Comparing algorithms for storm cluster detection

Jonathan Cotasson

November 2022

Abstract

In the year 1951, natural hazards have had an important impact in the USA. Across the country, various types of storms or other events have damaged proprieties or crop. Tornadoes, hail and thunderstorm wind, hurricanes, avalanches, drought, and even more disasters occurred and impacted the life of billions of people. In this project, my aim is to use machine learning algorithms to define cluster zone of their appearances.

1 Introduction

This project will expose the methods of data acquisitions for natural hazards in the USA in a given period. I will then use this data to predict where those hazards tend to develop. A real world application for this environmental question would be to see where they do the more damage on proprieties or crop. To be able to find those clusters, I will use two different algorithms: K-means and DBSCAN. They both are unsupervised classification but the latter one is capable of dense clustering and not the former one. It will be interesting to see what both are capable to predict and how they differ as well. The clustering methods depends on each research and data set. The scientific literature [1] [2]also suggest other machine and deep learning method to analyze natural hazards. Regardless, this work's aim is to show what is possible to do with the chosen methods.
The scientific question is the following. Where did extreme event occur in the USA in 1951?

2 Data and Methodology

2.1 Data

For this research, I have gathered an important data-set from the National Oceanic and Atmospheric Administration (NOAA) which works for the US Department of Commerce. They freely give access to storm data in large "csv" files. In machine learning, the more you feed your algorithm with data, the more precise your results will be. I chose the data file with all occurrences of natural hazards in the USA in 1951. It is one of the earliest data sets available and I find it interesting to figure out what could the first results look like, regarding natural hazards.

2.2 Methodology

This whole script work is done with Python coding language As I mentioned before, I will use K-means and DBSCAN in this paper. Both are unsupervised machine learning algorithms. This means that the models are not supervised using training data sets. The models finds hidden pattern with the help of the given data, it groups data by finding similarities.

2.2.1 K-means

K-means is the first algorithms I want to use. It basically measures the mean squared error of each data point from a general average. In a 2D plan, it is the same but measuring the euclidean distance. In this project, I want to find storm clusters and

figure out in which region they are particularly important. For this algorithm, I have to randomly select distinguished data points and make then initial clusters. Each other points will then measure its distance to those initial clusters and be assigned to the cluster where they are the closest. When each data point is related to a cluster, I calculate the mean of each cluster and start over again. Each iteration should have different result at first but shows similarities after a few tries. Once the clusters no longer change, we may have found the real clusters. To find the right number of clusters, we have to try a few times the workflow above and plot the cluster means in an "elbow plot". In this type of plot, we can easily see were the elbow is and how many clusters, or Ks we should use.

2.2.2 DBSCAN

The second algorithm is DBSCAN, which means Density-Based Spatial Clustering of Applications with Noise. While K-means can be useful, DBSCAN works with density clusters and seems to be more efficient. This algorithm can handle "nested clusters". The first step of this method is to assign a few core points within our data. To find these core points, we will give each point a buffer zone of the same size. We then look at how many other points are overlapped by each point's buffer zone. The points with the more overlaps are the initial core points. Those points will then spread around to every point within its buffer zone until they can not spread any more. The the points that are nearly in the buffer zones of the points at the extremities of the clusters are non-core points. Those points will be assigned to the closest buffer without being able to spread even more. That is how density clusters are formed and we repeat the same with every initial clusters. The points not assigned to any cluster are the outliers, they represent extreme data and are not representative.

3 Results

After importing the data set on Google Colab, I can show you the variables interesting for this project. As we can read in Fig. 1, the main variables are the latitude, the longitude, the date, and the state. I have to specify that the coordinates are those of the beginning of each event. The end of some events did not have data so it was not important to mention them in this case.

	BEGIN_LAT	BEGIN_LON	STATE	BEGIN_YEARMONTH
0	33.50	-88.43	MISSISSIPPI	195109
1	39.07	-100.23	KANSAS	195106
2	31.37	-95.60	TEXAS	195103
3	34.45	-98.32	OKLAHOMA	195105
4	34.80	-94.80	OKLAHOMA	195107

Figure 1: Data set main variables

We saw in our book reference [3] in class and with our notebooks, how to create cluster plots with data. The next figure 2 expresses the clusters in reference with the latitude and longitude axis.

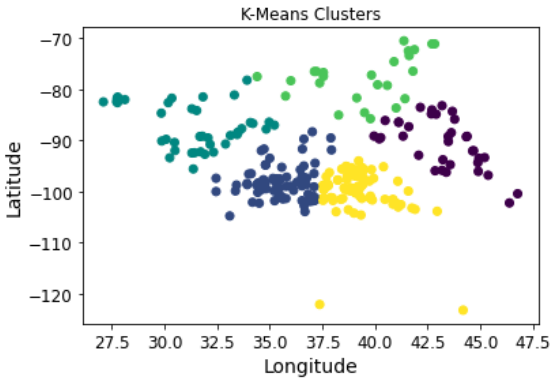

Figure 2: K-Means cluster plot

We can observe five different clusters in this plot. A number that could be correct based on the size of the region of interest and regarding the data set. We can verify if this number is correct using the elbow plot. This type of plot graphically shows the number of clusters that should be chosen for a given problem. As his name suggest, an elbow is visible

at the point where the optimal number of clusters is met.

