Data Mining Project Report

A.A. 2023-2024 Group 15

Emiliano Quaranta - 580667
Francesco Di Lena - 580821
Alessandro Querci - 578615

8th January 2024

Contents

1 Data Understanding and Preparation

The project focused on multiple datasets containing informations about gun accidents in the United States, poverty in the different States and the political results during each midterm election. The data has been cleaned in order to remove duplicates and invalid data and to allow for better results in the later tasks.

1.1 Dataset Features

The first Dataset contains a list of gun accidents in the United States. Before cleaning, these are all the features present in the Dataset.

Features in Incidents Dataset

Feature	**Type**	**Description**
date	object	date of incident occurrence
state	object	state where incident took place
city_or_county	object	city or county where the incident took place
address	object	address where incident took place
latitude	float64	latitude of the incident
longitude	float64	longitude of the incident
congressional_district	float64	congressional district where the incident took place
state_house_district	float64	state house district
state_senate_district	float64	state senate district where the incident took place
participant_age1	float64	exact age of one (randomly chosen) participant in the incident
participant_age_group1	object	exact age group of one (randomly chosen) participant in the incident
participant_gender1	object	exact gender of one (randomly chosen) participant in the incident
min_age_participants	object	minimum age of the participants in the incident
avg_age_participants	object	average age of the participants in the incident
max_age_participants	object	maximum age of the participants in the incident
n_participants_child	object	number of child participants 0-11

n_participants_teen	object	number of teen participants 12-17
n_participants_adult	object	number of adult participants (18+)
n_males	float64	number of males participants
n_females	float64	number of females participants
n_killed	int64	number of people killed
n_injured	int64	number of people injured
n_arrested	float64	number of arrested participants
n_unharmed	float64	number of unharmed participants
n_participants	float64	number of participants in the incident
notes	object	additional notes about the incident
incident_characteristics1	object	incident characteristics
incident_characteristics2	object	incident characteristics

The second Dataset includes information about the poverty rate in the various States. These are all the features present in the Dataset.

Features in Poverty Dataset		
Feature	**Type**	**Description**
state	object	
year	int64	
povertyPercentage	float64	poverty percentage for the corresponding state and year

The third Dataset includes information about the Midterm election results in each state district. These are all the features present in the Dataset.

Features in Elections Results Dataset		
Feature	**Type**	**Description**
year	int64	
state	object	
congressional_district	int64	
party	object	winning party fort the corresponding congressional_district in the state, in the corresponding year
candidatevotes	int64	number of votes obtained by the winning party in the corresponding election

totalvotes	int64	number total votes for the corresponding election

We've also sourced a fourth Dataset about the total Population in each state for each year from the StatsAmerica website.

1.2 Data Casting

In the first dataset, some of the features had the *object* type casted to them, such as *participant_age1*, *min_age_participants*, *avg_age_participants*, *n_participants_child*, *n_participants_teen* and *n_participants_adult*. They've been converted to a numeric type.
Likewise, some of the features with a *Float64* type casted to them have been converted to *Int64*.
We've also converted the *date* column to *datetime*, and split it into three separate values *year*, *month* and *day*, to allow for easier queries in the later tasks.
The types of the second and third dataset have been left untouched.

1.3 Data Semantics

Certain attributes in the first dataset had the wrong value assigned to them in some instances. For example, in attributes correlated to age, there were various values set to negative integers or strings. These have all been set to *null*.
It's worth noting how there were 4 incidents where the *longitude* values were very different from all of the other cases. Plotting all of the incidents on a map using their location data, these 4 incidents fell inside of the territory of the Republic of China.

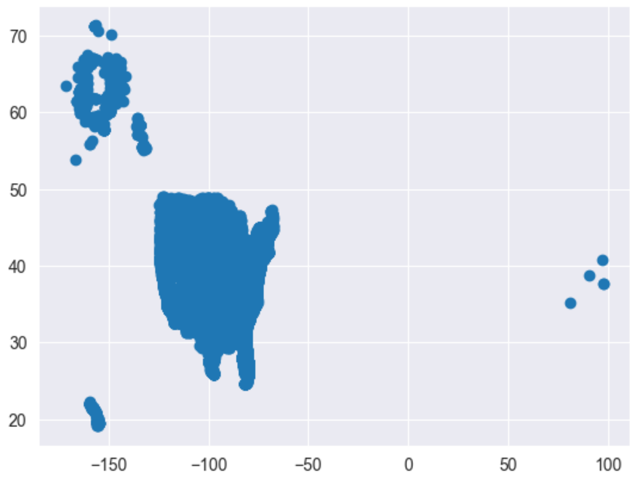


Figure 1: Location data for all incidents

Looking deeper into it, it was just an error where the *longitude* values were mistakenly inserted with a positive value instead of negative.
Certain incidents had a *year* value far into the future. We've dropped these samples from the dataset as we're uncertain of their credibility. We've also dropped samples from the year 2013 due to their low overall number which would skew the results of the analysis. Thus, the only

incidents considered are those from January 2014 to March 2018.
In the third dataset there was a single district win for the Democratic candidate Foglietta, having its own string *Foglietta (Democrat)* instead of just *Democrat* like all other discrict winners. This error has not been corrected since it's for an election year not covered by our analysis.
There are no semantic errors in the second dataset.

1.4 Removing Duplicates

In the first dataset we've removed 1146 duplicates based on a subset of features we considered critical for the description of an event (*longitude*, *latitude*, *date*, *state*, *n_killed*, *n_injured*, *city_or_county*, *n_participants*).

1.5 Outliers

While analyzing the first dataset there were 6 incidents that heavily skewed the metrics of the dataset due to their high number of participants and/or killed people. These are:

- Orlando Nightclub Shooting - 12th June 2016 - 50 killed
- Boston gang incident - 6th June 2016 - 60 participants
- Sutherland Springs church shooting - 5th November 2017 - 27 killed
- San Bernardino mass shooting - 2nd December 2015 - 16 killed
- Florida High school shooting - 14th February 2018 - 17 killed
- Colorado gang incident - 4th May 2017 - 52 participants

We've decided to drop these outliers in order to reduce the noise in the data.
We've found no outliers in the second and third dataset.

1.6 Dropping Features

In the first dataset we've decided to drop some of the features, in particular *participant_age1*, *participant_age_group1*, *participant_gender1*, *state_house_district*, *state_senate_district* and *address*, because they did not bring much insight into the analysis.
We haven't dropped any feature in the second and third dataset.

1.7 Data Transformation

We've combined the features *incident_characteristics1* and *incident_characteristics2* into a single column called *incident_category.* We've also mapped all values to a list of predefinied categories, which will be useful in the later tasks for the visualization and explanation of different clusters.

