Final Project: UK Traffic Accidents

Crystal Warta Sujan Shrestha

***Abstract*—In this project, we utilized a substantially large data-set that included details on UK traffic accidents. With this information, we created a database, performed data analysis and visualization, and then created a web app to show our findings. The goal of the project was to facilitate the process of deriving business intelligence from a large data-set.**

I. Introduction

For this project, we were given the task of finding a large, high-dimensional data-set which may not be clean or may have lots of anomalies/outliers. We were to also design and create a database with the data by following the principles of end-to-end data analysis workflow, use data analysis and data modeling techniques in order to do clustering, anomaly detection and visualizations. We were also instructed to enable web-based visualization by creating a JavaScript based web-application that utilizes D3 API. We first created the database, then started clustering, doing anomaly detection, forming clusters and visualizations, and thus created our JavaScript based web-application. Even though our team had done each of these steps before individually, we had never done each of these steps together on this big of a dataset so it was definitely a learning experience.

II. Project Description

A. Short description of the project

For this project, we chose a very large data-set (including more than 400000 data) provided from the UK government that consisted of details on all traffic accidents that occurred in the country from 2012 to 2014. After processing or cleaning the dataset, We stored the data in a No-SQL database: Firebase Real-time Database. Then, we used python language to write scripts to get the data from the database and perform data modeling and data analysis. When doing this, we used GMM to get clusters of the data, and used Isolation Forest to find the anomalies in the given data-set. With top 20000 outliers that were found, we decided to drop them since those outliers would only contribute towards abnormality in the report. We finally used this information to create a web app using D3 API that included easy_to_read, simplified visualization.

B. Why this problem is important

A total of 1,784 people died from traffic accidents in the UK in 2018 [1]. Creating a model from the data that is easily readable could raise awareness for both the UK government and its citizens to create a safer driving environment and to enforce safer driving rules and regulations.

C. Why is this problem of interest to us

In today's world, data is everything. Whoever cracks the hidden information inside data, rules the world. However, data without a meaning is useless and has no value. During our project, we were really intrigued by the data that we chose. We realized that using data modeling and analysis techniques we could find hidden patterns or meaningful information from the data-set and this information could be highly valuable in saving many lives. For example, when modeling and analysing the data-set, we found a correlation between accident severity and the speed limit of the road the accident occurred on. We noticed there was a certain speed limit, predicted to be the safest speed limit to drive on certain road. Once we had this information we knew we could present our findings in a way that would be easily readable, understandable from a business perspective, and would highly impact on safer drivings.

D. E-R data analysis workflow

1) E-R Database: We used the standard Entiry relation techniques to model our database. We began with conceptual modeling which categorized the data into four separate entities linked with certain relationship. The, during logical modeling we got normalized our data and got rid of any redundancies in the data-set. We also identified all the necessary attributes that would completely identify each entity. And, at the final stage of physical modeling, we denormalized our data as per the business requirement and stored it in a No-SQL Firebase real-time database.

2) Web Server: Currently, our database is being served globally by googles' firebase db server. In our project, we are using firebaseDb-admin-SDK which is a firebase library to run CRUD queries and retrieve real-time data accordingly from the firebase database.

3) Internet and Mobile Client: Firebase db server can serve the database to any mobile client provided that they are connected to internet. The client that's connecting to our firebase database however, will require a service.json key which will give admin privileges to the client.

4) Data Analytics and User Experience: During this stage, we processed the data retrieved from the firebase server and generated different clusters per given feature set. And that result was was being served as several visual charts in the form of a web page index.html. We used HTML for creating the index.html page and D3js library for generating the visual charts, and CSS for styling the graphs and the web page.

III. Data set

A. Source of data-set

We collected data on accidents from Kaggle.com. It is an online data repository free to any user. URL: https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales.

