Data Mining Report

https://github.com/biancaziliotto/DataMiningProject
Academic year: 2023/2024
Course code: 309AA

Filippo Biondi, Andrea Roncoli, Bianca Ziliotto

8 January 2024

1 Data understanding and preparation

1.1 Dataset description

Our dataset is composed of 3 `.csv` files:

- `poverty.csv` (884 rows × 3 columns), containing the poverty rate of all American States in different years;
- `house_district.csv` (10441 rows × 6 columns), containing information about the outcome of the elections in every congressional district in different years;
- `incidents.csv` (239677 rows × 29 columns), containing information about gun incidents in the USA. The complete list of attributes is provided in the next section.

1.2 Data quality assessment

In this section, we focus on assessing the quality of our data: after detecting and isolating data quality issues, we propose solutions to deal with missing or erroneous information.

1.2.1 Duplicates and missing values

At first, we identified and removed duplicates from the dataset. We dropped 296 duplicate rows, resulting in a dataset of 239381 incidents. Then we counted the number of missing values in the dataset: for each attribute, the percentage of missing values is reported in Table 1. In the data preparation phase, we will discuss how to reasonably fill missing values (Section 1.3.2).

1.2.2 Syntactic accuracy

After counting missing values, we checked the syntactic accuracy of the values of each of the attributes. The percentage of syntactical errors for each attribute is reported in Table 1. In all these cases, we encountered non-numerical strings where we were expecting numerical values, so we considered them as missing values and removed them from the dataset.

Attribute	DType	Missing values	Syntactically wrong values	Semantically wrong values
Date	object			9.609 %
State	object			
City or County	object			
Address	object	6.883 %		
Latitude	float64	3.305 %		
Longitude	float64	3.305 %		*5 incidents*
Congressional District	int64	4.983 %		
State House District	int64	16.177 %		
State Senate District	int64	13.491 %		
Participant Age 1	int64	38.509 %		0.001 %
Participant Age-Group 1	object	17.573 %		
Participant Gender 1	object	15.171 %		
Minimum Age Participants	int64	31.208 %	2.400 %	4.900 %
Average Age Participants	float64	31.134 %	2.467 %	4.920 %
Maximum Age Participants	int64	31.170 %	2.455 %	4.884 %
N. Participants Child	int64	17.570 %	0.002 %	0.005 %
N. Participants Teen	int64	17.570 %	0.003 %	0.005 %
N. Participants Adults	int64	17.570 %	0.001 %	0.006 %
N. Participants Males	int64	15.171 %		
N. Participants Females	int64	15.171 %		
N. Killed	int64			
N. Injured	int64			
N. Arrested	int64	11.526 %		
N. Unharmed	int64	11.526 %		
N. Participants	int64			10.376 %
Notes	object	33.803 %		
Incidents Characteristics 1	object	0.136 %		
Incidents Characteristics 2	object	40.782 %		

Figure 1: Quality assessment table.

1.2.3 Semantic accuracy

After ensuring each value type was consistent with the type of the corresponding attribute, we focused on assessing the semantic accuracy of the values. In this phase, the following issues were identified and addressed:

- **Dates of the incidents**. The incidents were collected from 1st January 2013 to 31st March 2018. Hence, each date outside this range is considered a semantic error. For a significant percentage (9.609%) of incidents, the reported year is between 2028 and 2030, hence necessarily wrong. In order not to lose an important number of values, we used the notes and incident characteristics that we found in the dataset to look for information about the true date of the incident on the Internet. As we couldn't check each incident manually, we randomly selected 3 incidents per year (from 2028 to 2030) and identified a probable mapping:

2028 → 2013; 2029 → 2014; 2030 → 2015.

- **Coordinates of the incidents**. In a first simple analysis, we identified as semantic error incidents with coordinates corresponding to locations outside the U.S.A. In particular, we identified only 5 incidents with an incompatible longitude value. However, we noticed that changing the sign of the longitude brought the location back to the correct city or county. Moreover, we noticed that the 5 incidents with this kind of error were all TSA actions near airports, which enforced our belief that they are affected by the same sign error. For this reason, we decided to keep this information by simply switching the sign of the longitude value. As for the consistency between coordinates and states, a visual check is provided in Figure 3.

- **Age attributes**. We assessed the semantic accuracy of age values by constraining the allowed range: we first removed all negative values, and we symbolically kept non-negative values up to 116 (the age of the oldest man on Earth). We also checked the following constraints:

  - Minimum age $\leq$ Average age $\leq$ Maximum age (100 % `True`)
  - Minimum age $\leq$ Participant age 1 $\leq$ Maximum age (100 % `True`)

  Notice that percentages are calculated with respect to the set of rows where the values exist and are syntactically correct.

- **Group composition**. We constrained all the attributes about the number of participants (e.g. N. males) to be non-negative, removing all the negative values. As for the total number of participants, we classified zero values as semantic errors, as we assumed that each incident involves at least one participant. We also constrained the total number of participants to be larger or equal to the number of participants belonging to a specific category, e.g. number of children, number of injured, etc., removing these latter attributes in case the inequality did not hold. Moreover, we did the following checks about the group composition:

  - N. participants = N. participants child + N. participants teen + N. participants adult (90 % `True`)
  - N. participants = N. males + N. females (92 % `True`)
  - N. participants = N. killed + N. injured + N. unharmed + N. arrested (95 % `True`)

  Even if these equations were not always satisfied, we decided not to consider such values as missing values because we estimated that they could still contain useful information despite not being exact. However, in the first two cases, we corrected these values in order to satisfy the constraint by adding some participants to each category without altering the original proportions. E.g.: if N. males is 80% of (N. males + N. females) but (N. males + N. females) < N. participants, we set N. males to the 80% of N. participants (and the rest will be females).

1.2.4 Timeliness and balancing

The dataset is not up to date, as no incidents were collected after 31st March 2018. Therefore, eventual trends or distributions that will be highlighted from our analysis are valid up to 2018 and cannot be applied to the following years without further analysis of updated data.

If we restrict to the temporal range 2013-2018, we observe that there are significantly fewer recordings in 2013, which unbalances the dataset towards the following years. Apart from this evident disproportion, we assume there aren't other sources of imbalance, as duplicates were removed, and we have no reason to doubt that some categories of incidents are more represented than others.

1.3 Data preparation

1.3.1 Data integration

After correcting wrong year values, we were able to integrate the 3 datasets by exploiting the information about the year, state, and district of each incident. Thus, we added state poverty percentage and district election outcome to each incident as extra attributes.

1.3.2 Filling missing values

Having identified all missing values (including wrong values that we couldn't correct), we used some strategies to fill these values without modifying the single (or joint) variables distributions. The most accountable way to fill in missing values would be fitting a regressive model that predicts a certain attribute given all the other attributes. The solution that we decided to adopt is much simpler and requires no training: for each missing attribute, we randomly sample a value from the same column of the dataset, eventually restricting the dataset to a subset of samples that share one or more of the other attributes. In this way, we are not brutally modifying the variables distribution (e.g. always filling with the same value), and we are still able to manually introduce some bias and exploit correlations (by sampling from a meaningful subset of the dataset).

- **Coordinates.** We replaced the missing coordinates by sampling a latitude-longitude pair from the incidents that happened in the same city (or state, in case the city was missing).
- **Age.** In case of missing information about the minimum, average, and maximum age, we filled the missing values by sampling from the incidents that occurred in the same state. Notice that the inequality constraints checked in Section 1.2.3 still hold, as we are picking minimum, average, and maximum age from the same sampled incident.
- **Number of participants and group composition.** In cases of missing information about the number of participants, we replaced such information by sampling from the incidents that happened in the same state and year. As for group composition, the goal was to fill in missing values while taking into account that some characteristics (e.g. male/female, adult/teen/children) are considered mutually exclusive and collectively exhaustive. For this reason, we computed the ratio of participants for each category (e.g. males ratio = N. males / N. participants). For each incident where this information was missing, we filled in missing values by randomly sampling the ratios from incidents that happened in the same state, and then we multiplied such ratios by the total number of participants in the incident, rounding to the closest integer in such a way to preserve sum-to-one constraints.
- **Party and votes.** For incidents where the outcome of political elections in the district of the accident was not available, we filled it with the party, candidate votes, and total votes

sampling from incidents that happened in the same city, year, and state (we used a subset of these attributes where it was not possible to find a correspondence with all of them).