1.8 Missing Values

In the first dataset several features have missing values. Depending on the feature, we've taken different actions.

- longitude and latitude: we've decided to drop all incidents with missing values on the location data, because we consider those value integral for describing the single incident.

- n_arrested : we've replaced every missing value with 0
- n_unharmed: we've replaced every missing value by subtracting the number of killed and injured from the number of participants.
- n_participants: while there are no missing values for this feature, we've decided to drop all incidents with a number of participants equal to 0 because we felt they did not bring much insight into the analysis.
- min/avg/max_age_participants: we've considered this set of features very important for our analysis but they all had more than 40k missing values. Instead of dropping them all, drastically reducing our incident count, we've decided to replace those values with the median value for that feature, calculated on the values of all incidents in the same state. Compared to using the mean value, this method allows for a much more natural distribution of all the possible values, and does not skew the metrics as much.

In the second dataset we have no data for the year 2012 but that does not concern us since that year is not included in our analysis timespan.
There are no missing values in the third dataset.

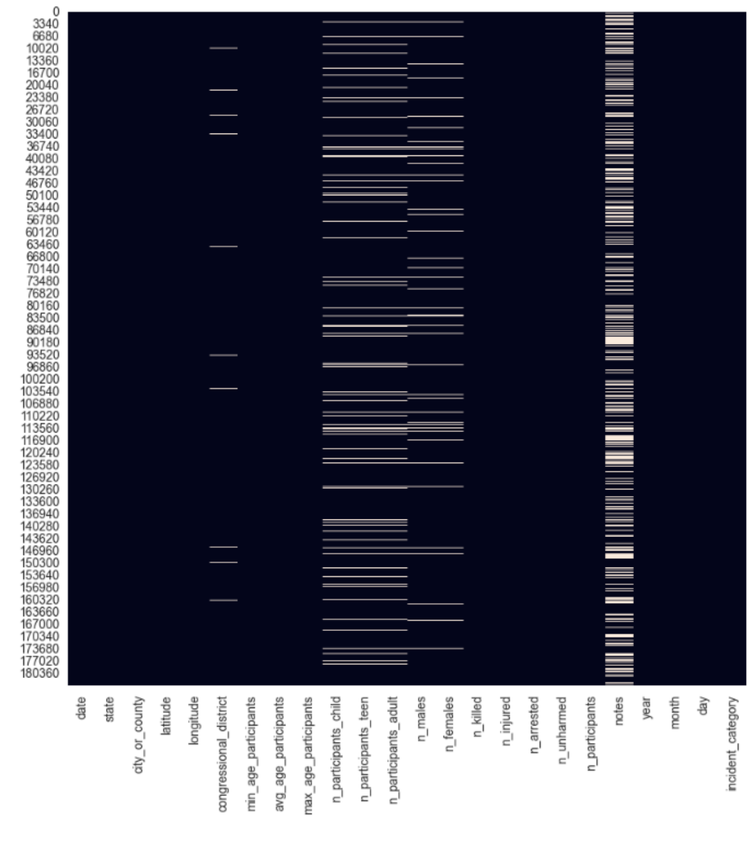


Figure 2: Missing values for all features in the Incidents dataset

1.9 Feature Extraction

Using the features already included in the analyzed datasets, we've calculated a collection of indicators to better describe each incident individually and also compared to other incidents in the datasets.

Features in Incidents Dataset

Feature	**Description**
kills_to_pov	the ratio of people killed to the poverty percentage of the state in that year
age_to_average	the ratio of average age of participants in each incident to the average age of participants in all incidents for that specific state
par_to_average	the ratio of number of participants in each incident to the average number of participants in all incidents for that specific state
n_involved	the number of participants involved in the incident
minors_to_pop	the percentage of minors involved in the incident compared to total minors population in the state
par_to_pop	the ratio between the number of participants and the population
kil_to_p	the ratio of killed to total participants
man_to_p	the ratio of male participants to total participants
fem_to_p	the ratio of female participants to total participants
arr_to_p	the ratio of arrested participants to total participants
unh_to_p	the ratio of unharmed participants to total participants
inj_to_p	the ratio of injured participants to total participants
n_minors	the number of underaged participants in the incident
teen_to_m	the ratio between the number of teen participants and the male participants
teen_to_f	the ratio between the number of teen participants and the female participants
par_to_tot	the ratio of participants in each incident to the total number of participants in all incidents for that specific month and year

Initially we also wanted to create features based on the third dataset but, upon further analysis, we were unable to extract meaningful data from it, given the difficulty of correlating the congressional districts to each incident, districts that could change over time, and also the high variability of the amount of data in each year, which made the analysis impossible to standardize across all states and all incidents.

1.10 Feature Correlation

Calculating the correlation matrix for the features in the first dataset, including the indicators specified in the previous section, shows how most of the features aren't highly correlated to each other. Notable exceptions are most parameters related to *n_participants*, which shows the importance this metric has on the description of an incident. Also worth noting the strong negative correlation between *man_to_p* and *n_females*, being much higher than the correlation between the opposite of those features. This indicates how it is much more common to have an incident with all male participants than one with all female participants, with the involvement of a male individual being much more likely.