B. Description of data-set

On a high level, our data can be classified into 2 separate categories. i.e. textual and numerical data. Attributes such as accident_severity, number_of_vehicles, number_of_casualties, speed_limit, longitude, latitude etc. fall under the numerical data-set where as road_surface_condition, weather_condition, light_condition, special_condition_at_site, urban_or_rural etc would be a part of textual data. The data-set we have is a mixture of discrete, continuous and categorical data as accident-severity can be either 1,2, or 3 but cannot be 2.5. Similarly, latitude, longitude are measure in decimals which is why they would fall under continuous data. And, each the numerical data carry certain meaning with it. For example, an accident with accident-severity 3 would belong to high accident severity category.

C. Type of data

The data that we have belongs to the structured data type family, since they follow certain structure. The data we collected from Kaggle was in the form of csv. The data was divided into different categories and separated using commas. Because of that, we can structure and organize data anyway we want. For example, we can organized the entire data-set in ascending order of speed-limit because of the fact that the data is well-structured.

IV. Data Modeling for storing data into Database

A. Type of database

The database we used to store and organize the data is Firebase real-time database. It is a No-SQL real-time database that supports real-time queries. i.e. real-time reads and writes, updates and delete. It basically it supports all of the CRUD operation in real-time. It is very similar to MongoDB database in a sense that it stores each data item as a single document. Each document is organized via uniquely generated key inside a collection. Firebase real-time db is atomic, just like MongoDB, since it provides atomic operations on a single document.

B. Data modeling step - Conceptual Modeling

Based on the data-set, we realized there needs to be at least 4 separate entities (accident, location, road, and weather). These 4 entities can carry all the information that would describe each accident in the data-set. In the original data that we collected from Kaggle, we noticed there were far too many repetitions for certain categories, so we decided to separate out some them into its' own entity. For example: accident, location, road and weather were all inside one big entity accident in the original data-set. And since there is a possibility of more than 1 accidents occurring at certain location, this would cause the location information to be repeated in many places inside the data-set. Similarly, a road can have many locations and be a part of different accidents. Because of this behavior, we came to a conclusion that it is best to have the four entities mentioned above and have some kind of relationship between those entities.

C. Data modeling step - Logical Modeling

Once we completed the conceptual modeling of our database, we quickly realized that our database had to be normalized in order for the conceptual modeling to work. We first identified all the attributes that would describe each entity in our database. So, for accident entity we identified accident_index, accident_severity, number_of_vehicles, number_of_casualties, date, and time as its attributes. For location entity, we identified longitude, latitude, location_easting_osgr, location_northing_osgr, urban_or_rural etc as its attributes. For road entity, we identified road type, road_surface_condition, special_condition_at_site, light_condition, junction_control and many more as its' attributes. After all the attributes were identified for each entity, we then assigned a primary key attribute to each entity which would identify unique combinations of each attributes for that entity. After all the attributes were established, we then performed normalization on our database to reduce or completely get rid of any redundancies that existed in the data-set. With this we ensured that each accident in our database has been uniquely identified and there is no possibility for duplication. After, normalizing we then identified the detailed the relationship between each entity. We have a strong many to one relationship between accident and location entity as seen in the ER diagram. This restricts any possibilities of an accident record existing in the database without a location, because in reality an accident cannot exist without a fixed location. And, we established a many to one relation because we realized from our data-set that a location can have witnessed multiple accidents, and thus the many to one relation. Same is the case with location and road, and location and weather. After the successful logical modeling we were left with the following ER diagram

D. Data modeling step - Physical Modeling

During this stage, we had to make business decision of how to store the data and where. Thankfully, the choice for us was very simple. We chose a No-SQL database because of the