1.3.3 Additional attributes

Some extra features were added to capture hidden or derived information in order to help the clustering and classification tasks:

- The month of the incident to eventually capture different criminality rates in different periods of the year.
- The population of the state of the incident (in the year of the incident). This will prove useful to visualize the criminality of each state in terms of the density of incidents rather than the absolute number (Section 1.4, Figure 4).
- A binary column stating whether the city where the incident happened has more than 700k inhabitants: this could be particularly informative from the moment that we will decide to drop the attribute of the city. Notice that very populous cities ($\geq$700k inhabitants) are only 34, and 23% of the incidents happened in one of these cities.
- Another binary feature that states if either the notes or the incident characteristics contain the words "shoot" or "shooting". This additional feature is introduced to provide some useful information for the classification task before eventually removing the notes. (Notice that we didn't use more explicit words such as "dead" or "suicide" in order not to overlap with the actual target of the classification task).

1.4 Variables distribution

In this section, we provide a preliminary visualization of the distribution of different variables. In Figure 2, we show the political color of the congressional districts where the incidents happened. The temporal division is due to the fact that political elections are held every 2 years. In Figure 3, we can simultaneously visualize the geographic distribution of the incidents and the respective states. The lower concentration of incidents in the central states must, of course, be interpreted considering the low population density of the area. A representation of the distribution of incidents normalized on the population is provided in Figure 4.

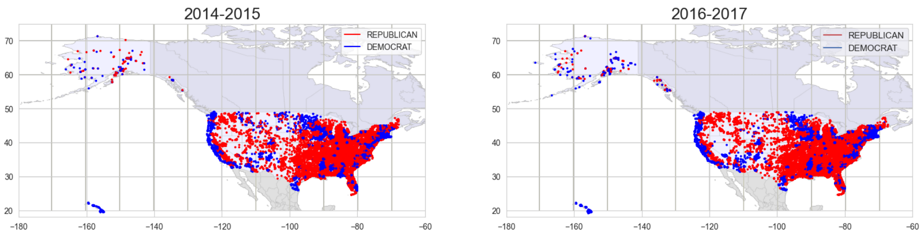


Figure 2: Geographical location and political color visualization.

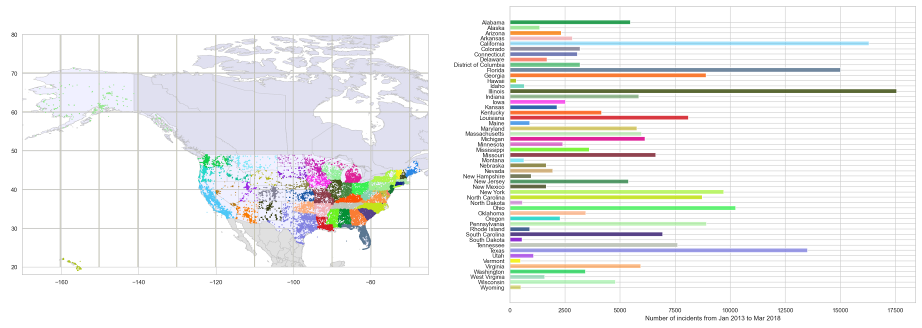


Figure 3: Geographical location and states visualization.

Figure 4 shows the geographical distribution of the incidents together with the poverty percentage of the year and state of the incident. As we can observe, the state with the highest density of incidents has also a very high poverty rate. Notice that in the histogram we are considering for each state the average poverty rate over different years (2013-2018).

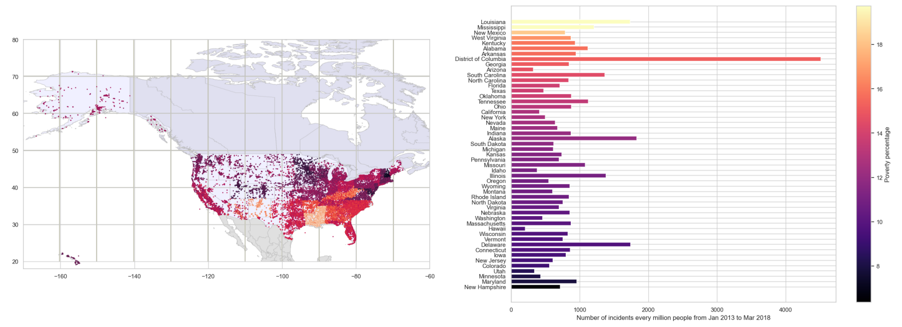


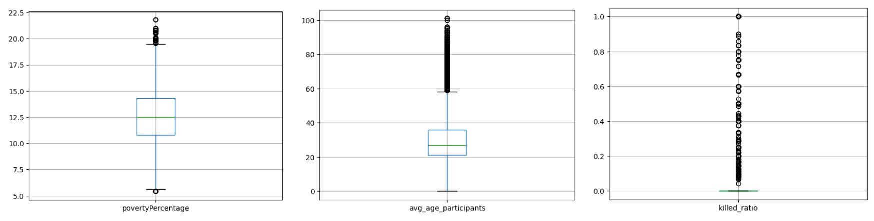
Figure 4: Geographical location and poverty percentage visualization.

1.4.1 Outlier removal

When looking at variables distributions, we are particularly interested in addressing outliers, i.e. data points significantly deviating from the majority. Such values can introduce noise and distort the patterns identified by clustering and classification algorithms. Therefore, a pragmatic outlier removal strategy was implemented to improve the accuracy and robustness of the algorithms.

The process of outliers removal involved a practical examination of data points using statistical techniques such as the interquartile range (IQR) method. Points falling outside predefined thresholds were considered potential outliers and were subsequently examined to determine if they were representative of an interesting phenomenon or just uninformative noise to be discarded.

In figure 5, we report box plots of some attributes to give examples of choices to keep or remove out of distribution points, motivating our decisions. As we can see in Figure 5a, the outliers identified by the boxplot are very close to the whiskers and are still realistic. We thus decided to retain all the values as we consider the poverty percentage a key attribute of our analysis. In



(a) (b) (c)

Figure 5: Boxplots for Poverty Percentage, Average Age, and Killed Ratio.

Figure 5b, we can observe instead a situation where we have values far from the whiskers and very unrealistic, with average ages of more than 90 years old, up to 100+, so we decided to remove these points considering them as outliers. Finally, in Figure 5c, we can see how the distribution is concentrated in 0, highlighting all different values as outliers. This is simply due to the fact that (luckily) most of the shootings ended up having no casualties. Despite this, information on killing during the shootings is very precious to the analysis (and crucial for classification as it will become the target information). As dropping outliers would mean dropping all lethal incidents, which is not desirable, we decided to keep all of these points.

1.4.2 Pairwise correlation

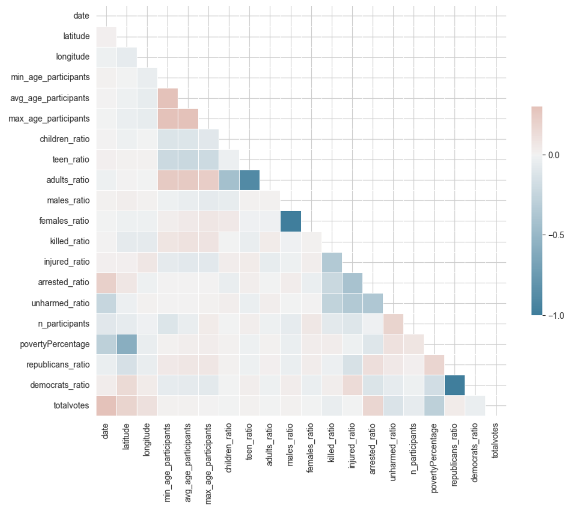


Figure 6: Pairwise correlation matrix

Figure 6 shows the pairwise correlation among numerical attributes in the dataset. As we can observe, the only correlation values that are high enough to remove the attributes are the ones that we expect to satisfy a sum-to-one constraint. As we have forced this constraint to hold for each sample, some attributes can be considered completely redundant and will then be dropped in the data preparation phase. Apart from these stronger correlation values, some slightly positive or negative correlations between some attributes can be checked intuitively. For example, we can see that minimum, maximum, and average age are positively correlated with each other. Another value that can be noticed is the negative correlation between latitude and poverty percentage, suggesting that southern states are poorer than northern states. Finally, the percentage of arrested participants is positively correlated with the date, suggesting that arrests have increased, while the poverty percentage is negatively correlated with the date, representing a slight temporal decrease of this indicator.

