Data Mining(NWI-IBI008) Final Report

Identifying Accident Prone Areas In The United Kingdom

Group 63
Matthijs Neutelings s4804902
Vamsi Yerramsetti s1032599

1. Introduction:

For over 15 years (2000-2016), the British government has collected data regarding traffic accidents. This resulted in an impressive database containing over 1.6 million entries, which can be accessed from the dataset repository Kaggle[2]. Attributes of the data include the longitude and latitude of the accidents and their severity. We want to use **DBSCAN** and a **unique K-mean clustering algorithm** to find out which locations have a high accident-density, and we will evaluate their performance by using cluster validity measures. We also want to investigate how efficient these algorithms are on such a large dataset.

2. Abstract:

We are attempting to solve the problem of identifying areas in the United Kingdom where the chance of involvement in an accident is high.
We are doing this by first collecting two sets of data which are the traffic flow and the number of accidents in the UK. We then normalize and compress this vast data and combine the two sets of data based on their coordinates. We then apply a DBSCAN clustering algorithm and obtain 7 clusters or areas where the chance of accidents is high. For our second clustering algorithm, we use a custom K-means algorithm. We initiate K to 100. So we get the UK split into 100 areas of high accident density and then Select the highest density clusters. We again get 7 areas where the chance of accidents is high. We compare the results of both algorithms to obtain a list of the most dangerous areas to drive. We observed that in both algorithms the areas/clusters we obtained were near to identical showcasing the efficiency of our algorithms. We also observed that these areas were not just the obvious populated cities/towns but areas where the risk of an accident is high.

3. Application domain and the research problem:

Our application domain is the British Parliamentary Advisory Council for Transport Safety (PACTS) and the Department of Road Traffic UK. They supervise the safety improvements and projects concerning transport by roads in the UK. With our results,

they can identify the areas the chance of future accidents is high, and what areas in the UK need improvements in their road safety taking into account the traffic flow of the area. The clusters we got to showcase the areas in need of road-safety improvement. We are solving the problem of identifying accident-prone areas in the United Kingdom with the highest chance of getting involved in an accident.

4. **Previous Attempts to solve the problem:**
   Research conducted by professors at Northwestern University used a K-mode clustering algorithm with association rules and used Adaboost to find road accident patterns. They primarily focused on predicting future accidents and under what conditions they will occur in the UK by using support vector regression. They took multiple factors into account such as weather and terrain type. They used heat-maps to represent their data.[1]

5. **Database and techniques we used:**
   1. **Data Set:**
      We used the data set from Kaggle which is our traffic accident data with over 1.6 million traffic accidents in the UK.[2]

      The other data set we used is from the Department of Road Traffic UK for the Traffic flow data.[3]

   2. **Data Collection:**
      We downloaded the data from the dataset at 3 different points of time to strengthen its credibility and uploaded it into our jupyter-book from the above resources. From there we began our preprocessing methods.

   3. **Preprocessing Methods:**
      We used a few preprocessing methods. To normalize the data we first rounded the coordinates of the locations where accidents occurred to two decimals. We then started grouping the coordinates that were the same. This gets rid of duplicate/close-proximity accidents in the same area. This made a huge difference as the number of accidents reduced from around one million to a little over 100,000.
      We then combined the two sets of data. For each data point in the "count of accidents" dataset, we matched it with its nearby traffic flow using the coordinates for both. We used `locations.dropna` to get rid of data points that had no value.
      This is helpful when we did our DBSCAN which we will explain further below.

6. Our Approach and Reasoning:

1. Criteria for the algorithms

Our goal is to find the places in the UK where traffic accidents are most likely to occur. The input for our clustering algorithms will be a list of longitudes and latitudes and a weight for each location that indicates how likely a traffic accident is to occur at that place. This weight is equal to the total amount of accidents in that location divided by the traffic flow The reasoning for this is that the same number of accidents should have more weight in a place with a lower traffic flow than in a place with a high traffic flow. For the output, each cluster should represent a location with a relatively high amount of traffic accidents. Since we want the clusters to coincide with locations, we want them to be rather small, and since we want only to find locations with an elevated risk, most data points should not be allocated to a cluster.