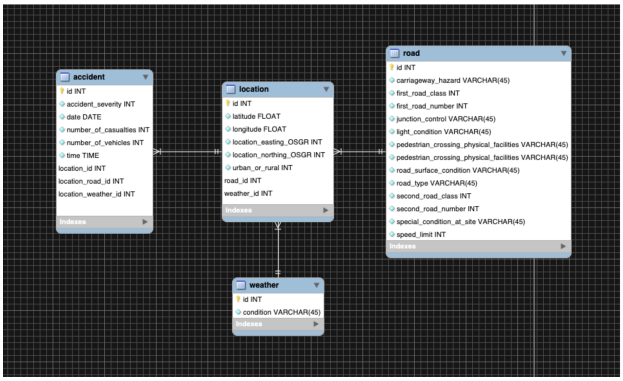

Fig. 1. ER Diagram of our NoSQL Database

sheer size of data we were dealing with. With almost half a million data entries needing to be organized and stored in an efficient manner, only a NO-SQL database could handle such load in efficacy. During this stage, we dropped some data entries that didn't match the restrictions that we had set in our database during first two stages of data modeling. After the bad data were skimmed off, we were left with almost 300000 data entries which needed to go inside our database. But, before inserting any data in the database, we made a business choice of de-normalizing our database. This choice was driven by the fact that we were using a No-SQL database and it made the most sense as per the business requirement from the database.

V. Data Modeling for Analysis

A. Justification of the type of analysis used

When deciding what type of analysis we should use for this project, we had to take into consideration what kind of data-set we had. As we mentioned earlier, we had 2 types of data in the data-set, textual and numerical. So, the first thing we did was a simplified version of feature extraction for the textual data. During the extraction, we created an array of numbers for unique text items inside each feature, and those numbers would be used instead of the actual text data during clustering.

When considering what kind of clustering to perform on our data-set, we had three choices: K-means, GMM, or DB-SCAN. They all obviously have their upsides and downsides. We knew that K-means is a faster algorithm compared with GMM [2], which was a big advantage for us since our data-set was very large. However, we also knew that GMM generally performed and found accurate clusters better than K-means. We decided that we preferred GMM over K-means for this reason. Then we had to choose from GMM and DB-SCAN. One drawback we knew of DB-SCAN is that it might not perform very well if our clusters had varying density [2]. Looking at the data we realized there would probably be some variance in density among our clusters, so with that in mind, we decided to use GMM.

Since we also had such a large data-set (over 400,000 data), we knew we had to remove some of it, otherwise the GMM algorithm we would run would take way too long. Also, because we were using a Firebase database, we realized we could only access a certain GB of data every month before we had to pay for a membership. Thus, we decided to remove all data that had null values in certain fields in the data-set. After that, we had almost 300000 data left. And furthermore, we chose to use a third of that data-set for clustering because of the time-complexity. We then removed the top 2000 anomalies from the data-set using the Isolation Forest technique.

B. Description of the type of analysis used

Before we decided the type of analysis we were going to use, we had to decide which features of the data we would use to perform clustering. We noticed we had over 30 features in our data and GMM would take a substantial amount of time to run if we were to use all the features. Apart from that, we chose to isolate our clustering and analysis to only certain features for better quality result. This gave us an edge both in the time complexity and it generally produce a better quality result. For example, isolating our feature set to accident severity and speed limit gave us better clusters in less time than using more features. And, since our end goal for this cluster was to find if the speed limit did affect accident severity, using only those 2 features to perform clustering made the most sense. We took the same approach with other feature sets to perform clustering as well.

When first starting to build the GMM algorithm into our project, we used the Bayesian information criterion (BIC) and Akaike information criterion (AIC) in order to find optimal number of components to use in GMM. However, we found that when using Python's Sklearn module, it always gave us a really high optimal number of clusters. In fact, when giving the algorithm a range of optimal clusters it could choose from, it almost always chose the highest number we gave it. We knew too many clusters wouldn't be very useful to our analysis as too many clusters would not be very easy to model and in each cluster the data would be very similar to each other. Thus, we decided to then try finding the optimal number of clusters using the Silhouette score. Using the Silhouette score to find the clusters gave much better results.

C. Modeling techniques used for text data

For text data, we decided to make each type of text have their own number. For example, in Figure 12 that Frost/Ice is type 0 road surface condition, Snow is type 1 surface condition and so on. In this way were able to easily model the graph.