1.5 Data representation

The last step of data preparation consists of finding potentially better representations of numerical attributes and providing an encoding for categorical features:

- **Group composition**. The attributes of the group composition (e.g., N. males) were normalized on the number of participants, replacing absolute values with ratios (e.g., N. males / N. participants). Notice that such ratios hold sum-to-one constraints.
- **Number of votes**. We decided to remove the column of candidate votes, and we replaced it with two derived attributes: ratio of republican votes and ratio of democrat votes (assuming sum-to-one constraint).
- **Date**. We assign an integer (1) to January 1^{st} 2013 and add 1 every following day up to March 31^{st} 2018. The date is thus encoded in an integer in [1,1852]. This allowed us to drop the year as a redundant feature.
- **State**. Information about the state is encoded in a one-hot vector of length 51 (including the District of Columbia).
- **Month**. The month variable is also categorical, but unlike the state, it cannot be encoded in a one-hot vector since the distance among different months is not the same for each pair of months. However, encoding the month with an integer in [1,12] wouldn't be correct either since, for example, December and January would have distance 11, rather than 1 as it should be. In fact, this attribute should not encode temporal information but rather capture seasonality, possibly highlighting the role of atmospheric temperature or holiday periods. For this reason, our solution consists of mapping 12 months at equal distances on a circle of radius 1 and then encoding the corresponding angles using their sine and cosine.
- **Party**. We decided to encode the party as a binary variable, thus without adding any column. We unified the class DEMOCRATIC-FARMER-LABOUR with DEMOCRATS, and we assigned 0 to DEMOCRATS and 1 to REPUBLICANS.

1.6 Dimensionality reduction

We decided to drop some features that are useless or intractable for all the tasks that we will address.

- Features about a randomly selected participant;
- Redundant features that can be inferred thanks to sum-to-one constraints or high correlation (females ratio, children ratio, unharmed ratio, democrats ratio);
- Notes and incident characteristics;
- Address, congressional district, state house district, state senate district.

At the end of preprocessing, the final list of features is the following: *state*, *city or county*, *latitude*, *longitude*, *min age participants*, *avg age participants*, *max age participants*, *teen ratio*, *adults ratio*, *males ratio*, *killed ratio*, *injured ratio*, *arrested ratio*, *n participants*, *povertyPercentage*, *party*, *totalvotes*, *shooting*, *month*, *republicans ratio*, *democrats ratio*, *state population*, *populous city*. Some of these features will be removed at the beginning of the following sections if considered of little or no use for specific tasks.

2 Clustering analysis

In this section, we conduct an analysis of the dataset using various clustering techniques (K-means, DBSCAN, hierarchical clustering, and X-means) to reveal hidden structures. In particular, we aim to discover spatial, temporal, social, and political patterns in gun incidents, identifying meaningful clusters. The strengths and weaknesses of different clustering algorithms will be discussed in the following paragraphs.

2.1 Preprocessing for clustering

2.1.1 Dimensionality reduction

Our dataset exhibits a high-dimensional nature due to the inclusion of numerous variables and the one-hot encoding of the *state* attribute. It is known that high-dimensional datasets pose several challenges to clustering algorithms. In fact, the increased sparsity of data points in high-dimensional spaces makes the distinction between clusters becomes less apparent; furthermore, the proximity between data points becomes harder to discern, leading to suboptimal clustering results.

Recognizing that not all features contribute equally to the identification of meaningful patterns, we examined each variable in the dataset and decided to drop the following features for the reasons here explained:

- The minimum and maximum age of the participants, since they are correlated to the average age of the participants and provide a less interesting insight into the nature of the incident, being more influenced by stochasticity.
- Because children-ratio represents a very small portion of the dataset, we decided to consider children and teens together, so due to the sum-to-one constraint, only the information about the adult-ratio is sufficient.
- The states and cities/counties, as the dimension introduced by one-hot encoding was elevated, and the information brought by this variable was partially conserved with other attributes (such as geographical encoding with longitude and latitude).

2.1.2 Scaling

Before applying all our clustering algorithms, we performed a min-max scaling (of each attribute), ensuring that all the values for every attribute were enclosed in the range [0, 1] in such a way that every attribute could contribute equally to the clustering results, independently from its original scale.

2.2 K-means

2.2.1 Choice of K

In our clustering analysis, we used the Elbow to determine the optimal number of clusters (k) in the k-means algorithm. This method allows us to strike a balance between model complexity and goodness of fit. Inspecting the plot of Sum of Squared Errors (SSE) across various k values, compared with Silhouette score and Davies-Bouldin index, we identified a good trade-off between capturing meaningful patterns in the data and the risk of overfitting (Figure 7). In particular, we found $k = 8$ to be a reasonable number of clusters.

2.2.2 Cluster analysis

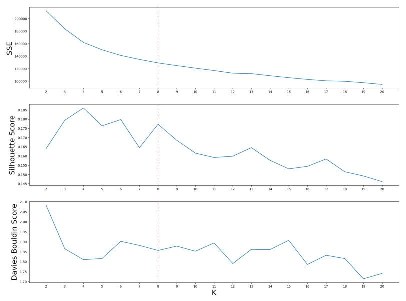


Figure 7: K-means: combined plot of SSE, Silhouette, and Davies-Bouldin over the number of clusters.

Having established the optimal value for k, we applied the K-means algorithm to the dataset. The subsequent analysis of centroids and clusters provided valuable insights into the underlying patterns. The cluster centroids, calculated as the mean of the feature values for each cluster, served as a summary of the characteristics of the data points within the clusters. In Figure 8a, all the distances between these centroids are reported, showcasing a well-distributed set in the space, thereby increasing our confidence in the clustering results.

Further exploration involved the examination of similarity between members of various clusters. We performed a downsampling of the dataset comprising 1000 points to be able to compute and display the matrix in Figure 8b. This matrix allowed easy identification of clusters on the diagonal, but there are also noticeable similarities between specific pairs of clusters, in particular between clusters 1, 5, and 6, clusters 2 and 3, and clusters 3 and 4.

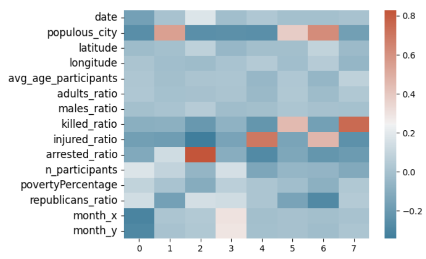
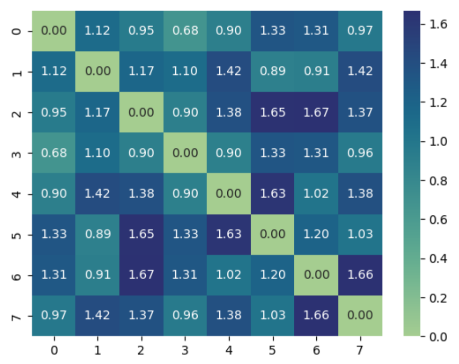


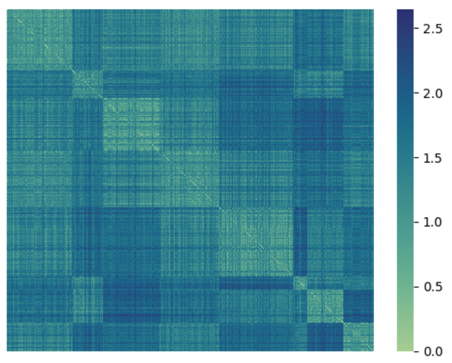
Figure 9: K-means: matrix of correlation coefficients between the value of an attribute and cluster membership.