In short, we want to use algorithms that satisfy the following criteria:

- The algorithm should be able to handle weighted input
- The clusters should be small
- Most data points should not be allocated to a cluster

These considerations lead to the following choices:

2. DBSCAN

DBSCAN , which stands for Density-based spatial clustering of applications with noise, seemed like the most fitting algorithm to us for this problem. To see why, let us have a look at the steps of the algorithm. The parameters that are used are written in cursive.

1. Select a random data point x that is not part of a cluster yet.
2. Find the total number of points that are within distance *epsilon* of x. This is called the epsilon-neighbourhood of the point.
3. If the epsilon-neighbourhood of x contains more points than *minpts*, or in case of weighted data, if the sum of the weights of the data is higher than minpts, we label every point in the epsilon-neighbourhood of x as part of the same cluster. Otherwise, x gets labeled as noise.

4. We repeat step 2 and 3 for every other point in the epsilon-neighbourhood as x, and for every new epsilon-neighbourhood that gets discovered this way.
5. If there are still unlabeled points, we go back to step one. Else, the algorithm terminates and returns a list of labels.

As the name of the algorithm implies, it returns clusters of high density. This also becomes clear in the steps the algorithm takes. Regions with high density are marked as a cluster, and get expanded while the density remains high. This means that with the right choice for epsilon and minpts, we obtain a list of small, high-density clusters, with most of the data getting labeled as noise.

3. KMeans

The standard KMeans algorithm does not match all of the criteria mentioned in 6.1. To see why, let us have a look at the steps of the algorithm. Parameters are again in cursive.

1. Select *K* initial (random) cluster centroids
2. Allocate each data point to its nearest centroid
3. For each centroid, move it to the mean of all the points that are allocated to this centroid. If we have weighted data, data with a higher weight has more impact on where the mean is.
4. Repeat steps 2 and 3 until the centroids in step 3 move less than a certain threshold, which indicates that the centroids have converged.

Note: while the centroids will always converge, their final location is highly influenced by the initial choice of the centroids. For this reason, KMeans is run multiple times with different initial centroids.

Let us take a look at the criteria again. KMeans allows for weighted input, which is critical for this problem. However, since every point of the data set will be allocated to a cluster, the third criterion is not satisfied. We can satisfy the second criterion by choosing a high value for K. This leaves us with a high amount of small clusters, covering the entire dataset. We then select the top *x* clusters with the highest densities from among these clusters, and mark all other clusters as noise. In this way, we make sure to end up with only a small amount of dense clusters.

4. Interpreting the clusters

The output from both algorithms gives us several small clusters, each of which we allocate to a location. These locations will then be, according to our metric, the areas where a driver has the highest chance to get involved in an accident.

7. Results:

1. DBSCAN

We ran DBSCAN with the following parameters:

- epsilon = 0.1
- minpts = 10

Since the coordinates are rounded to 2 decimals, this means that the radius of the epsilon neighbourhood is 10 points large. Our choice for minpts indicates that the total weight of the points should be larger than 10 for an epsilon-neighbourhood to be considered part of a cluster. For comparison, the average weight of the input is around 0.01. Running with these parameters gave the following clusters:

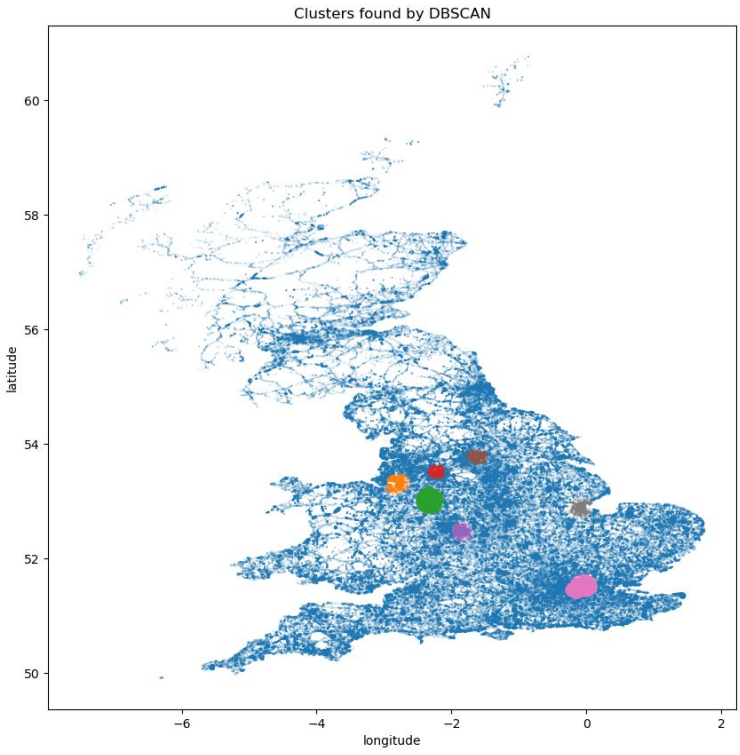

Image 1: Clusters found with DBSCAN

2. KMeans

We ran KMeans with the following parameter:

- K = 100

This gave us sufficiently small clusters. We then selected the top seven highest density clusters from among those, since we wanted to compare them to the seven clusters that DBSCAN returned. This gave the following results:

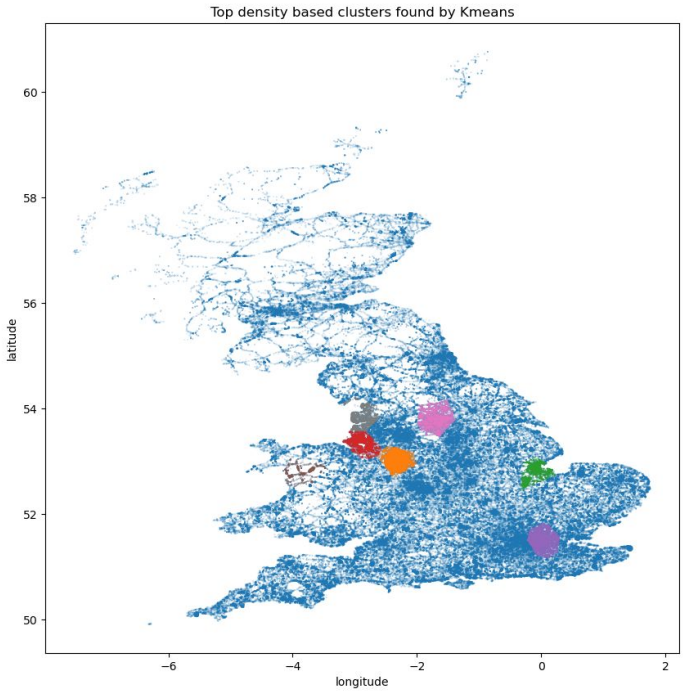

Image 2: clusters found with KMeans

3. Comparing results

Most clusters are represented in the output of both algorithms. The locations that correspond to these clusters are London, Liverpool, Manchester, Stoke-on-Trent (just below Manchester and Liverpool) and Boston (at the coast). According to this work, these are the places where drivers have the highest chance of getting involved in a car accident.

8. **Analysis of the results:**
   Our final list contains three big cities and two smaller areas. When looking at the clusters, it stands out that the clusters corresponding to big cities have more points with a smaller weight, and those corresponding to the smaller locations are heavily influenced by a single data point with a big weight. This is what we expected to happen when we corrected for traffic flow, and we are happy that we obtained a list containing a list of big and small names. Even though two of the clusters were essentially carried by a single big data point, we feel like this is not necessarily a reason for concern since a large data point consists of multiple accidents happening in an area with low traffic flow.

9. **Personal Evaluation of the Results:**
   We were very happy with our results. The near to identical similarity in the results of the two clustering algorithms show that our idea and approach was in the right direction and that we have, to a large extent identified the top 7 optimal clusters showcasing areas where the risk of an accident is high. Hence we feel we have met our expectations with little to no gap.

10. **Potential Future Improvements:**
    Euclidean distance is not optimal when using longitude and latitude coordinates, since we don't take the curve of the globe into account. The haversine metric computes exact distances between two points with longitude and latitude coordinates, so this would be an improvement to our approach, although we doubt that there would be significant differences.

    The KMeans algorithm seems to give slightly different results every run. It would be best to combine these results into a more robust result instead of selecting one of the outcomes like we did.

11. **Insight of the code:**
    We added our jupyter notebook which gives a clear insight of the code. Note: to run the notebook the data has to be downloaded from the given sources. The files were too large to submit to Brightspace.

12. **References:**
    [1]https://www.researchgate.net/publication/341665300_Towards_Big_Data_Analytics_and_Mining_for_UK_Traffic_Accident_Analysis_Visualization_Prediction

[2]https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales

[3]https://roadtraffic.dft.gov.uk/downloads