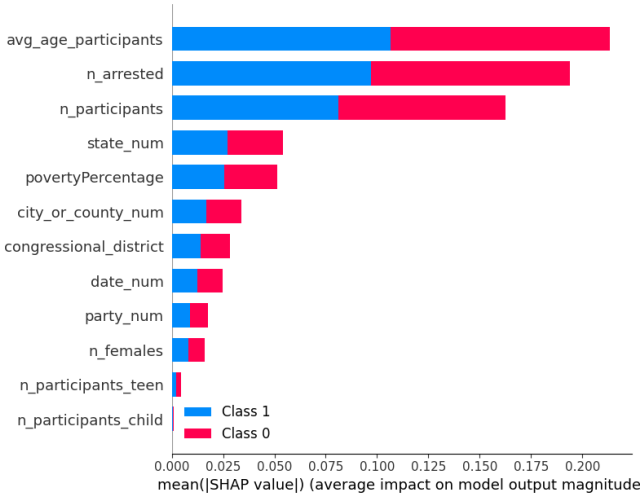
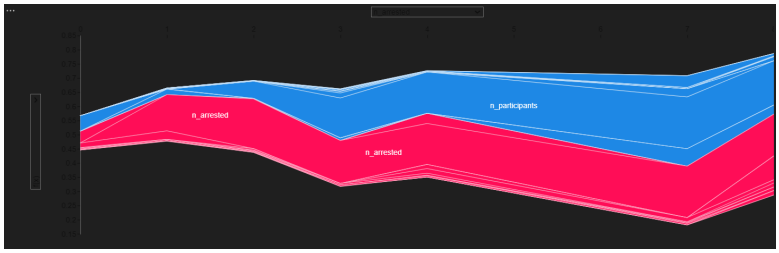
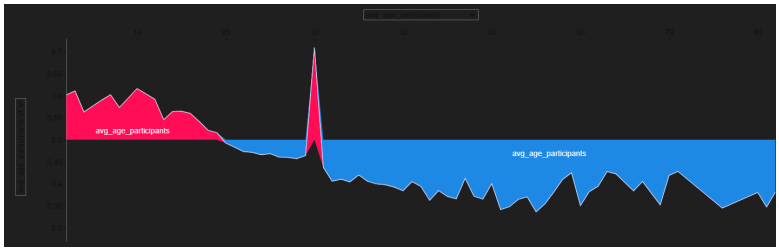


Figure 24: Summary plot (Random Forest)



(a) Summary plot filter n_arrested on f(x)



(b) Summary plot filter *avg_age_participants* with relative effect

Figure 25: Summary plots (Random Forest)

In Figure 25b, the plot is filtered by *avg_age_participants*, showcasing its relative effect on the model prediction. It becomes apparent that when *avg_age_participants* is lower than 20, this feature negatively impacts the positive label. This aligns with the fact that we observed that 80% of instances with such values in the dataset result in no killed people. On the other hand, for values greater than 30, this feature demonstrates a positive impact on the *isKilled* class label.

Local explanation

In order to explore the local explanation capability of SHAP, we analyzed a specific test instance, the instance number 10, from the 200 explained by the method.

The plot in Figure 26 represents the contribution of each feature for the classification of instance number 10. In this case, the effective label, 0, is correctly predicted by the black box model (Random Forest). First of all, we can observe that the average prediction across all rows is 0.5: this is the result of using under sampling technique for training the Random Forest. Then, from this plot, we analyzed the three most important features as follows:

- *avg_age_participants* (equals to 30) decreases the confidence for label 1 by 0.23 with respect to the mean prediction over the training set. In fact, as indicated in the summary plot (Figure 24), this feature generally contributes to an increase in confidence for predicting label 0;

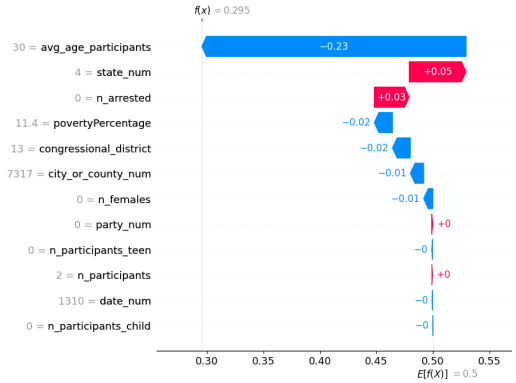


Figure 26: Instance 10 SHAP explanation (Random Forest)

- *state_num* (equals to 4), instead, increases the confidence for label 1 by 0.05, showing us how this value for this feature is correlated with an higher percentage of incidents involving killed people;
- *n_arrested* (equals to 0) increases the confidence for label 1 by 0.03. This specific value differs from the behavior expressed by this feature in the summary plot (Figure 24).