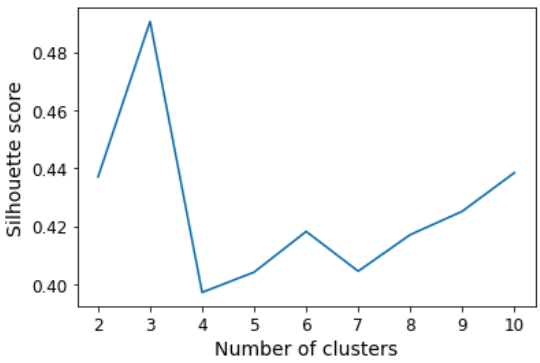

Figure 3: Elbow plot

This one can be tricky but 4 is the point at which adding additional clusters does not significantly improve the compactness and separation of the clusters, and is therefore a good choice for the number of clusters. I can now change the initial number of clusters and see want kind of results we obtain. The clusters seem to be well distributes, except for

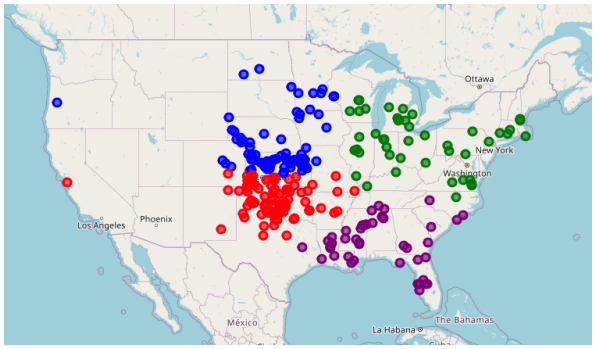

Figure 4: K-Means using 4 clusters on the USA map

the extreme values that have been assigned to the nearest cluster. We can also visualize the same result using DBSCAN. Note that this second algorithm automatically chose to perform with 5 clusters. The next step for this work is to evaluate the performance of those two algorithms. For this, I share a part of my code that enabled me to get the

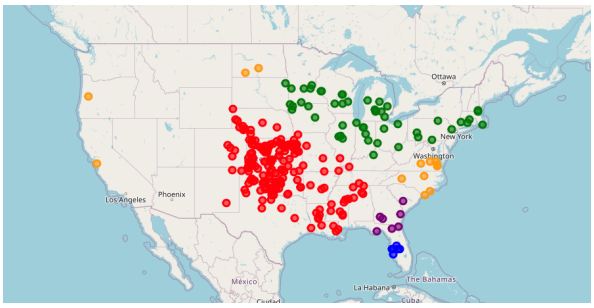

Figure 5: DBSCAN clusters on the USA map

scores of both algorithms. We can see on the the bottom of the figure 6 that the K-Means have a higher coefficient score than the DBSCAN.

```
# Select the latitude and longitude columns
X = df[['BEGIN_LAT', 'BEGIN_LON']]

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train a K-Means model with 5 clusters
kmeans = KMeans(n_clusters=5)
kmeans.fit(X_scaled)

# Train a DBSCAN model
dbscan = DBSCAN()
dbscan.fit(X_scaled)

# Evaluate the K-Means model
kmeans_score = silhouette_score(X_scaled, kmeans.labels_)

# Evaluate the DBSCAN model
dbscan_score = silhouette_score(X_scaled, dbscan.labels_)

print(f'K-Means score: {kmeans_score:.3f}')
print(f'DBSCAN score: {dbscan_score:.3f}')
```

```
K-Means score: 0.404
DBSCAN score: 0.381
```

Figure 6: K-Means and DBSCAN scores

The silhouette coefficient for a sample ranges from -1 to 1, where a value of 1 indicates that the sample is well-separated from the other clusters, and a value of -1 indicates that the sample is poorly separated and should be assigned to a different cluster.

4 Discussion

Regarding the results of this project, their are several information to discuss. First of all, the extreme values should not be taken into account. Specially using the DBSCAN method, we can see that the results can appear to be surprising. We can see that 4 out of the 5 clusters seem adequate, but the orange clusters is distributed across the USA map. Some of those values could, or should be assigned as outliers. This means that they should not be assigned to any of the clusters. Following the elbow plot 3, I could have chosen only 3 clusters as the silhouette score was the highest at that point.
The K-Means algorithm outscored the DBSCAN on this project. A reason for that is maybe that the region of interest was to large for a DBSCAN, as it operates better for dense clustering situations. One could easily see that the East side of the United Stated is at risk concerning natural hazards. The results show that only a couple of events occur on the West coast. They are extreme data as they do not fit in any clusters.

5 Conclusion

To conclude this work, I could advance that both algorithms were relatively accurate regarding their score 6. DBSCAN does not seem to fit well for this specific case as the data is not dense enough. Another algorithm I could have used is Gaussian Mixture Model. A Gaussian Mixture Model (GMM) [3] is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions. GMMs can be used for clustering by assuming that each cluster corresponds to a different Gaussian distribution. GMMs are good for finding clusters of arbitrary shape and can handle overlapping clusters, but they can be sensitive to the initial parameter estimates.
To answer the initial question of this paper, we can observe on the maps 4 5 that extreme events occur in the center and the East side of the USA.

6 Other resources

Link to the data-set used: here
Complete Python code: Click here for the complete Colab notebook

References

[1] D. J. Gagne II, S. E. Haupt, D. W. Nychka, G. Thompson, Interpretable deep learning for spatial analysis of severe hailstorms, Monthly Weather Review 147 (8) (2019) 2827–2845.

[2] M. Tonini, M. D'Andrea, G. Biondi, S. Degli Esposti, A. Trucchia, P. Fiorucci, A machine learning-based approach for wildfire susceptibility mapping. the case study of the liguria region in italy, Geosciences 10 (3) (2020) 105.

[3] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, " O'Reilly Media, Inc.", 2022.