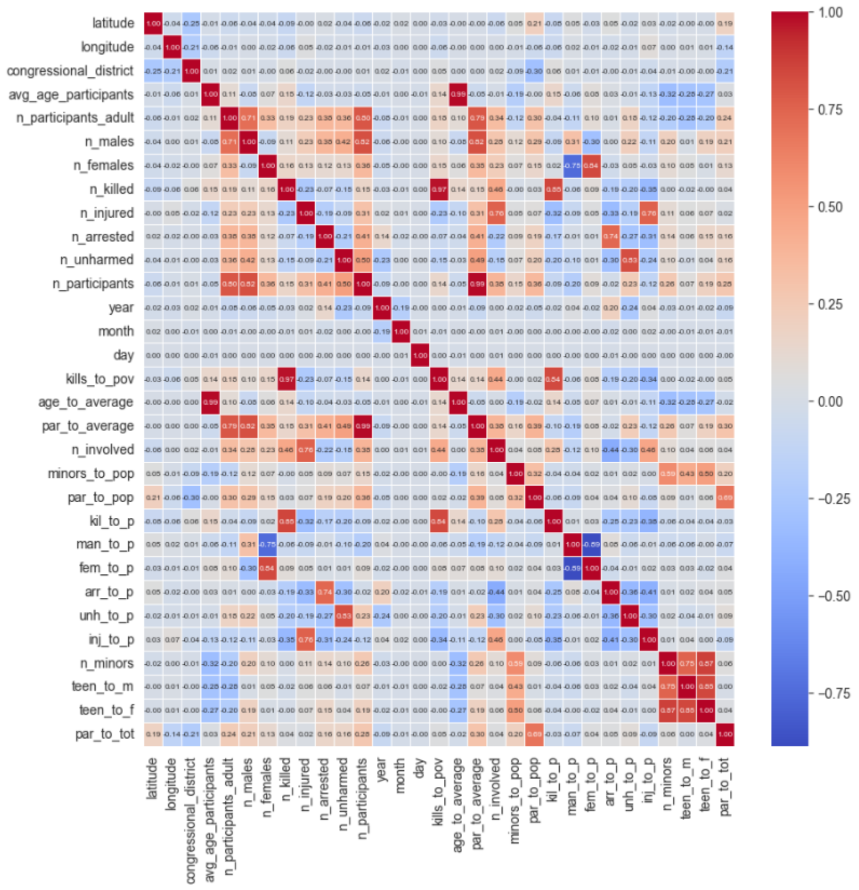


Figure 3: Correlation Matrix for Incidents dataset

2 Clustering Analysis

2.1 KMeans

2.1.1 Choice of features

For the Kmeans clustering analysis, we have chosen the features *n_participants*, *n_involved*, *age_to_average* and *par_to_tot*, as we believe they are the most distinct and the most representative to describe a single incident both in and of itself and against other incidents. Further testing with different features proved this belief.

2.1.2 Normalization

The data has been preprocessed using a MinMax Scaler, which is known to be effective in this type of analysis.

2.1.3 Finding best K

By running the KMeans algorithm 15 times, we found k = 7 as an ideal choice, and a good balance between an high silhouette value and low Davies-Bouldin scores. The result produced 7 clusters of different sizes, with no anomalous clusters of just a few samples.

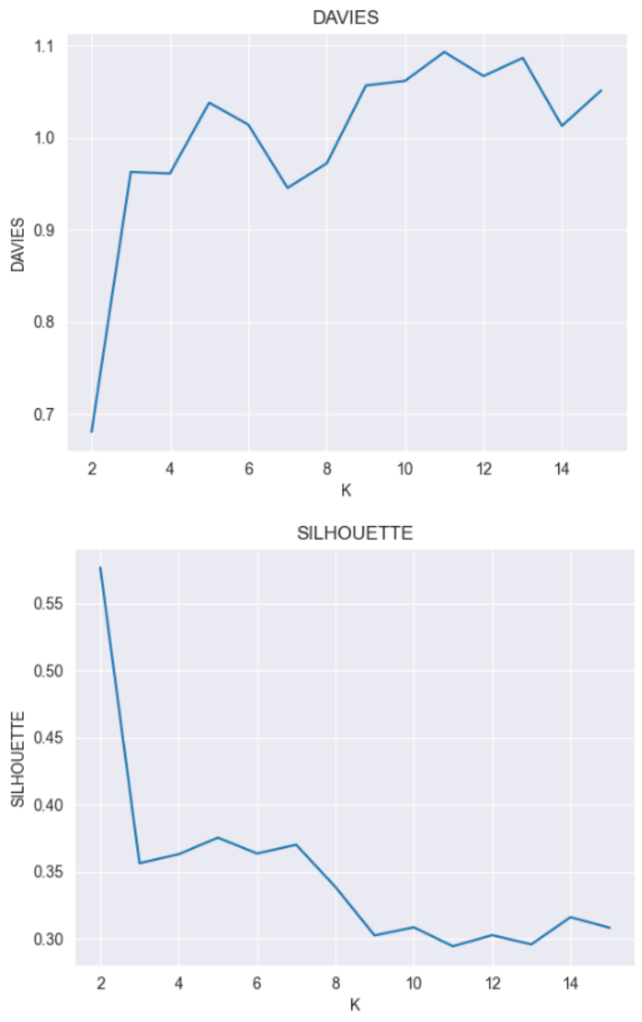


Figure 4: Silhouette and Davies-Bouldin scores for K-Means

2.1.4 Clusters Analysis

The K-Means algorithm divided the dataset into 7 clusters based on the values of the features we chose before. By looking at the samples in each of the clusters, these can be divided into 2 categories.

- 5 of the clusters represent the majority of the incidents, which consist of a low number of participants. The samples are thus divided based on the average age of its participants.
  For example, cluster 3 has a much lower average age than all of the other clusters and this reflects both in number of minors involved in each incident and in the much higher percentage of incidents related to school shootings.

- the other 2 clusters have, on average, an high number of participants.
  In cluster 2, the number of people killed or injured in each incident is, on average, more than double than in all other clusters.
  In cluster 5, the participants of the incidents make up a large percentage of all the people involved in a shooting incident in that same year. With all other values being in line with the the other clusters, these samples are usually drug related, a type of crime where the number of participants is usually very high but with relatively low bloodshed.

Due to the similarity of clusters 1 and 6, having similar values on the average participant age in each sample, they have been merged, leaving us with 6 clusters defined as mentioned above.

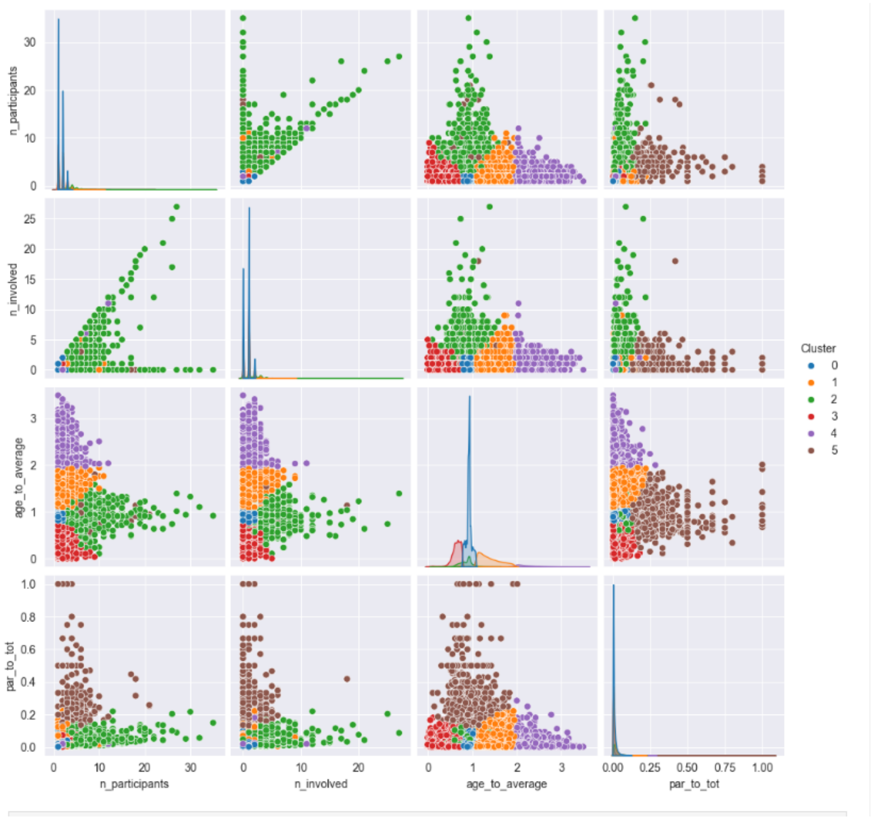


Figure 5: Scatterplot for K-Means after merging the clusters

2.2 DBScan

2.2.1 Choice of features

For DBScan we chose the same subset of features we used during the Kmeans Analysis to keep the clustering consistent. We selected California as the sample state, as it is one of the biggest states in the USA in terms of population.