VI. Data Modeling for Web-Based Visualization

A. Dimension of the data

Our data-set was high-dimensional. However, we decided that the best way to represent and to make sense of the data was to make strictly 2-dimensional graphs.

B. Dimensionality reduction technique used

No dimensionality reduction technique was used.

C. Data models of D3 used

For data modeling using D3, we decided to use bar charts and line charts. We used bar charts specifically to compare the sizes of the clusters. We used the line charts to show if y-axis was directly proportional to any changes in x-axis. i.e. if x goes up, y goes up and if x goes down, y also goes down. So, in the case of silhouette score vs cluster line graphs, we can see the silhouette score increase with increasing number of clusters. The idea, behind this was to help the business user visualize the increasing or decreasing curve if any for the given model.

D. Description of the data models

For the accident_severity and speed_limit, using the Silhouette score and GMM, we wanted to see different clusters predicted by the algorithm. We used the standard Sklearn's Silhouette score algorithm to find the optimal number of clusters. As you can see in Figure 2 where we graphed the Silhouette Score using D3 in relation to the number of clusters, the Silhouette score increased with increasing clusters. We also used GMM to predict the optimal number of clusters, and the we chose the best out of those two. In Figure 3, you can see that we then graphed the number of accidents that were in each cluster. The first cluster had over 65,000 accidents in it, while the other clusters had much less, with the minimum cluster having only 434 accidents. In Figure 4, we have a graph that shows the average accident severity by speed limit for each cluster. For example, in cluster 5, the average speed limit was 20 and average accident severity was 2.8. Cluster 0, 6, and 8 also had an average speed limit of 30, however, they all consisted of different accident severities of 3, 2, and 1 respectively. Once we graphed all this figure also Figure 3 which consists of the sizes of the clusters, we start to notice the trend in the data. For example, most accidents occurred at 30 miles per hour. It interesting to see that most 20 mile per hour accidents were very severe, we were not expecting this. This tells me that the accident_severity may or may not be directly proportional to speed_limit. Although, noticed in general higher the speed limit, the higher is accident severity, but we also noticed an exception: in cluster 1, accident severity was only at a level 2 for a high speed limit of 60. This gives us a business intelligence that driving at speed_limit is generally safer depending on whether driving in a city or a highway.

For accident_severity and time of day, using the Silhouette score and GMM, we wanted to see if the accident_severity is any way related to time of the day. In Figure 5, you can see the Silhouette Score in relation to the number of clusters. Our program then decided that the optimal number of clusters was 8. In Figure 6, we have our graph that shows the number of accidents that were in each cluster. As you can see, cluster 2 has over 55,000 data, while the others have much less. In Figure 7, we plotted the average accident severity by time for each cluster. We noticed a very interesting fact, at around 11:00am (cluster 2) we see the accident severity was 3 with a total of 55,000 accidents. This tells me that driving at around 11:00 am (lunch hour) wouldn't be a safe decision. Same was the case for rush hour (4:00pm-5:00pm), we saw the accident-severity was really high. We did notice accident severity was high around 8:00pm but the number of accidents was low so it could just an abnormality or could be driven by the fact the may be the light condition on those roads where the accidents occurred was bad which we will investigate in another model.

For the next cluster set, we wanted to cluster using the features accident_severity and number_of_vehicles in the accident. In Figure 8, you can see the Silhouette Score in relation to the number of clusters. In Figure 9, we have our graph that shows the number of accidents that were in each cluster. As you can see, cluster 0 has over 60,000 data, while the others have much less. In Figure 10, we plotted the average accident severity by number of vehicles involved in the accident for each cluster. By looking at the graph, you can see there is quite a zigzag but in general, the accident severity increases as the number of vehicles involved increases.