An additional critical matrix for our analysis is the correlation matrix between the value of each attribute and cluster membership, as shown in Figure 9. For instance, in this matrix, we can see how clusters 1, 5, and 6 exhibited positive correlations with the attribute *populous_city*, explaining the observed similarities in the earlier analyzed similarity matrix.

From this correlation matrix, we can gain other semantic insights for each cluster. For example, let's take cluster 0, which is negatively correlated to *month_x* and *month_y* and *populous_city*. This indicates that this cluster contains incidents that happened in the third quarter of the year in cities with a population of less than 500k.



(a) Matrix of distances between centroids.



(b) Matrix of similarity for a downsampled dataset.

Figure 8: K-means: matrices capturing spatial separation and feature similarity between clusters.

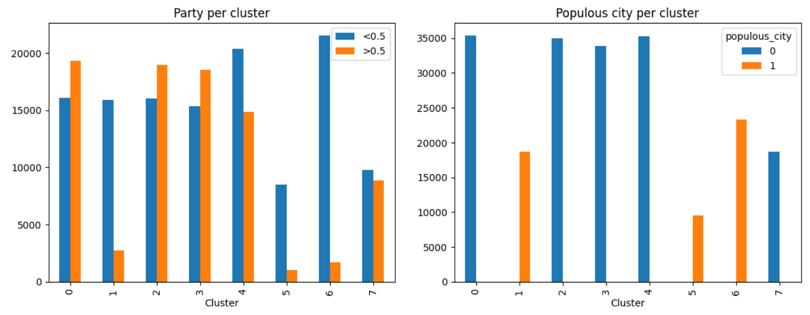
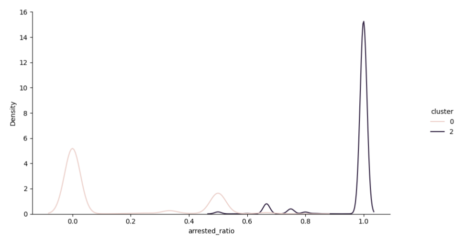


Figure 10: K-means: comparison of distributions among clusters of the ratio of people voting Republicans and cities having a population of more than 500k.

We proceed now to analyze the distributions of some selected attributes in the various clusters. As a first thing, we observe the political tendencies in each cluster, analyzing when the attribute *republican_ratio* is greater or smaller than 0.5, indicating that the district where the incident happened was mostly Republican or Democrat. We can see in Figure 10 that most clusters have mixed political orientations, except for clusters 1, 5, and 6. This can be explained by looking at the plot on the right, where those same clusters are associated with shootings that happened in populous cities, thus reinforcing a well-known fact that the Democrat party has stronger support in larger cities.

Finally, we inspect a continuous distribution, the ratio of arrested people. From Figure 11b, it is possible to observe a three-peak distribution over the whole dataset with peaks in 0, 0.5, and 1. When we plot the distribution of the same attribute in cluster 2 (which is positively correlated with the arrested ratio) and cluster 0 (which is negatively correlated), as in Figure 11a, we observe how the two distributions are very different. This means that each cluster captured a different mode of the original multimodal distribution, which is desirable in clustering as it means that the


(a) Distribution of arrested ratio on positively and negatively correlated clusters.

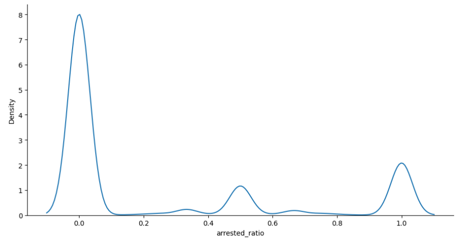

(b) Distribution of the arrested ratio attribute in the whole dataset.

Figure 11: K-means: comparison of the distribution of the arrested ratio attribute among different clusters and the whole dataset.

clustering performed a semantically meaningful separation.

2.3 DBSCAN

The subsequent clustering, with its relative analysis, is performed on the dataset restricted to the incidents that happened in the state of Florida.

2.3.1 Choice of *eps*

To obtain good clustering results with DBSCAN it is fundamental to choose a good value for *eps*. Here we use the knee method, plotting the ordered distance from the 5th neighbor, Figure 12, to choose a value where the obtained curve started to increase the steepness. However, when picking a too-low value for *eps*, we obtained only one cluster for noise and another cluster containing all other points. For this reason, we opted for an intermediate value of *eps* = 0.48. This yielded, unfortunately, a high number of points labeled as noise, but higher choices of *eps* didn't achieve any clusterization of non-noise points. With this value, instead, resulting clusters had a reasonable amount of points and good separation and cohesion.

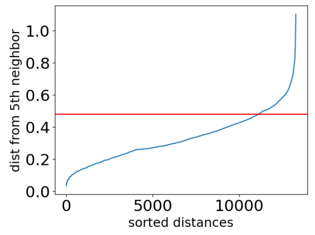
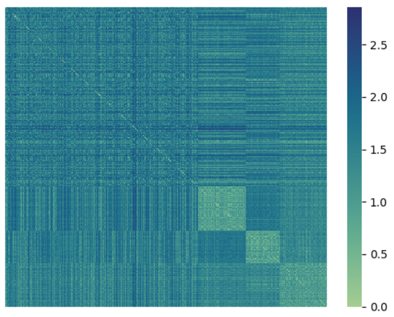


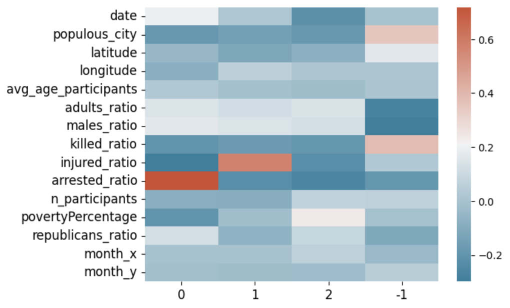
Figure 12: DBSCAN: Sorted distance from the 5th neighbor.

2.3.2 Cluster analysis

Since DBSCAN doesn't yield globular clusters, we skip the centroids analysis as these are not representative of the points in the clusters. Similarly to what we did for k-means, we analyzed the ordered similarity matrix (Figure 13a) and the correlation matrix (Figure 13b). From the former, we can see how more than half of the points have been classified by DBSCAN as noise, with the remaining points being partitioned into three well-defined clusters. From the latter, we can see how the three clusters differ mainly in the number of arrested, injured, or unharmed people.



(a) Matrix of similarity for a downsampled dataset.



(b) Matrix of correlation coefficients for the correlation between the value of an attribute and cluster membership.

Figure 13: DBSCAN: matrices capturing spatial separation and feature similarity between clusters.

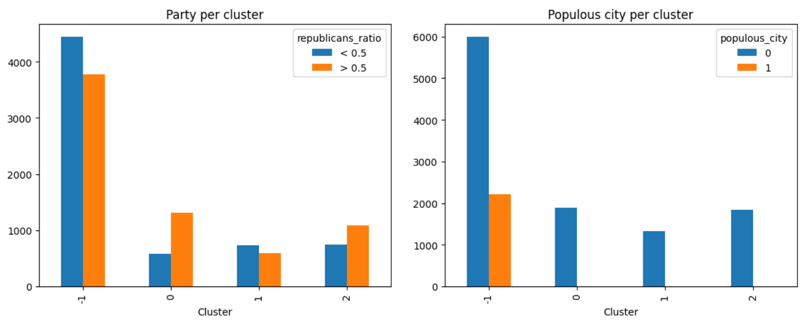
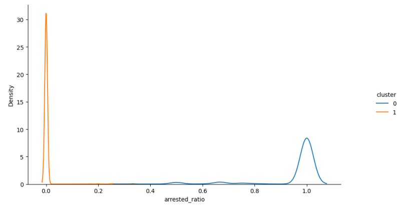
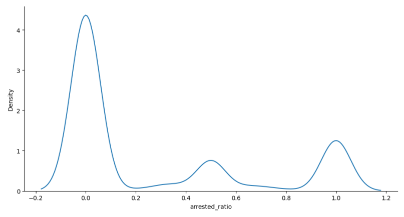


Figure 14: DBSCAN: comparison of distributions among clusters of the ratio of people voting Republicans and cities having a population of more than 500k.