2.2.2 Normalization

We did a little outlier search and removal, since we were working on a subset of the original data, dropping just 4 incidents that had a way higher number of involved participants.
We have tried both ZScaler and MinMax scaling to see which one was the best.

2.2.3 Parameter Search

The DBScan is based around two parameters, the minimum points needed to create a cluster and the maximum distance between two points to create a neighborhood (epsilon).
To find a range of epsilon to test in grid search, we plotted the graphs of distances to the k-nearest-neighbor for 3-5-9-12 as K and then looked for an elbow point.
For the standard scaled data we took a range of [0.05, 1],while for the minmax scaled data of [0.01 : 0.1].
The difference in ranges is due to the zscaled data having a max distance of 20, while the minmax's reached at maximum 1.
We set a range betwen 3 and 25 as the range of the min samples to create a cluster.
The grid search algorithm computed the silhoutte score of the clusters made with the parameter combinations, excluding results with only 2 clusters (meaning main cluster + noise) as they were not of interest.
The results of these searches with the minmax scaled data only got high (0.75) silhouette scores with diminishing number of clusters, meaning that the data was all collapsing either in noise or in a main cluster.
As seen before with the knn distances, it's likely that, in relation to the chosen features, there was a single high density center and only a small portion of data was scattered far from it, resulting in it being classified as noise.
On the other hand, with the standard scaled data we got more heterogeneous results. The silhouette score was overall way lower on the grid search (0.26 at best) with a very high number of clusters (over 15). We tried to pick the parameters pair with a decent score and with a manageable number of clusters to analyze.
With eps=1.0 and min_samples=12 we got 6 clusters that looked promising when plotted, perhaps biased by the number of involved participants, since it's the only feature with a more balanced distribution of values.

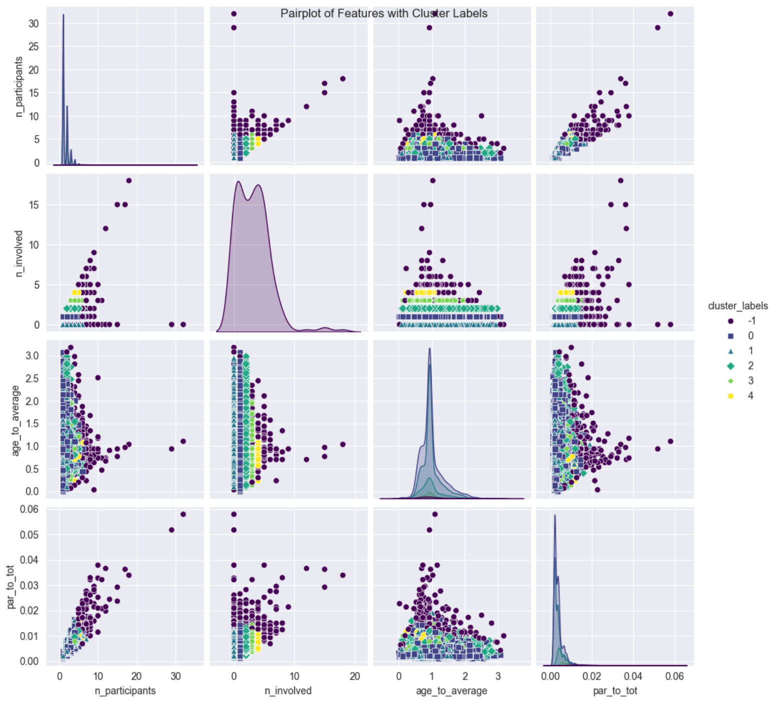


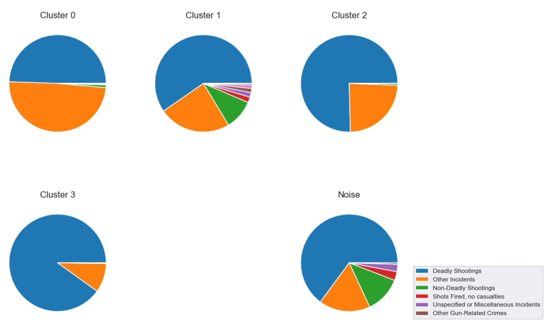
Figure 6: Scatterplot for DBScan

2.2.4 Cluster Analysis

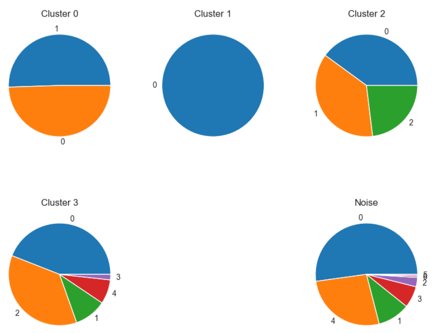
We decided to merge cluster 3 and 4 into one as they are the smallest clusters and fairly close in the plots. Then we merged the cluster labels to the main dataset for analysis.
By checking the incident categories and number of people killed in each cluster, we observed that:

- Cluster 0 had an almost even split between deadly incidents with just one victim and non-deadly incidents.
- Cluster 1 had no incidents with deaths but a wide array of categories, including deadly incidents. Interestingly, it might be clustering all the incidents marked as "shots fired, killed" without a specified number of people killed.
- Clusters 2 and 3 mostly consisted of incidents with more than one victim, constituting more than half of their incidents.
- The noise cluster contains all the outliers for each category, especially for the number of deaths.

Furthermore, by checking the stats of the clusters, we can notice how the ratio of arrested and male participants is higher in the first two clusters, while the other 3 have higher number of female and minors.
Regarding the time distribution, all the clusters seem pretty balanced.



(a) DBSCAN Incident Categories



(b) DBSCAN Number of n_killed

Figure 7: Results of DBSCAN Clustering

2.3 Hierarchical

2.3.1 Choice of features

For the hierarchical clustering we used the same subset of features used in the previous clusterings. We chose Texas as the state to carry out this task as it is a very populated area and gun ownership is very high.

2.3.2 Normalization

The data has been preprocessed using a MinMax Scaler, common for this type of analysis.