Examining the original dataset, we observe that the rows, in which the combination of the most important feature values of the previously examined test instance is present, show for 80% of the time that no person is killed. This confirms the reason why the model predicted the label 0 for this instance, which turned out to be accurate. In fact, the faithfulness score for this instance is 0.85, indicating a high level of alignment between the ordering of features and their actual importance in the classification. However, the monotonicity score is False, suggesting that the model performance does not strictly increase monotonically with the features considered relevant for SHAP.

5.4 LIME explanation

Differently from SHAP, LIME is a local-only method. It's been used to explain the contribution of each feature to the prediction of some test samples. Since it's a local method, the results may change if different samples are used and the explanation is dependent on the random generation of the neighbors of the considered instance, which might affect the results. We briefly examined two test instances (6 and 10) explained by LIME for each black box methods (Random Forest, SVM, Neural Network) to explore the explanation capability of this method.

Random Forest

For both instances, number 6 and 10, the actual labels, 1 and 0 respectively, are correctly predicted by the model, and the Lime explanation is aligned with the model predictions, as we can see in Figures 27 and 28.

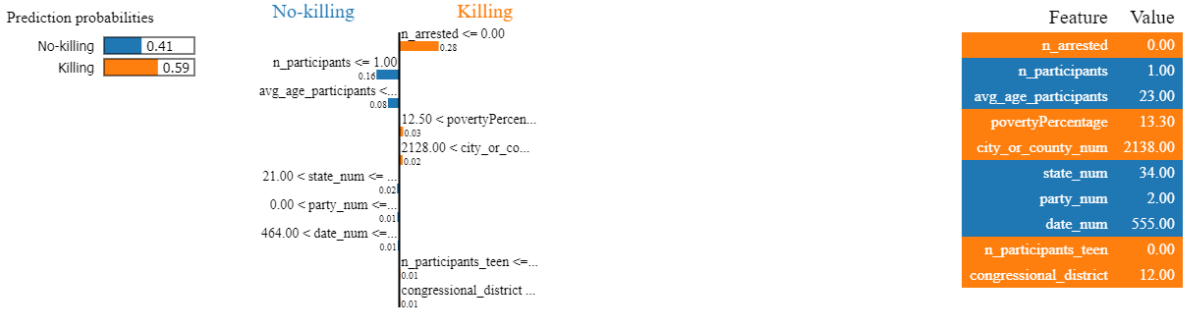


Figure 27: Random Forest LIME explanation instance 6

SVM

For the test instance number 6, despite the SVM committed a misclassification, the probability of it is only 0.53. This shows that the classifier appears to be hesitant in assigning a definitive class to this instance, as the

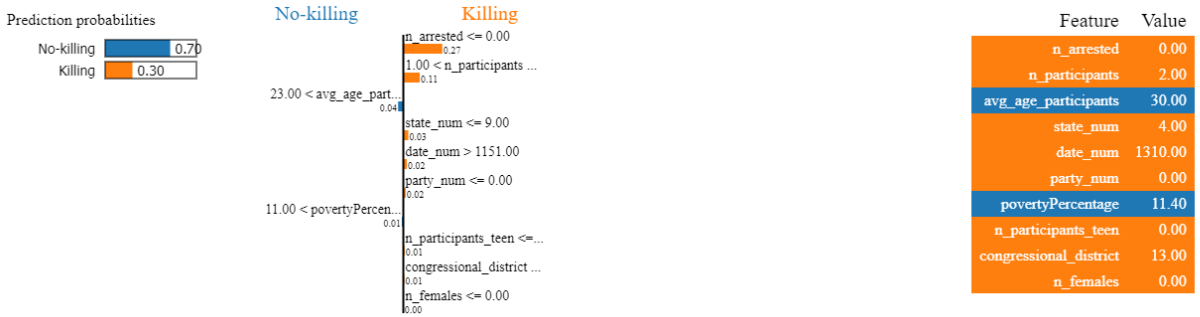
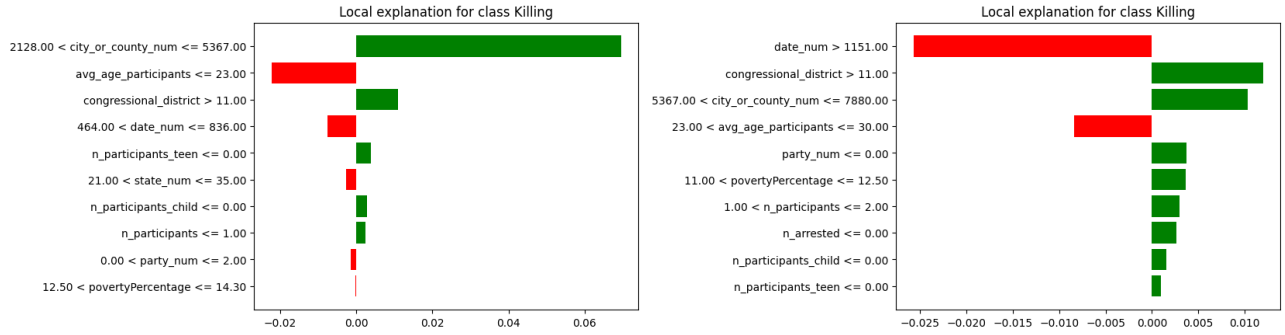


