Introduction

What is your study about?
The objective of this study is to identify dangerous sections of roads in Texas by performing a clustering analysis using the locations of all crashes in Texas. The number of crashes in a specific section of road does not disclose much information on how dangerous the road is because busier roads will always have more crashes than less busy roads. It might be that many crashes on an interstate result in far less injuries and deaths than a few crashes that take place at a dangerous intersection that receives far less traffic volume than an interstate. As a result, dangerous sections of roads in this study are identified through two different methods, the number of fatalities per crash in a roadway section, and the number of injuries per crash in a roadway section. I am experimenting with ways to combine these two metrics but haven't put much thought into it yet.

What motivates this study?
If dangerous sections of roads are identified through this crash data analysis then the information can be shared with civil engineering firms who can present the information to local municipalities or with Texas Department of Transportation (TXDOT). The engineering firms can petition to reconstruct the roadways to improve the overall safety of the community.

Background

Provide technical background information on the Big Data solution you have studied
Data for this study was obtained from the Crash Record Information System Database (CRIS) Database. It is a requirement to report all motor vehicle crashes that result in injury, death or property damage to the authorities and to TXDOT. Each crash is then meticulously logged into CRIS. Currently, CRIS does not have a public API and all crashes must be requested through email and the data will be sent via CSV file chunks. CRIS has been actively collecting crash data since 2014.

Two clustering algorithms were utilized as part of the Big Data Solution for this project, K-Means, and Gaussian Mixture Model (GMM). GMM was chosen because it excels at clustering data with various sizes, shapes and densities, which is optimal when trying to cluster crashes which can form shapes and densities of various sizes. However, GMM is computationally intensive for a Big Data Solution and estimating the number of clusters can be difficult. K-means was chosen for its fast and efficient algorithm, however similar to GMM estimating the initial number of clusters for accurate clustering results can be difficult. Additionally, K-means struggles to cluster irregular shapes which could lead to poor clustering results in downtown sections or other high density areas where it might cluster parallel roads in the same groups. DBSCAN and Hierarchical clustering were explored on small subsets of the data using scikit-learn and pandas, however a big data solution was not implemented using either of these algorithms. There is a lot of potential to improve clustering methods using DBSCAN because it relies on density based metrics and can cluster irregular groups very well, however it was not used because there was not a built-in implementation in spark.ml.clustering.

Two equations were utilized to obtain an accurate metric on the danger of a roadway section:

$$Fatality\ Rate\ = \frac{Fatality\ Count\ of\ the\ Cluster}{Count\ of\ Crashes\ in\ the\ Cluster}$$

$$Injury\ Rate\ = \frac{Injury\ Count\ of\ the\ Cluster}{Count\ of\ Crashes\ in\ the\ Cluster}$$

Implementation

Describe the Big Data task which you have implemented for this study, If you used any dataset, provide dataset size (KB/MB/...) and description of features (already existed or extracted), **Provide the implementation details, If you have used any module along with built-in methods, you must describe what each built-in method does.**

Dataset

The dataset utilized is 1.3 GB and is stored as a CSV file in an s3 bucket on AWS. Each row in the dataset corresponds to a single crash. The crash can contain multiple vehicles and people. A few key features were utilized from the dataset:

1. Latitude (float): Latitude of crash
2. Longitude (float): Longitude of crash
3. Sus_Serious_Injry_Cnt (int): The number of serious injuries
4. Tot_Injry_Cnt (int): The total number of injuries in a crash
5. Death_Cnt (int): The total number of deaths in a crash

Methods

The data was first loaded into a spark batch py file called `src/create_regions.py` that broke the dataset into four small regions for simple development and testing on a local computer. The four small regions included crashes in Boerne, TX, Downtown Austin, TX, Sugarland, TX, and Downtown San Antonio, TX. Exploratory Data Analysis (EDA) was performed on the regions using Pandas. Maps were created plotting the crashes and preliminary results of Hierarchical Clustering and DBSCAN clustering algorithms using Scipy and Scikit-learn. The results of the EDA using Pandas on the four small regions can be found in notebooks/test_pandas_clustering.ipynb file. Classes and functions defined for the Pandas EDA can be found `src/pandas_clustering.py`.