(a) Distribution of arrested ratio on two clusters, one positively and one negatively correlated to the attribute.



(b) Distribution of the arrested ratio attribute in the whole dataset.

Figure 15: DBSCAN: comparison of the distribution of the arrested ratio attribute among different clusters and the whole dataset.

Unlike in k-means, the clustering didn't separate shootings that happened in cities with a population greater than 500k. Indeed, as can be observed in Figure 14, incidents in populous cities are all labeled as noise. As a consequence, the separation between shootings in cities with Democrat or Republican predominance is not picked up, as can be seen in the plot on the right. Fortunately, some attributes have been correctly separated, such as the *arrested_ratio* plotted in Figure 15, for which, as in k-means, the multimodal distribution for this attribute present in the dataset is well-separated in the clusters.

2.4 Hierarchical Clustering

Like DBSCAN, hierarchical clustering was performed only on incidents that happened in Florida. To perform hierarchical clustering, we decided to employ the agglomerative clustering algorithm.

2.4.1 Choice of linkage

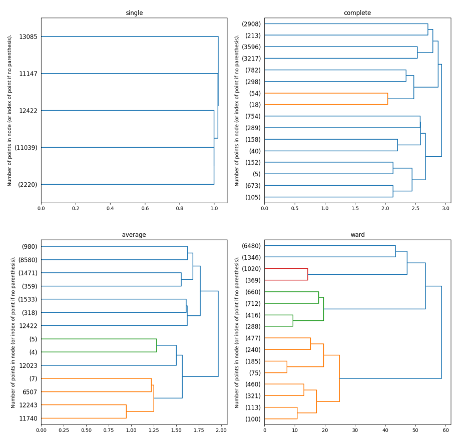
With this clustering technique, it is fundamental to choose the best linkage criterion (relative to our dataset) to achieve good results, so we decided to experiment with the four most commonly used linkages: single, complete, average, and Ward. In figure 16a is plotted the dendrogram for each linkage criterion. As can be observed, Ward is the only linkage technique that yields well-separated clusters, even if with a slightly unbalanced cluster. After the choice of the linkage strategy, we applied once again the Elbow Method, which helped us to identify $k = 4$ as a reasonable number of clusters.

2.4.2 Cluster analysis

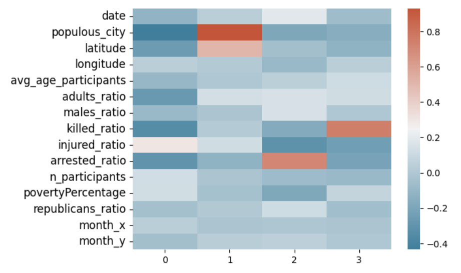
With the selected number of clusters, the hierarchical clustering produced three well-defined and separated clusters, as can be seen in Figures 17a and 17b, and one big cluster with more variability and less separated from the other clusters. This reminds us of the cluster of noise points created by DBSCAN.

Figure 16b reveals that the similarities of the obtained clusters with those obtained with DBSCAN are just apparent, with relevant differences in the attribute correlated to cluster membership.

Indeed, we can see that in the case of hierarchical clustering, incidents that happened in big cities have been grouped in one cluster, a fact that can also be seen in the right plot of Figure 18. In the left plot, instead, we can observe that incidents in predominantly Republican or Democrat cities have not been separated. In this case, big cities and Democrat prevalence are not correlated, as Florida was a



(a) Dendrogram for each linkage criterion.

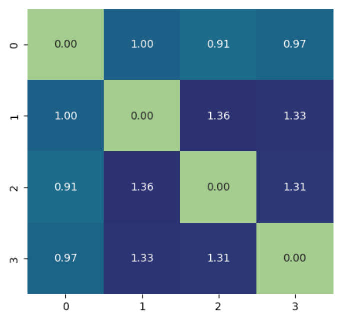


(b) Hierarchical clustering: matrix of correlation coefficients between the value of an attribute and cluster membership.

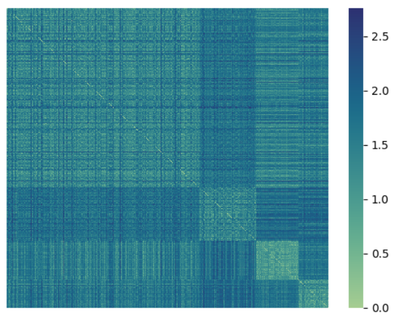
Figure 16

mostly Republican state in the years when the data were collected.

As with the precedent clustering approaches, the *arrested_ratio* (together with *killed_ratio*) is a crucial attribute for the cluster composition, and this yields once again a good separation of the distribution on the entire dataset into two opposed distributions.



(a) Matrix of distances between centroids.



(b) Matrix of similarity for a downsampled dataset.

Figure 17: Matrices capturing spatial separation and feature similarity between clusters.

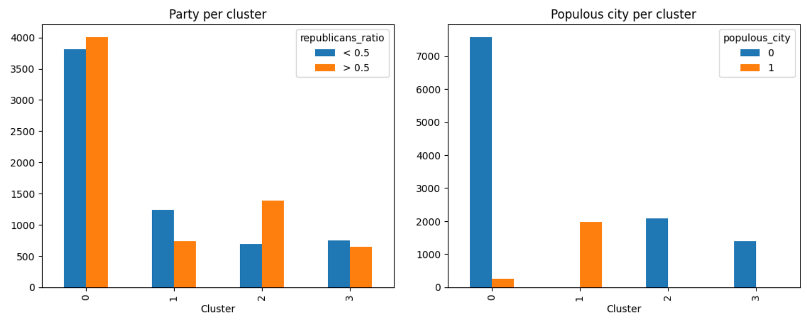
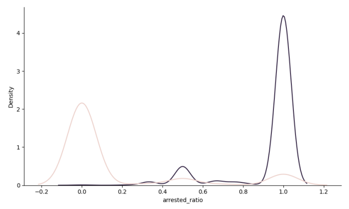


Figure 18: Hierarchical clustering: comparison of distributions among clusters of the ratio of people voting Republicans and cities having a population of more than 500k.

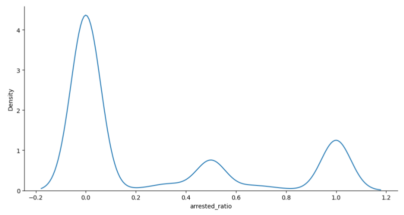
2.5 Alternative clustering approach: X-means

As an alternative clustering approach, we decided to explore X-means. This clustering algorithm has the advantage of not having to set other hyperparameters apart from a range for the number of clusters, which we set to [3, 10] based on the result of previous approaches. We applied X-means to the entire dataset, in order to better compare the results with those of K-means.

2.5.1 Cluster analysis

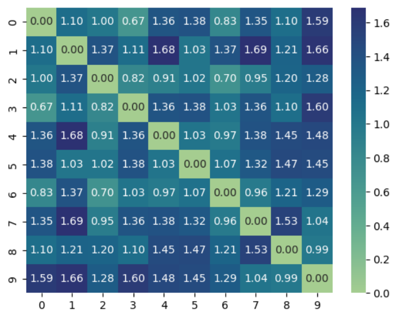


(a) Distribution of arrested ratio on positively and negatively correlated clusters.