In the next cluster set, we wanted to cluster using the features accident_severity and road_surface_conditions. In Figure 11, you can see the Silhouette Score in relation to the number of clusters. Our program then decided that the optimal number of clusters was 8. In Figure 12, we plotting the average accident severity by average road surface conditions for each cluster. In the small table at the bottom of the figure, you can see that this graph is slightly different that the other graphs. We had text data for this graph, thus we will generally not see any increase or decrease. By looking at the graph, you can see that when weather conditions included frost or ice, the average accident severity was really high. Having snow also made accident severity slightly increase. For Wet/Damp and Dry conditions of the road, we noticed the accident severity fluctuated between 3(high) and 1(low), which could have been affected by other categories of the accident.

For the next cluster set, we wanted to cluster using the features accident_severity and light_conditions. In Figure 13, you can see the Silhouette Score in relation to the number of cluster. Our program decided that the optimal number of clusters was 7. In Figure 14, we plotted the sizes for each cluster. In Figure 15, you can see that accident severity was fairly high when the road had street lights, but they were not lit or there weren't any lights at all. And, the roads that had proper light conditions had less accident severity as seen in figure 15.

For the last cluster set, we wanted to cluster using the features accident_severity and weather_conditions of the accidents to see if there exists any relation. In Figure 16, you can see the Silhouette Score in relation to the number of clusters. In Figure 18, we plotted the average accident severity by the weather condition during the accident for each cluster. By looking at the graph, you can see that in most conditions, the accident severity was pretty high, except for when the weather was fine without high winds, there is a sharp decline in the accident severity. This is exactly what we were expecting from our model.

VII. Conclusion and Future Work

GMM was successful in finding only some meaningful information hidden inside the data like the correlation between accident severity and time of the day accident occurred, which was really interesting finding. But, GMM failed to detect pattern between accident severity and road surface conditions and others. Both GMM and the Silhouette score failed to predict the actual number of clusters for the given features, which might have contributed negatively to the result. This, tells me in future, may be we can use another clustering algorithm with a better refinement in the data to obtain more quality information.

References

[1] "Road safety questions and answers." [Online]. Available: https://www.racfoundation.org/motoring-faqs/safetya1. [Accessed: 27-Apr-2020]

[2] Olkhovets, "Why You Need Multiple Clustering Algorithms," LinkedIn, 02-Jun-2017. [Online]. Available: https://www.linkedin.com/pulse/why-you-need-multiple-clustering-algorithms-anatoli-olkhovets/. [Accessed: 27-Apr-2020]