2.3.3 Complete linkage clustering

In the complete method we noticed that the clusters are completely unbalanced (cluster0: 9757 elements, cluster1: 2 elements) and with a high silhouette score (=0.8937616718981922).

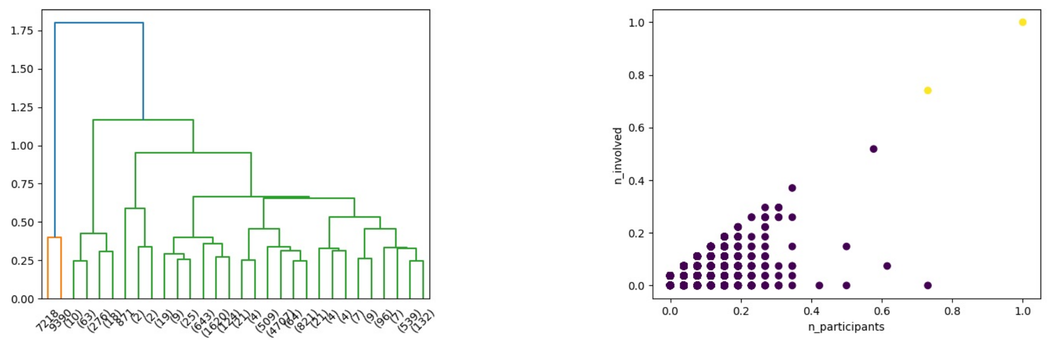


Figure 8: Complete linkage clustering results

The cluster with two elements contains data with an high number of participants and an high number of people involved so we hypothesized that they could be outliers. We redid the dendogram without them.

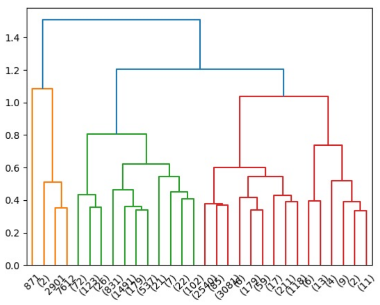
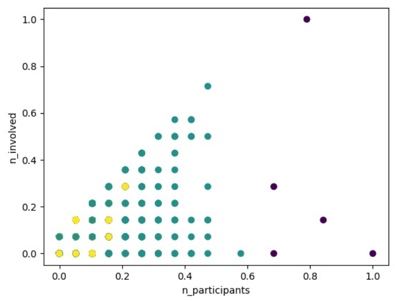



Figure 9: Results after removing the outliers

We noticed that this time the clusters are still unbalanced although less than before (cluster1: 6341, cluster2: 3411, cluster0: 5), at the cost of a big drop in the silhouette score (=0.2864240442313897). However we noticed a cluster with few elements (5) which, same as before, had data with a high number of participants and involved.
We made a third dendrogram without these 5 values and what we obtained was very similar to the first one, i.e., completely unbalanced(cluster0: 9720,cluster1: 30, cluster2: 2) and with a high silhouette score(=0.7120784180798325).
Therefore, it can be deduced that the following method for the selected features is highly influenced by noise and outliers.

2.3.4 Single method

Using single method we immediately noticed that the result obtained from the dendrogram this time is identical to the first one (same clusters and same silhouette score), that is, completely unbalanced and with a high silhouette score value.
We tried to do the same thing as in the previous method but this time the result didn't change, i.e. completely unbalanced clusters with a high silhouette score. Furthermore, using this method we can see more clearly how the data taken into consideration for the hierarchical clustering are is similar to each other and that the distance between most of them is almost zero.

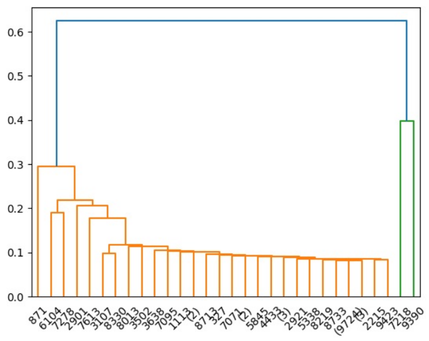
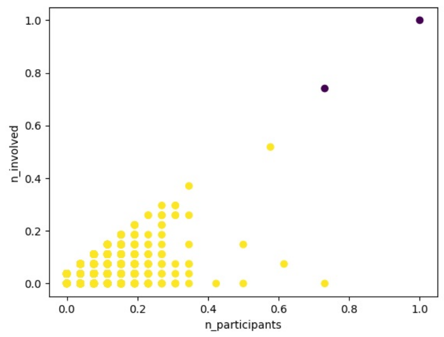



Figure 10: Results using the single method

2.3.5 Average method

Using the average method we obtain the same results obtained in the methods illustrated previously and here too we tried to cut out the two possible outliers, obtaining an almost identical result (unbalanced clusters and high Silhouette score).

2.3.6 Ward method

The ward method is certainly more interesting as it provided us with completely different results, returning more balanced clusters than the previous ones, although still remaining unbalanced (cluster0: 7971, cluster1: 1788) and with an average silhouette score(=0.5252010649169515).
It certainly proves to be the most robust method to the presence of noise and outliers, compared to the previous ones.
Furthermore, the division of the clusters, unlike the previous ones, does not have to do with the number of those involved or the number of participants but rather it more closely follows the average age of the participants.

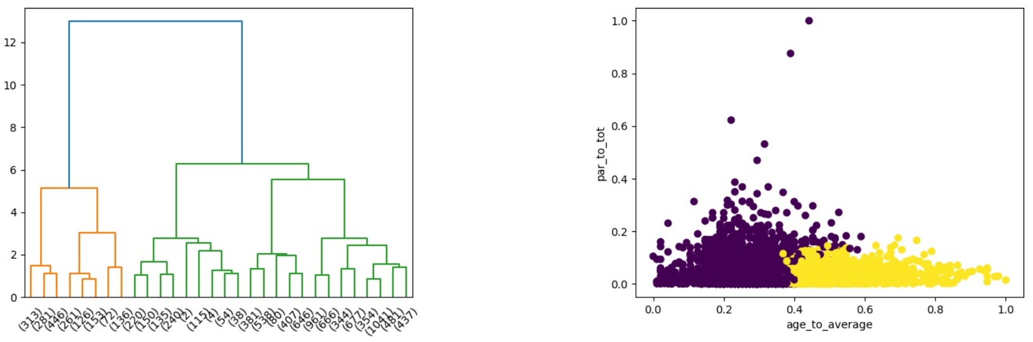


Figure 11: Results using the ward method

3 Predictive Analysis

3.1 Dataset Preparation

First, we converted a few categorical fields to int, then we generated a new feature called *any_killed*, such that we could use it with the classifiers.
We then selected a set of features to be used for the binary classification. To avoid data leaking, which would result in an useless classifier, we had to avoid features derived from *n_killed*. After some tests, we agreed to use a set of features that would cover space, time, number of participants by age and gender plus the number of injured and arrested.
We also added *povertyPercentage* as feature extracted from the poverty dataset.
The distribution of the labels in the dataset was around 70% incidents without deaths and 30% with at least one. We decided to test without the use of oversampling first and only use SMOTE later to artificially rebalance the dataset and assess the differences.