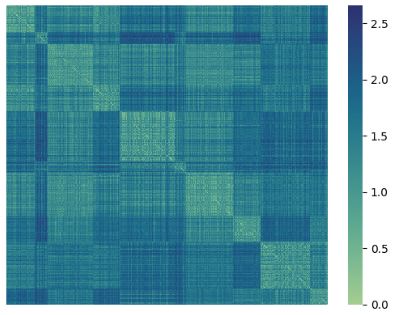


(b) Distribution of the arrested ratio attribute in the whole dataset.

Figure 19: Hierarchical clustering: comparison of the distribution of the arrested ratio attribute among different clusters and the whole dataset.



(a) Matrix of distances between centroids.



(b) Matrix of similarity for a downsampled dataset.

Figure 20: X-means: matrices capturing spatial separation and feature similarity between clusters.

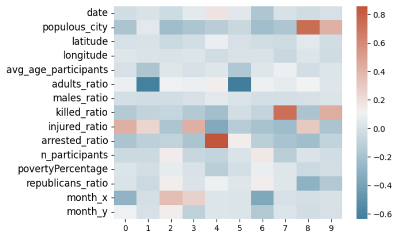


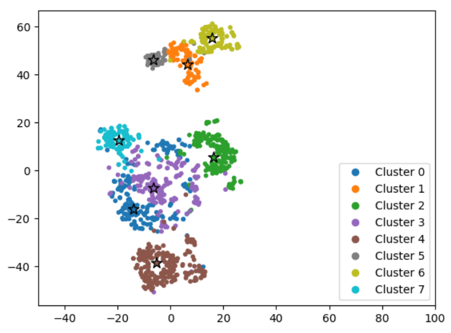
Figure 21: X-means: matrix of correlation coefficients between the value of an attribute and cluster membership.

X-means algorithm yielded 10 clusters (the maximum according to the imposed constraint), which are not equally distributed in the space, as can be seen in Figure 20a. Furthermore, the similarity matrix in Figure 20b highlights evident similarities between points belonging to different clusters. The high number of clusters also brought a great specialization of the clusters: from Figure 21 we can see that four clusters are correlated with the attribute *injured_ratio*, but each of them is characterized by the correlation with at least another attribute.

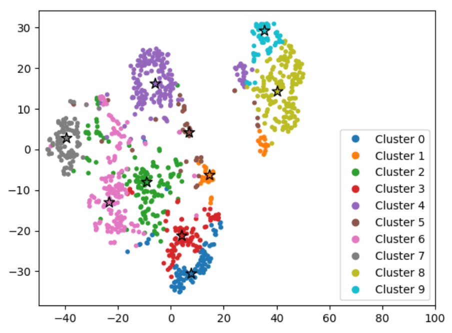
2.6 Discussion

The clustering analysis performed on this specific dataset highlighted relevant downsides for some algorithms. In particular, the DBSCAN algorithm classified most of the data points as noise; thus, we had to perform a careful selection of the *eps* parameter to obtain reasonable results. Regarding X-means, we found the algorithm to select always the maximum available number of clusters, even if the maximum specified was very high (up to 50 clusters), reducing the main advantage of the algorithm of not having to choose the number of clusters.

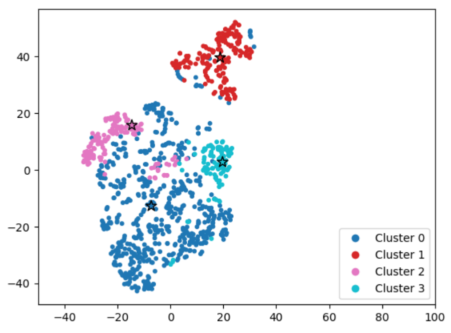
To visually compare the results of the four clustering approaches we used, we report in Figure 22 the t-SNE representation of the clustered dataset for each algorithm (with centroids reported when meaningful). Despite every approach having its own weaknesses, we can conclude, also by looking at the t-SNE plot, that the one that seems to perform better in our case is the k-means algorithm. In addition, we found the elbow method to be a simple and efficient technique to perform the choice of the algorithm parameters for k-means (i.e. the number of clusters) with respect to the techniques we had to employ in the other clustering approaches.



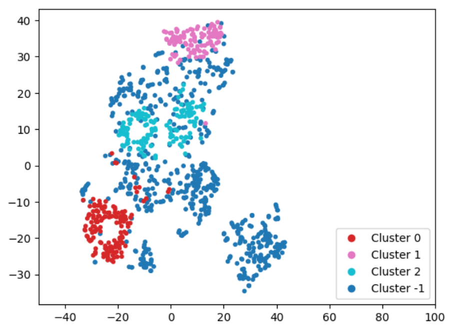
(a) Kmeans clustering



(b) Xmeans clustering



(c) Hierarchical clustering



(d) DBSCAN

Figure 22: TSNE Plots for K-Means, DBSCAN, Hierarchical, and X-Means Clustering.

3 Classification

In this section, we deal with a classification problem that consists in predicting whether an incident has caused casualties. The target is thus a binary variable stating if the number of killed participants is strictly greater than zero. To address this task, we explore several classification techniques and compare their performance in terms of accuracy and confusion matrix.

3.1 Dataset

As in the case of clustering, we decided to remove some features that we retained poorly informative or that excessively increased the dimensionality of the dataset. We removed the minimum and maximum age of the participants and the state and the city in which the incident happened. When working with classifiers relying on distances or scaled features, we performed standard scaling, ensuring all the features were normally distributed. To assess and perform a final comparison between all the tested models, we kept separated from the training data a test set, whose size was 20% of the original dataset.

Unbalanced Target Class Since the major part of the incidents present in the dataset ended up with no killed participant, the subdivision of the training data between the two classes is very unbalanced. To deal with this problem, we decided to perform undersampling, reducing the number of training data (which was higher enough to permit this) in such a way as to achieve an equal subdivision between the two classes. The undersampling has been performed only on the training set after it has been separated from the test set.

3.2 Hyperparameter selection

Almost all the models we employed for this classification task need careful hyperparameter tuning to achieve the best performances. To select the best values of the hyperparameter of each model, we performed a K-fold cross-validation (with k=5) exploring the hyperparameter space with a grid search when the number of configurations was reasonable and a random search when an exhaustive search would have required too much time.

3.3 Decision Tree

As a first classifier, we tested the Decision Tree, a simple but effective model that allowed us to obtain good results with a relatively short training time. In Table 1 we list the hyperparameters we tuned, and the ranges we searched on. Best values (highlighted in bold in Table 1) were selected according to the K-fold cross-validation. Finally, we evaluated the results on the test set of the best-performing model. As we can see in Table 7 the model has been able to learn, reaching an accuracy of 0.86 on the test set.

3.4 Random Forest

Encouraged by the results obtained with Decision Tree we tested the Random Forest, performing model selection on the same hyperparameters used with Decision Tree in addition to the maximum number of features and the use of bootstrapping in the training of the model (Table 2). This model,

Hyperparameter	**Values**
Criterion	**Gini**, Entropy
Maximum depth	3, **5**, 7, 10, None
Minimum samples split	random int ∈ [10, 50] (**32**)
Minimum samples leaf	random int ∈ [10, 50] (**50**)

Table 1: Hyperparameter space explored with random search for Decision Tree

as an ensemble of Decision Trees, improved or reached the performances on all the metrics measured as we can see in Table 7.

Hyperparameter	**Values**
Criterion	**Gini**, Entropy
Maximum depth	3, 5, **7**, 10, None
Minimum samples split	random int ∈ [20, 50] (**41**)
Minimum samples leaf	random int ∈ [20, 50] (**43**)
Maximum number of features	random int ∈ [1, 20] (**19**)
Bootstrap	True, **False**

Table 2: Hyperparameter space explored with random search for Random Forest

3.5 AdaBoost

We decided to test another ensemble of Decision Trees with AdaBoost, using as base classifier the Decision Tree initialized with the best hyperparameters found in Section 3.3 obtaining, as highlighted in Table 7, the best-performing classifier, among the ones we tested, on all the metrics measured. As this is the best classifier, we report also the metrics of this model independently for each class in Table 3.