Figure 28: Random Forest LIME explanation instance 10



(a) SVM lime instance 6

(b) SVM lime instance 10

Figure 29: SVM lime explanations

confidence difference between the classes is really small. The explanation given by LIME, shown in Figure 29a, reveals that the feature *city_or_county* is the most significant one for this instance, contrary to what we saw for the random forest model explanation. This discrepancy highlights the potential source of the misclassification. At this point, we found the corresponding effective value of *city_or_county* (namely *Columbus*) for the instance number 6, and we examined the values of the other test records sharing the same city. We observed that most of the records have *isKilled* class set to 0 (75%). This might have influenced the model incorrect guess, because SVM gives importance almost only to this feature.

The previous observation is confirmed also by the good LIME faithfulness score of 0.73 for the instance number 6. In fact, this indicates a good degree of alignment between the ordering by relevance of features and their actual importance in the classification.

For the test instance number 10, instead, the model performs a correct prediction, but also in this case with uncertainty, showing only 0.54 of confidence. The faithfulness metric and monotonicity metric both return low values (0.488 and false, respectively), so the explanation doesn't really reflect the real importance of these features. So this specific explanation is not helpful to understand the choices made by the model to predict the incident class. The plot of the explanation can still be viewed in Figure 29b.

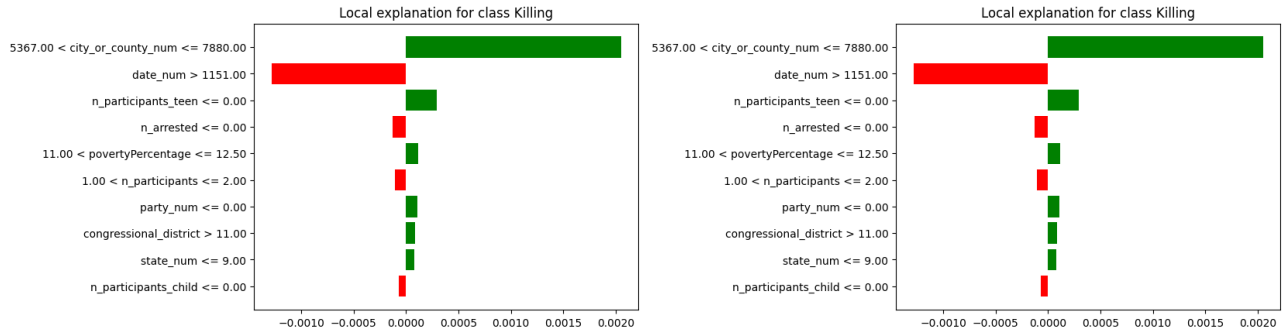
Neural Network

The third and last model explained is the Neural Network. In the picture in Figure 30a, we can see the ranking of the most important features for the instance number 6. Again, although the prediction is correct, the confidence (0.51) of the prediction is not really high with respect to the negative class. This behavior is shared by each black box model analyzed, so this shows us that the instance is very hard to be classified. For instance number 10, the classifier is more sure (0.7) about the classification and correctly classifies the label as non killing.

The faithfulness score for the instances 6 and 10 is moderately high (0.87 and 0.66 respectively) and the monotonicity is True for both instances, showing that we can trust the explanation of this method. The relevant features of both examples can be seen in Figures 30a and 30b.

5.5 Results

In Table 10, the results obtained from LIME and SHAP are compared, showing for both methods the average and the standard deviation of the faithfulness over the first 200 samples of the test set. In the same Table, we



(a) Neural Network LIME instance 6

(b) Neural Network LIME instance 10

Figure 30: Neural Network LIME explanations

can observe the proportion of the test samples for which the monotonicity has been evaluated to True. The best explanation approach between LIME and SHAP do not emerge clearly if we look only at the faithfulness values, because they are pretty low for all the black box models, both for LIME and SHAP, but the standard deviation is high for the former, which means that for some instances the faithfulness is good and for others is bad, and very low for the latter, which means that the Shap values are close to each other for all instances and they are not reliable. Additionally, looking only at the monotonicity values calculated for Random Forest and SVM prediction explanations, no clear winner can be elected. The only sharp difference between LIME and SHAP emerged from the monotonicity results for Neural Network, in which we can see that SHAP approach resulted the best explanation method.

model	**LIME faithfulness avg**	**SHAP faithfulness avg**	**LIME faithfulness std**	**SHAP faithfulness std**	**LIME monotonicity**	**SHAP monotonicity**
RF	0.056	-0.172	0.424	0.457	1/200	5/200
SVM	-0.069	-0.307	0.588	0.739	0/200	0/200
MLP	0.285	-0.997	0.790	2.2e-16	124/200	200/200

Table 10: Average values and standard deviation for Faithfulness and proportion of true values for Monotonicity over the first 200 instances of the test set