3.2 Dataset Split and Rescaling

We split the dataset in a dev and a test set with a 85-15 ratio, then the dev set into training and validation set to reach a 70-15-15 ratio, as some libraries integrate a split of the data and some explicit train-val division for cross validation.
The StandardScaler (Z-scale) was then applied to all the resulting datasets, using stratify to keep the label ratio consistent.

3.3 KNN

We tried two different approaches to choose the k for the Nearest Neighbours model, in the range of values between 1 and 20:

- Plotting the training and validation accuracy and then selecting the smallest k for which the distance between the two curves is low. For this, we selected $k = 8$.
- Using cross-validation scores with 5 folds and selecting the highest mean score, resulting in $k = 15$.

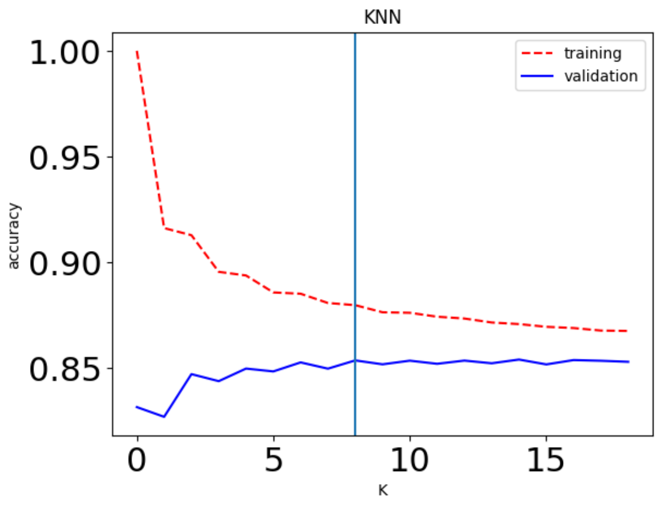


Figure 12: Plot of training and validation score relative to k

3.4 Bayesian Classifiers

For the family of Bayesian classifiers, we tried the Gaussian and the Bernoulli classifiers by using the train-validation sets, then checked the results of the better of the two on the dev-test sets.
Bernoulli gave a more balanced accuracy, but both had high accuracy on determining incidents with no death and low accuracy on the ones with killed individuals.

3.5 Decision Tree & Random Forest

For the Decision Tree and the other algorithms below we used a Grid search with kfold (k=5) on a wide set of parameters.
After CV, the model with the best set of hyper-parameters found is retrained on the whole

development set, then tested on test set. The models are all implemented using the Scikit-learn library.
For the Decision Tree the tuned hyperparameters are:
criterion, splitter, max depth, max features, min samples split, min samples leaf.
For the Random Forest they are:
n_estimators, max depth, max features, min samples split, min samples leaf

3.6 SVMs

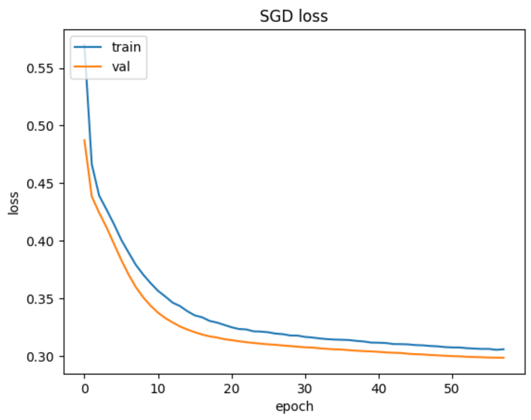
Since each kernel has different parameters to tune, we followed the same strategy as before but with a different set of parameters for each kernel.
The tested kernels were: rbf, sigmoid, polynomial and linear.
A maximum number of iterations had to be set as the SVMs were often not able to converge to a result in an acceptable time window.
In the end, the Linear kernel had a bit more luck, reaching almost 80% of accuracy, but still way lower than the previous classifiers.

Kernel	**Best parameters set found**	**Best accuracy score found**
Rbf	C: 10, gamma: auto, max_iter: 200	0.682
Linear	C: 5, loss: hinge	0.796
Polynomial	C: 1, degree: 4, gamma: auto, max_iter: 100	0.457

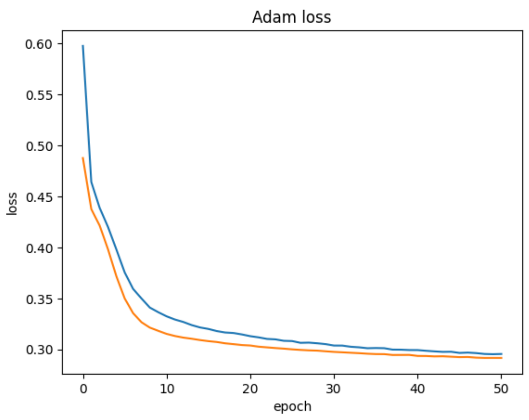
Table 5: SVMs Results

3.7 Neural Network

For the Neural Network we tried using the grid search + k-fold strategy again to find a baseline for the parameters but we also experimented with Keras, since the model from Scikit-learn was very limited in terms of customization.
Our model was made of three hidden layers, starting with 50 neurons, then two set of 25 with tanh activation, and a final layer with sigmoid activation.
A dropout layer (0.3 drop rate) was added before the sigmoid to help with regularization, and callbacks for dynamic learning rate reduction and early stopping, both based on validation accuracy, allowed us to have better control over the training flow and to stop before going into overfitting, setting a maximum of 150 epochs for the training.
The same model structure was tested with both SGD and Adam optimizers, with also a final retrain on the whole dev set for the SGD before testing on the test set.



(a) SGD Loss



(b) ADAM Loss

Figure 13: Comparison of Losses

3.8 Testing with SMOTE

We applied the oversampler SMOTE to balance the amount of data with positive *is_killed* labels to reach an even distribution, then reapplied scaling and splitting. We retrained the the best models to test the behaviour of the rebalanced dataset.