Class	**Precision**	**Recall**	**F1 score**
No killed	0.97	0.85	0.91
Killed	0.67	0.91	0.77

Table 3: Metric measured independently for the two classes

3.6 RIPPER

We also experimented with a rule-based classifier, RIPPER. Training this classifier required too much time to perform training on the entire dataset; for this reason, the following results are obtained training the model on a smaller portion of the training set. Since the hyperparameter space (Table 4) was sufficiently small, we explored it with a grid search to find the best hyperparameter values. The results obtained with this classifier are a bit inferior to those obtained with Random Forest and AdaBoost, with a much higher training time.

Hyperparameter	**Values**
Prune size	**0.5**, 0.6, 0.7
k	**1**, 3, 5

Table 4: Hyperparameter space explored with grid search for RIPPER

3.7 K-nearest neighbors

As a distance-based classifier, we decided to test a simple K-NN, exploring values for K of 1, 3, 5, 10, and 20. We found the best value to be 1, but even with the best classifier emerging from the grid search, the results were worse on the test set with respect to the other tested models (Table 7), showing also a bit of overfitting, with a training accuracy significantly higher than the test accuracy.

3.8 Naive Bayes Classifier

We then tested Gaussian Naive Bayes, a simple probabilistic model that required no hyperparameter tuning (since we didn't want to impose any prior on the data distribution). With this classifier, as we can see in Table 7, we obtained slightly worse results compared to the best-performing classifier.

3.9 SVC

Another classifier we tested is SVC. As RIPPER, this classifier also required too much time to be trained on the entire training set, so again we reduced the dimension of the training set to perform the grid search for this model. The explored hyperparameters were the regularization C and the kernel type, whose tested values can be seen in Table 5. As reported in Table 7 the results are good but still inferior to those obtained with the ensemble methods.

Hyperparameter	**Values**
C	**0.1**, 1, 10
Kernel	**Radial basis function**, Linear, Polynomial

Table 5: Hyperparameter space explored with grid search for SVC

3.10 Multi Layer Perceptron

Finally, we decided to test a Multi Layer Perception. The architecture of the network was decided using hyperparameter tuning, which had also the purpose of selecting the best learning rate (the details can be observed in Table 6). To obtain a network capable of doing classification on a binary class, the output layer was fixed to have 1 unit and sigmoid activation. To prevent the network from overfitting, we employed early-stopping on a separate validation set with a patience of 10 epochs (on a total of 100 epochs of training). In addition to results in Table 7, where we can see the model performing similarly to the ensemble models, we report in Figure 23 the learning curves for both training and validation loss and accuracy.

Hyperparameter	**Values**
N. of layers	**2**, 3, 4, 5
N. of units per layer	5, 10, 15, ..., **45**, 50
Activation function	**sigmoid**, tanh, relu
Learning rate	0.5, **0.1**, 0.01, 0.001, 0.0001

Table 6: Hyperparameter space explored with random search for MLP

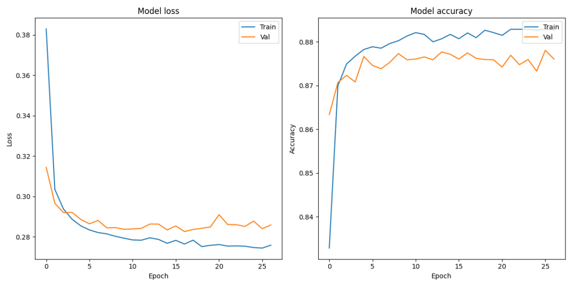


Figure 23: Learining of the best Multi Layer Perceptron

Model	**Training accuracy**	**Test accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	0.88	0.86	0.89	0.86	0.87
Random Forest	0.88	0.87	0.90	0.87	0.87
AdaBoost	**0.89**	**0.87**	**0.90**	**0.87**	**0.87**
Ripper	0.86	0.86	0.88	0.86	0.86
K-NN	0.87	0.83	0.87	0.83	0.83
Naive Bayes	0.85	0.86	0.88	0.86	0.86
SVC	0.88	0.86	0.89	0.86	0.87
Neural Network	0.88	0.87	0.89	0.87	0.87

Table 7: Results all classifier

3.11 Conclusion

In conclusion, from table 7, we can see how ensemble classifiers, based on Decision Tree, slightly outperform the other tested models, with AdaBoost being the best-performing one. Also, from the point of view of computational cost, the Decision Tree emerged as a simple but effective model, with ensemble classifiers based on it, being able to improve the performance with only a constant multiplicative increase in time complexity (which is in practice reduced thanks to the easy parallelization available for ensemble models) on both training and inference. Finally, it is worth noting, in Table 3, that the precision for class ”Killed” is significantly inferior to the precision of the other class (and this result emerged in all the tested models), while the recall is higher, leading to the conclusion that even if undersampling helps the models avoiding false negatives, it is not sufficient to obtain good precision for the ”Killed” class.

4 Time Series Analysis

In this section, we only consider the incidents that happened from 2014 to 2017. For each sub-task, we will define a score function. According to this score function, a time series will be created for each city in the dataset, by calculating the score value of each week from 2014 to 2017.

4.1 Time series construction

First of all, we noticed that in the dataset the same city appears with multiple names, thus some preprocessing was needed in order to make the process straightforward. We removed all information in brackets, all spaces and turned all to upper case. E.g. Incidents that happened in "New York (Manhattan)" and "New York (New York City)" are grouped in "NEWYORK". After this operation, the number of different cities dropped from 12596 to 10331. Before building the time series, we defined a threshold of criminality to filter the cities, i.e. excluding the ones with a low number of weeks with incidents. In particular, we considered only cities where at least 15% of the weeks were characterized by one or more incidents. 713 cities survived this filtering operation; thus, for each of them, we generated a time series. The length of each time series is 208, i.e. the number of weeks in the years [2014, 2017].

Score function We defined the score function as the *perceived risk*, and we decided to compute it in an "eligibility trace fashion", borrowing this concept from reinforcement learning rewards. The score of a city c in a week w is then given by the following equation: $score_{c,w} = \alpha * score_{c,w-1} + n\_incidents_{c,w}$. As we can see, the perceived risk is a combination of the number of incidents in the current week with the score of the previous week decayed by a factor α. This score function will be used for both clustering and shapelet extraction.

4.2 Time series clustering

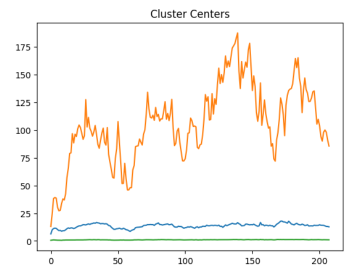
The goal of this task is to group similar cities through the use of the created time series based on the defined score. After analyzing clustering results, we will extract motifs and anomalies in the time series for a deep understanding and exploration.

4.2.1 Scaling time series

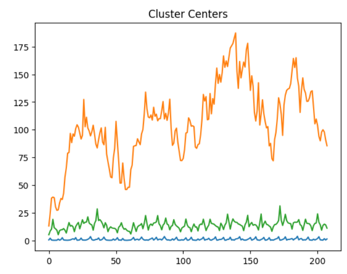
Before applying any clustering algorithm, we considered the problem of scaling the dataset in the range [0,1]. However, it is reasonable to assume (and easy to visualize) that most of the information contained in the time series is in their mean value: in fact, we expect cities with a high criminality rate to show high score values throughout the whole temporal sequence, and vice-versa. This suggests that min-max scaling applied independently to each time series would cause a high loss of information in our setting. For this reason, we computed a fixed min-max scaling for all time series.

4.2.2 Shape-based clustering

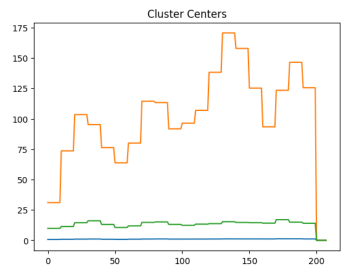
In this section, we explore a type of time series clustering algorithm that consists of applying traditional clustering procedures by replacing the distance metrics with a suitable distance for time series. We will denote the K-means variant for time series as *TS K-means*.