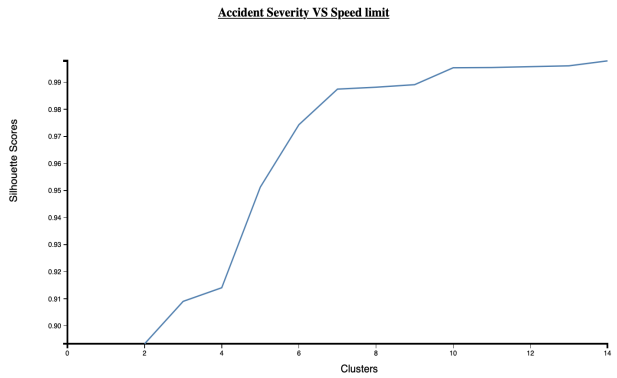

Fig. 2. Accident severity vs speed limit: Silhouette Scores over Clusters

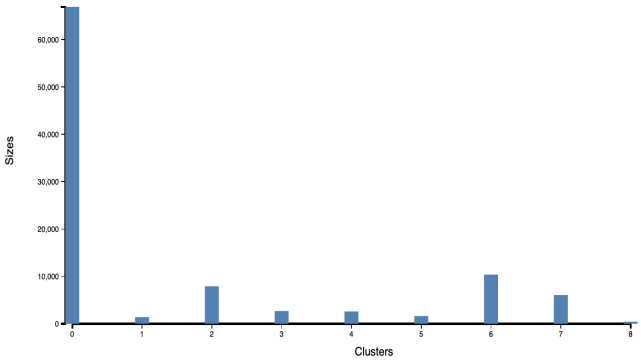

Fig. 3. Accident severity vs speed limit: Cluster sizes

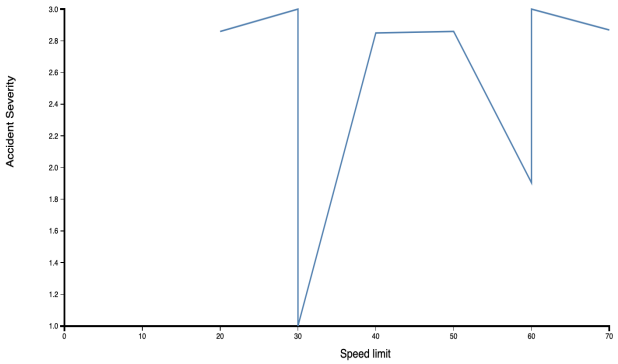

Fig. 4. Accident severity vs speed limit: Averages for each cluster

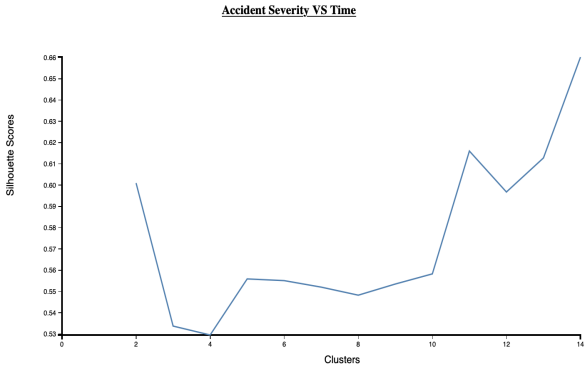

Fig. 5. Accident severity vs time: Silhouette scores over clusters

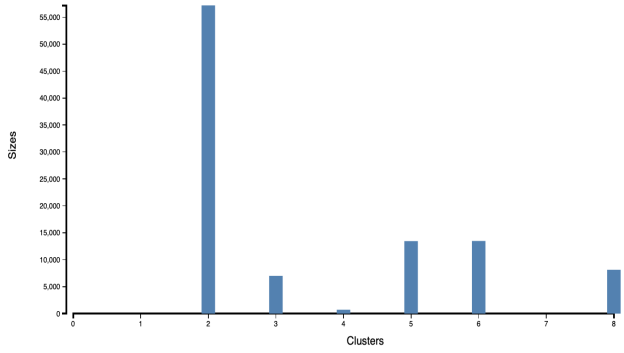

Fig. 6. Accident severity vs time: clusters sizes

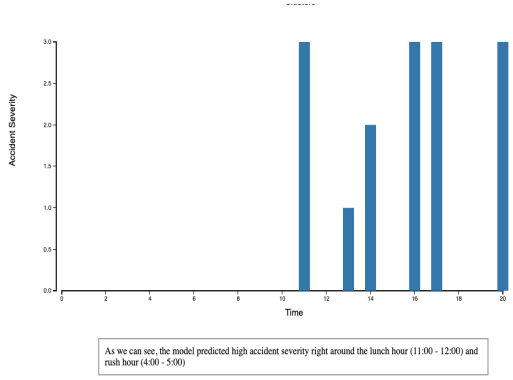