3.9 Results

Classifier	**Training**	**Test**	**Parameters**
Knn	0.882	0.855	k = 8
Knn	0.873	0.855	k = 15
Naive-Bayes	-	0.62	Bernoulli NB
Decision Tree	0.862	0.862	gini, max_depth:10, min_samples_leaf:10, min_samples_split:5, splitter:'best'
Random Forest	0.873	0.876	max_depth:20, max_features:log2, min_samples_leaf:2, min_samples_split:5, 150 estimators
SVM	0.795	0.814	linear kernel, 'C': 5, 'loss': 'hinge'
NN (SGD)	0.862	0.862	lr:1e-3, nesterov_momentum:0.9
NN (Adam)	0.862	0.861	lr:1e-4

Table 6: Results and parameters of best classifiers

From this first analysis we can see how the Random Forest gets the highest precision on the test set, with an accuracy of almost 88%. The worst classifier is Naive_Bayes, which only manages to successfully detect incidents without deaths, behaving poorly on the ones with killed individuals (92:23 for Gaussian, 89:35 for Bernoulli on dev set). KNN and NN seem to have pretty similar results, around 86%, slightly behind the Decision Tree which, although simpler, manages to get really good results.

Classifier	**Test Accuracy**	**Test Accuracy (no SMOTE)**
KNN (k = 8)	0.879	0.855
Bernoulli BN	0.75 (both classes)	0.62 (0.89:0.35)
Decision Tree	0.80 (0.806)	0.868
Random Forest	0.80 (0.798)	0.876
LinearSVM	0.858	0.814
NN (Adam)	0.884	0.861

Table 7: Test Accuracy of Classifiers with and without SMOTE

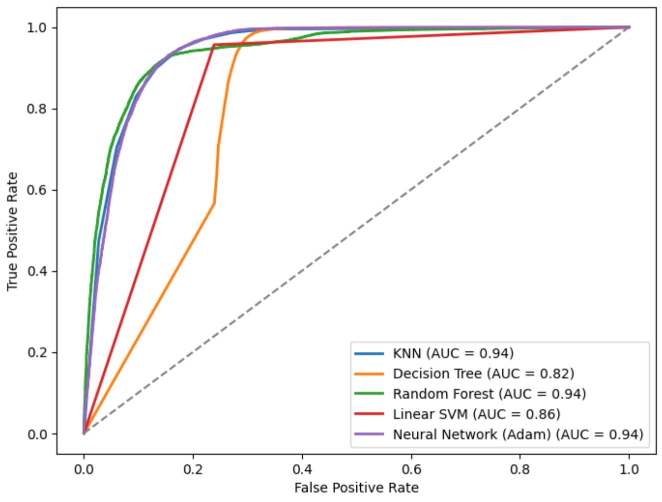


Figure 14: ROC Curves for classifiers with SMOTE

We only attempted to rerun the parameter search for RandomForest and Decision Trees, since they were performing poorly on the dataset rebalanced with SMOTE, but with no improvements on the test set.
The Bernoulli BN and Linear SVM get a big improvement, especially the BBN, which manages to get way more true positives on the incidents with deaths.
The NN and KNN manage to get the best results over both datasets, reaching 88% after SMOTE. So the oversampling generating such a big amount of data might have compromised the behaviour of models, which is why we ran the test without it first.

4 Time Series Analysis

4.1 Preprocessing

We first created the feature *is_killed* and filtered the data to be in the range of years between 2014 and 2017, as required.
Then we noticed that the *city_or_county* column had a lot of inconsistency; without any preprocessing some data about a single city would have been spread over multiple time series.
We used the fuzzywuzzy library to discover which entries might have been typos or equivalent names of the same city. Then, by using a regex to uniform certain words and a dictionary for the

discovered synonyms, we merged what was possible without manually scrolling over the whole dataset.
At this point we computed a score for each incident and a week index, since the time series had to span over multiple years and each point was a week.
We used the following formula for the score:

$$\text{score} = n\_participants + n\_injured + n\_killed \times \left(2 + \frac{n\_killed}{n\_participants}\right) - \frac{n\_unharmed}{4}$$

We gave more importance to the number of participants killed in relation to the total participants, and gave a penalty depending on the number of unharmed participants.

4.2 City Filtering

By calculating the number of incidents per week for each city and then dividing it by the total number of weeks we have, we get a ratio of weeks with incidents for each city.
We use this score to filter out the cities with not much data, by only taking the ones with at least 25% of the weeks having an incident.
From this filtered dataset, we group the incidents by week and city by summing the incident score to obtain our actual collection of time series.

4.3 Time Series Normalization

With the time series obtained, we start by searching for trends and stationary time series.
To do this, we use the Augmented Dickey-Fuller test.
In this case, no stationarity was detected, so we proceeded with denoising and scaling.
We used a window of size 3 for the denoising after seeing that the graph of SAD scores had a knee point around that window size, and then used the standard scaler for the time series to normalize it.

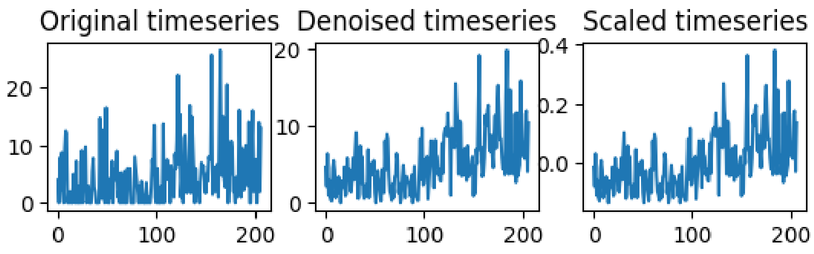


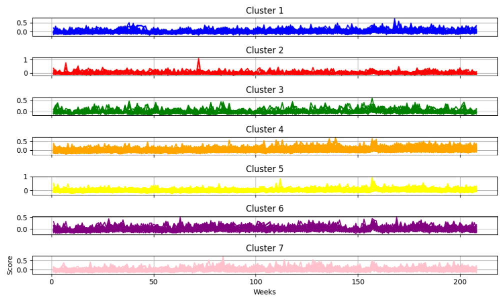
Figure 15: Time series of Austin before and after normalization

4.4 Time Series Clustering

4.4.1 Shape based KMeans

To select the best k for our K means algoritms, both the classic and DTW variant, we used again the combination of knn distances, silhoutte and dave-bouldin scores which hinted towards k = 7 and k = 6 in both variants.