(a) TS K-means (Euclidean)



(b) TS K-means (DTW)



(c) PAA + K-means(Euclidean)

TS K-means with Euclidean distance. We used the Elbow method to determine the optimal number of clusters (in a range k$\in [2, 20]$), and we found the best result for k=3. The resulting centroids are plotted in Figure 24a.

TS K-means with DTW. Using DTW as distance metrics guarantees scale and shift invariance, despite being much more computationally expensive. In fact, the time required by the algorithm was two orders of magnitude higher than the previous approach. Before applying the Elbow method to find the optimal k, we tried to reduce the computational cost. Thus, we used a constrained version of DTW for clustering and Euclidean distance for the Silhouette score, and we identified the optimal number of clusters k=3. The resulting centroids are shown in Figure 24b.

Approximation-based clustering. We experimented with the use of Piecewise Aggregation Approximation. In Figure 24c, we can observe that even with a restricted representation of the time series, the results were pretty much equivalent, thus enforcing the idea that PAA is a good solution for the efficiency/computational complexity trade-off.

4.2.3 Clustering transforming time series to points

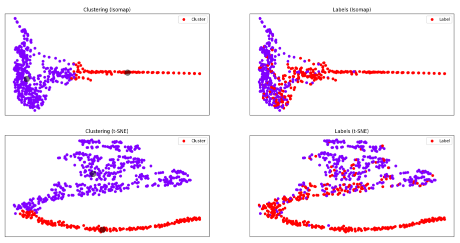


Figure 25: Compression-based clustering results.

In this section, we use two different approaches for time series clustering, which have in common the fact that they transform time series into data points and then apply point-wise clustering algorithms (such as the vanilla K-means or DBSCAN).

Feature-based transformation. This approach consists in extracting features (e.g average, standard deviation, median) and then representing each time series as a point in this feature space.

Compression-based transformation. This approach consists of converting each time series with a string, encoding it in `utf-8`, and subsequently compressing it with the DEFLATE algorithm by `zlib`. Subsequently, we define the distance function between points as

$$cdm(x, y) = \frac{len(compress(x + y))}{len(compress(x)) + len(compress(y))}$$

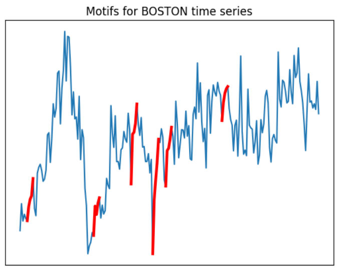
and we used the matrix of pairwise distances to perform clustering.

4.3 Motifs extraction and anomalies detection

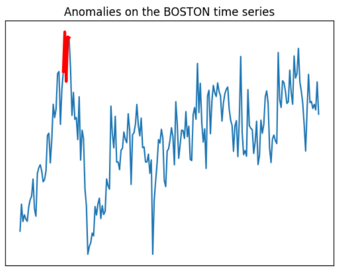
In this section, we compute the matrix profile of the BOSTON time series, using a sliding window of 5 weeks, and we analyze it to extract motifs and anomalies in the time series.

Motifs extraction The algorithm for motifs extraction takes a maximum number of motifs as a hyperparameter and identifies motifs by looking at the smallest values in the matrix profile. An example of motifs extraction can be visualized in Figure 26a.

Anomalies detection. To detect anomalies, we computed the z-scores for all the entries in the matrix, and we applied a threshold mask of 3 standard deviations. In this way, sub-sequences differing from all other sub-sequences of a particularly high distance are considered anomalies. Detected anomalies can be visualized in Figure 26b.



(a) Motifs extraction.



(b) Anomalies detection.

4.4 Shapelets extraction

Shapelets are discriminative sub-sequences of time series that best predict the target variable in a classification problem. In this section, we will attempt to learn the most discriminative shapelets for predicting the target variable, i.e. `IsKilled` (number of killed in a specific city over a certain threshold). We defined this threshold at 25 kills per city in the time span of the analysis to obtain a reasonably balanced dataset. The dataset is the same as the one used in the clustering task, with the addition of labels, as we are now in the supervised learning setting.

Classification methods. We initially split the dataset into training and test set, using the training set for the shapelets learning. Then we used the learned shapelet transform on both the training and test set to turn the time series into points with shapelet-defined features. We then used this dataset to train classifiers, using a k-fold cross-validation approach for hyperparameter search. Ultimately we performed model assessment on the test set.

4.4.1 Scaling time series

In the previous tasks, it made more sense to take into account the amplitude of the time series, as explained in Section 4.2.1. In shapelet extraction, however, it is more instructive to perform independent scaling on each time series (amplitude scaling) for actually just focusing on the shape. Validating this opinion, we also analyzed the behavior of the shapelet algorithm with the fixed scaling and observed an, in fact, significantly better performance for the classification of the IsKilled variable. This makes sense as the relative amplitude of the time series is very important for separating them in the IsKilled(0,1) classes because the amplitude is tied to the number of incidents and, thus, likely also to the IsKilled target. Arguably, this could lead to the simple conclusion that shapelet extraction is not the correct way to go for this classification task, as it focuses more on shape than it does on amplitude, whilst the task at hand values amplitude over shape.

4.4.2 Shapelets discovery

We use the Learning Time-Series Shapelets (LTS) method to learn shapelets. Initially, the number of learned shapelets is defined algorithmically once given the lengths as hyperparameters. Next, the actual learning of the shapelets takes place. The method relies on a classification model that is differentiable with respect to shapelets: this allows shapelets to be updated in a stochastic gradient descent optimization fashion by taking steps towards the minimum of the classification loss function.

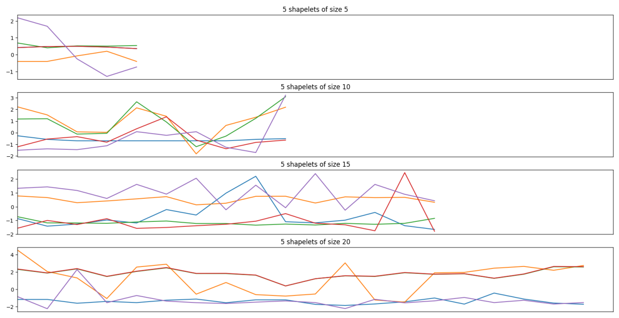
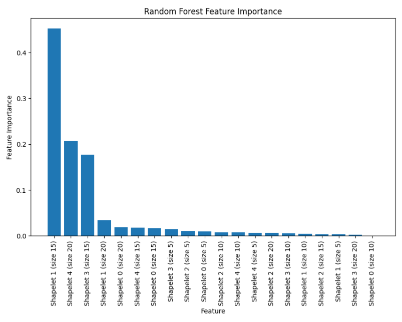


Figure 27: Shapelets learned with the Learning Time-Series Shapelets method.

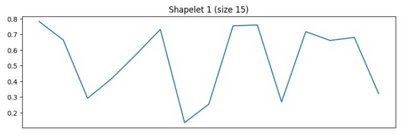
4.4.3 Classification with shapelets

After generating shapelets on the training set, the Shapelet Transform is applied to both the training and test sets. The resulting shapelet-transformed sets can be finally used to train the classifier. We train a K-NN classifier and a Random Forest Classifier, both with K-fold cross-validation. The models reach accuracies of over 80%, meaning that the shapelet transform is encoding features valuable for distinguishing in the `IsKilled` class.

4.4.4 `IsKilled` shapelet



(a) Importance assigned to shapelet-based features by the random forest.



(b) Most discriminative shapelet for the `IsKilled` class.

Finally, we want to extract one single shapelet that contains the most information about our target variable. Our approach tries to exploit the explainability of random forests; in other words, we use a random forest to solve the classification task while exploring the features on which the model relies the most for the final decision. This way, we identify the best shapelet as the one having generated the most discriminative feature according to the random forest. This shapelet is reported in Figure 28b, and we can observe how the association between this subsequence and `IsKilled = True` might be that cities with high oscillation of perceived risk are subject to many incidents and thus many people are killed.