Fig. 7. Accident severity vs time: Averages for each cluster

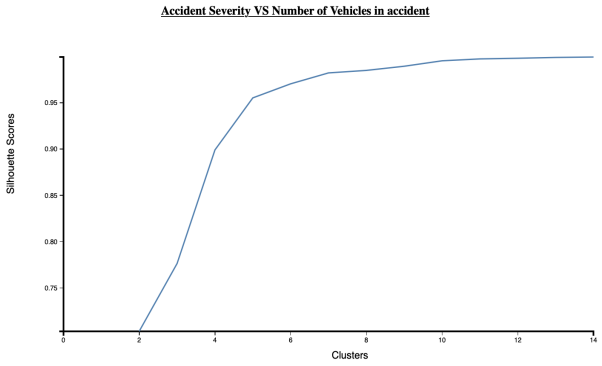

Fig. 8. Accident severity vs Number of Vehicles: Silhouette scores over clusters

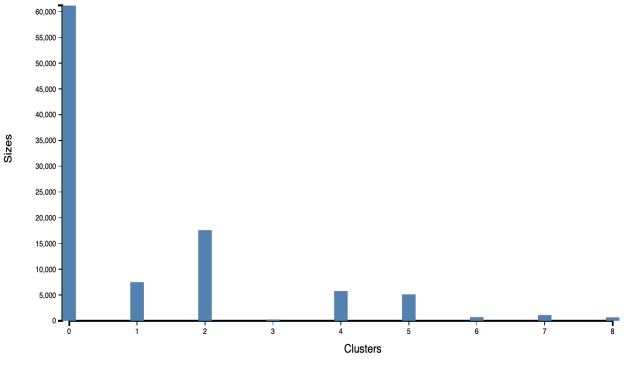

Fig. 9. Accident severity vs Number of Vehicles: cluster sizes

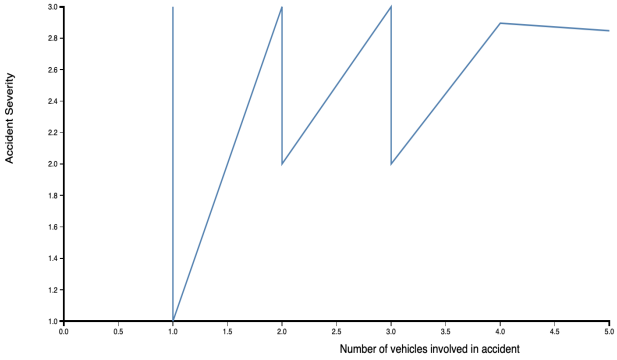

Fig. 10. Accident severity vs Number of Vehicles: Averages for each cluster

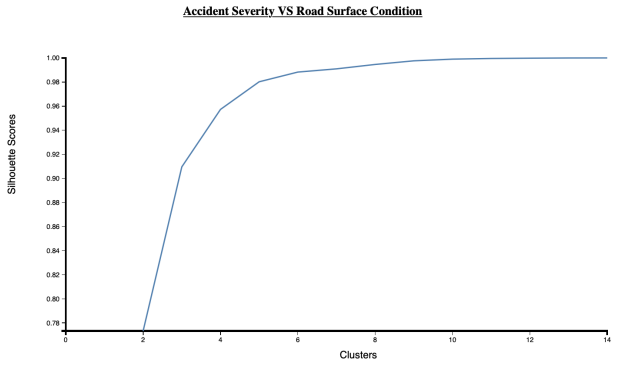

Fig. 11. Accident severity vs road surface conditions: Silhouette score over clusters