A module containing Spark classes and functions was then created to perform the clustering analysis that would be capable of scaling to a large dataset. The spark module was then tested on the four small regions. This module can be found in `src/spark_clustering.py`. Results of the

Spark Clustering Analysis can be found in `notebooks/test_spark_clustering.ipynb`. The implementation steps follow this outline:

1. **Load CSV file as a spark.sql.dataframe**. A spark.sql.dataframe was used because this class object is compatible with spark.ml.clustering.kmeans and spark.ml.clustering.GaussianMixture machine learning algorithms.
2. **Clean spark.dataframe.** All crashes containing NaNs of the five key features outlined above were removed
3. **Implement spark.ml.clustering.KMeans and spark.ml.clustering.GaussianMixture.** KMeans and GaussianMixture were chosen because spark has built in methods already developed for spark.dataframes. Initially I wanted to perform a DBSCAN algorithm for this project but implementing DBSCAN in a distributed way without built in support from spark seemed really complicated. Clusters were chosen by dividing the number of total crashes by 250. This number was chosen because the four small regions indicated that there were roughly 250 crashes per intersection in the last 5 years of data. Therefore, determining the number of clusters that results in 250 crashes per cluster resulted in moderately distinct clusters per intersection. Distinct clusters for intersections is crucial when determining risk because specific intersections can oftentimes be much more dangerous than others even though they are in close proximity to each other.
4. **Plot results of clustering models in the four regions.** The results of the clustering models were plotted in order to evaluate the effectiveness of the clustering algorithms.
5. **Compute Fatality and Injury Metrics and plot results.** The fatality and injury metrics were calculated for each cluster and the final results were plotted on a map.
6. **Run EMR Cluster on the entire dataset.** The notebook with all classes and functions defined was then converted to a spark batch file and the computations were run on an EMR cluster. Results were saved as a CSV file in an s3 bucket.
7. **Plot results.** New plots were created in different regions in Texas other than the four small regions in a jupyter notebook to visualize the results of the analysis.

Results and Conclusion

Plots below show the results of the clustering model in different regions in the state. For the final draft I will show the regions that had the highest fatality rate and I will also have a table that ranks the fatality rate from highest to lowest of the top 10 clusters. **Figures 1 and 2** show DBSCAN and KMeans clustering results in Boerne, TX (~4,000 crashes). These are preliminary results using a small subset of the data.

The clusters created from the DBSCAN algorithm using hyperparameters of epsilon=0.003 and min_samples=100 resulted in clusters containing distinct intersections and roadway segments as in the **Figure 1**. The clusters from the KMeans algorithm in **Figure 2** resulted in clusters of semi-distinct intersections and roadway segments. A few roadways segments were combined during the KMeans algorithm.

DBSCAN is certainly preferable to KMeans for the traffic clustering task given the results of the clustering below. However the KMeans algorithm should be sufficient for determining high danger traffic areas, and it is much more computationally efficient when applied to big data than DBSCAN. Additionally there is already a built in method for using KMeans with spark dataframes.