The results are relatively balanced in terms of cluster size.
Euclidean Clusters size: (38,21,17,142,74,35,37)
DTW Clusters size: (17,79,72,109,57,30)


(a) Results of KMeans, $K = 7$

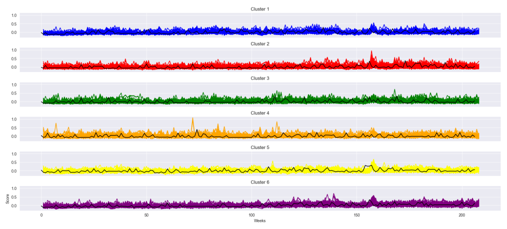

(b) Results of DTW, $K = 6$

Figure 16: Comparison of KMeans and DTW Clustering

4.4.2 Feature based KMeans

We used the mean, standard deviation, variance, median, 10th, 25th, 50th, 75th and 90th percentiles, interquartile range, skewness and kurtosis of the time series as features for the algorithm. We then tried to ran the KMeans (K = 5) and the Hierarchical clustering over those features. For the Hierarchical clustering we tested all linkage methods and both types of scaler, but ended up using the "Ward" method as the resulting clusters were more balanced. The results of the KMeans were pretty unbalanced (88,3,258,14,1) by having a two major clusters and three tiny ones, while the ward clustering only yielded two clusters.

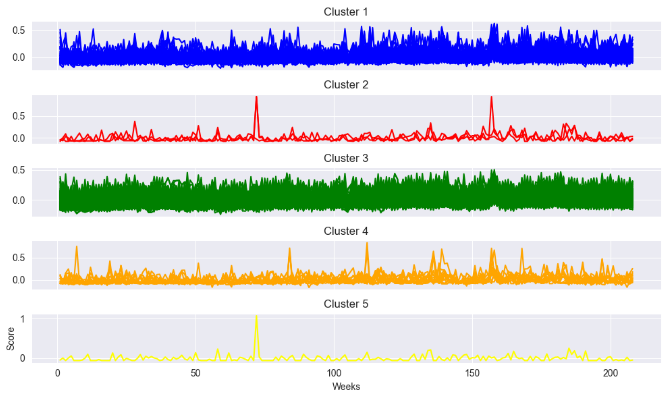


Figure 17: Feature Based KMeans

4.5 Cluster Analysis

We tried to look for geographic clues by plotting the cities on the map and checking the distributions of data over the clusters. Since the data is clustered by cities, the mean values for each cluster seem to be smoothed over a common average. Moreover, the bigger cities are split over multiples counties and part of their data might have been filtered out. The feature based KMeans have less uniform clusters so there are slightly more marked slices, but not notable differences.

4.5.1 Motifs and Anomalies

For this subtask we used the clustering obtained with the DTW KMeans. To search for the motifs of each cluster we had to select a candidate for each one, so we tried to find the time series most similar to the cluster's centroid shape.
We then ran the matrix profile with a window of 4 week(equivalent to a month) to search for motifs in each cluster and then for anomalies.

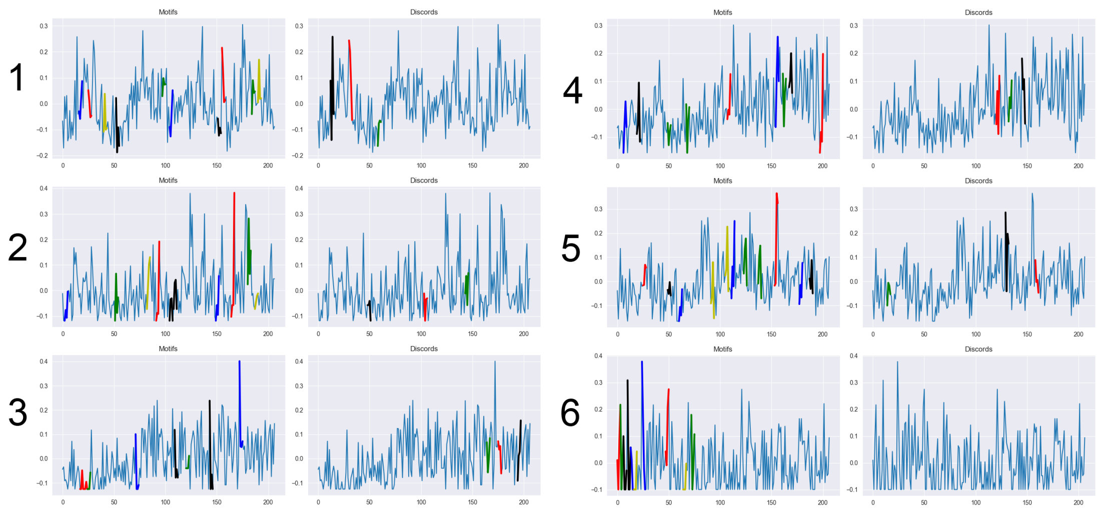


Figure 18: Motif and Anomalies

4.6 Shapelets

4.6.1 Preprocessing

Before finding shapelets in the time series, we had to create a new formula for the score, since the previous one was dependant on *n_killed*. The new formula for the score is the following:

$$\text{score} = (2 \times n\_minors) + (n\_participants - n\_minors)$$

The time series classes are defined on the variable is_killed, which is True when at least one person has died in a gun incident in that city. We did not remove any city from the list, to maintain a balance in the number of entries for both classes.

4.6.2 Calculating Shapelets

In order to find shapelets, we used Grabocka heuristics. After some testing, we found the optimal size to be 4, with 6 shapelets created.
We then trained a Shapelet model, using the LearningShapelets method from the tslearn library. We split the timeseries with a train-test split of 80%-20% and we achieved an accuracy of 56%.

4.6.3 Results

In the end, we obtained 6 shapelets, which we overlapped in this example on top of one of the timeseries.

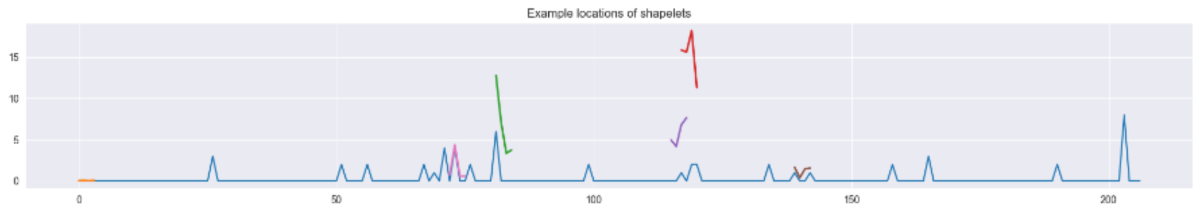


Figure 19: Shapelets on top of a time series

The extracted shapelets can be seen down below.

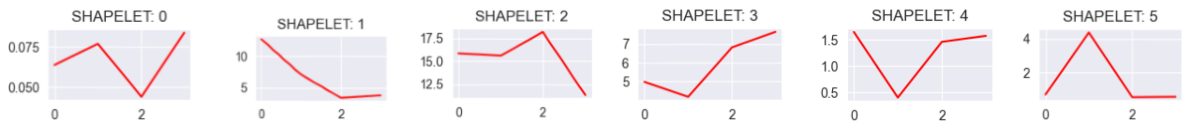


Figure 20: Extracted Shapelets