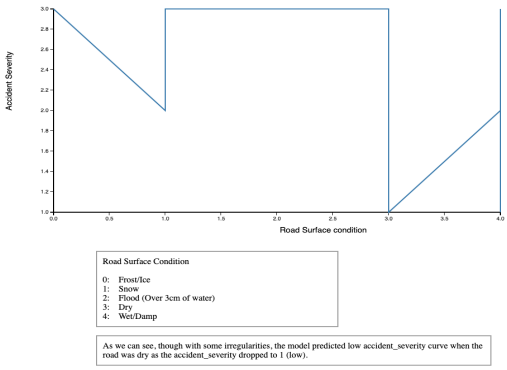

Fig. 12. Accident Severity vs road surface condition: Averages for each cluster

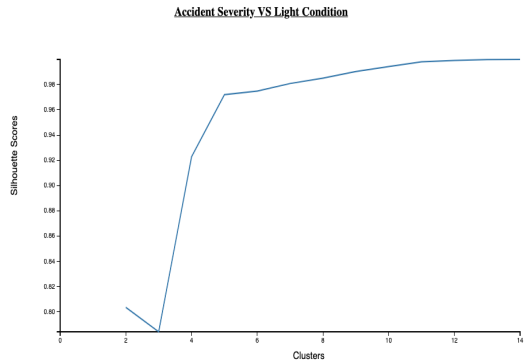

Fig. 13. Accident severity vs Light conditions: Silhouette scores over clusters

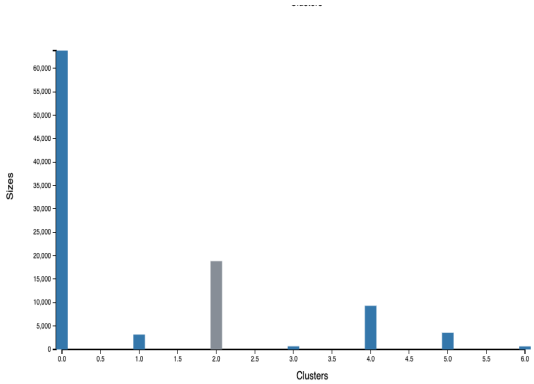

Fig. 14. Accident severity vs Light Conditions: Size of each cluster

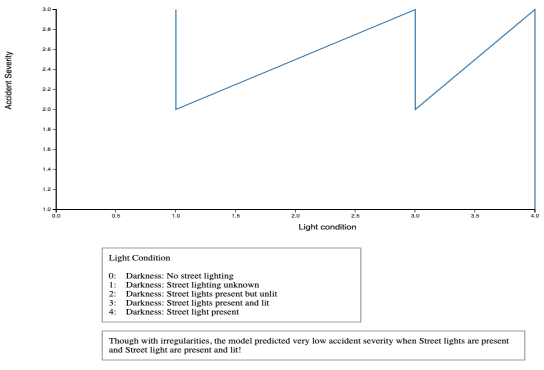

Fig. 15. Accident severity vs Light Conditions: Averages for each cluster

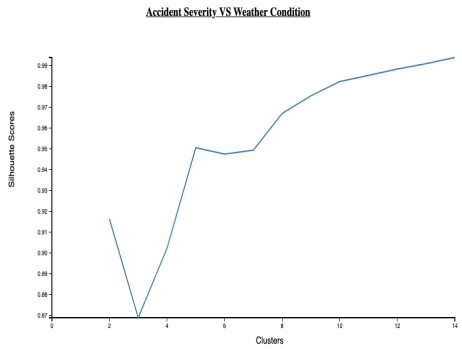

Fig. 16. Accident severity vs Weather conditions: Silhouette scores over clusters

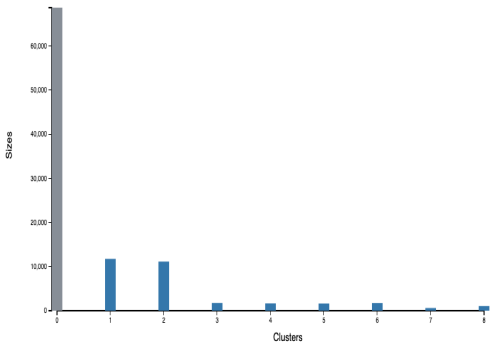

Fig. 17. Accident severity vs Weather Conditions: Size of each cluster

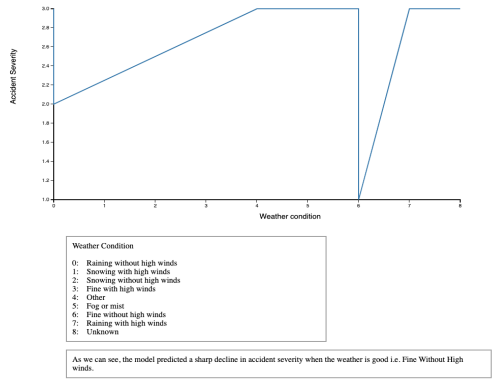

Fig. 18. Accident severity vs Weather Conditions: Averages for each cluster