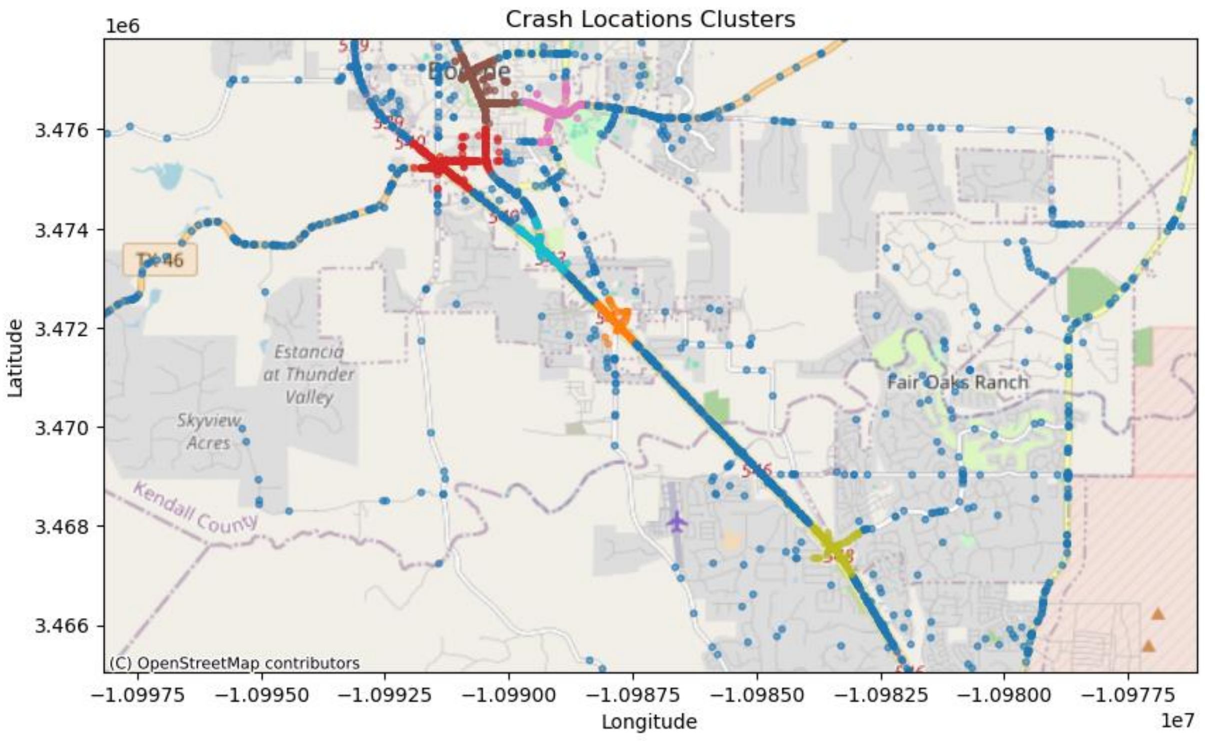

Figure 1: Boerne, TX Clusters produced by DBSCAN algorithm

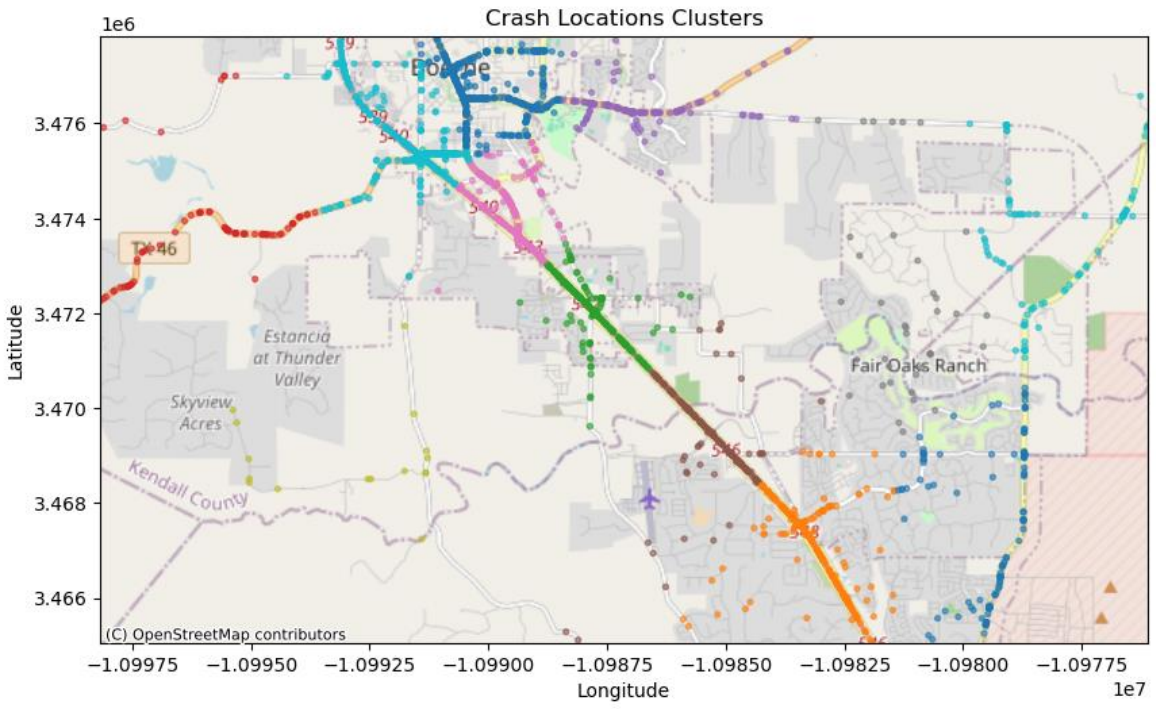

Figure 2: Boerne, TX Clusters produced by KMeans algorithm

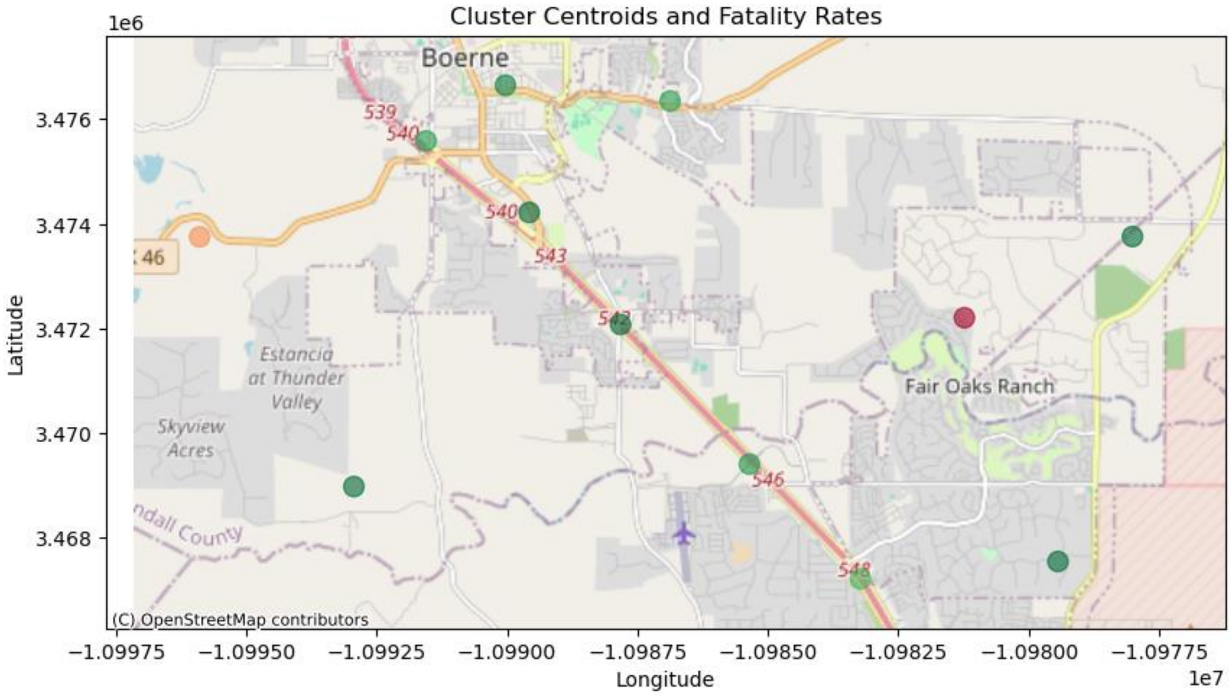

Figure 3: Centroids representing the Fatality Rate of each Cluster in Boerne, TX. (Clusters were determined from the KMeans algorithm)

The cluster centroids in **Figure 3** show the fatality rates of each cluster in the Boerne area. The cluster representing crashes in Fair Oaks Ranch has the highest fatality rate in the Boerne area at 0.053.

Future Work

Determining the size and number clusters is the most crucial step when performing this analysis and I think there could be lots of improvement in that area of the analysis. There are three different methods that I want to experiment with during future work in this study: Use DBSCAN to determine the cluster, use shapefiles and roadway segments determined by TXDOT to determine clusters, cluster using more dimensions to determine dangerous clusters.

1. **DBSCAN:** This algorithm would be advantageous for determining clusters in crash data because it is density based and can model irregular shapes very well. Crash clusters are almost always in tightly grouped irregular shapes either along a roadway segment or at an intersection which is a perfect use case for DBSCAN. I implemented DBSCAN using scipy and scikit-learn on a small scale using pandas dataframes with good clustering results after I tuned the hyperparameters. I did not initially use DBSCAN during the spark

implementation of the project because there was no built-in implementation that supported the DBSCAN algorithm using a spark.sql.dataframe.

2. **Merging on Roadway shapefile:** TXDOT has shapefiles mapping out all major roads in Texas and has broken the roads into segments based on various characteristics. It could be very advantageous to merge the road segments and the crash points based on their locations. An unsupervised ML algorithm would not be required in this scenario because the clusters would be the roadway segments determined by the TXDOT shapefiles. There are built-in methods for merging geographic points and regions in geopandas which would make this kind of merge very simple on a small scale. I do not know how easy this kind of merge would be on a larger scale.
3. **Clustering using more Dimensions:** Instead of using a fatality rate equation to determine dangerous clusters, it would be interesting to cluster using more dimensions than just latitude and longitude. The analysis could yield better results by applying GMM or DBSCAN on the data with latitude, longitude, fatality count and injury count. The results of this model alone might provide an accurate summary of the most dangerous intersections and roadways segments in Texas.

References

1. https://www.nhtsa.gov/research-data/crash-injury-research
2. https://highways.dot.gov/public-roads/septemberoctober-2016/big-data
3. Source Code for the project can be seen in the attached zip file.