Clustering and Labeling of a V2V Communication Dataset based on CAM and DENM Messages with Malicious Data Injection: Analysis and Implications

Alessandra Blasioli

[a]*University of Bologna MCs Computer Engineering*

Abstract

This study explores vehicular communication systems' potential for enhancing road safety and reducing traffic congestion. By leveraging two initial datasets representing Cooperative Awareness Messages (CAM) and Decentralized Environmental Notification Messages (DENM), the research focuses on data preprocessing, clustering, and the development of a system to identify potential malicious outliers.

Keywords: V2V, clustering, labeling, outliers

1. Introduction

Vehicular communication systems are computer networks where vehicles and roadside units (RSU) serve as communicating nodes, facilitating the exchange of information. This information can include safety warnings and traffic updates. These systems have the potential to significantly reduce accidents and alleviate traffic congestion.

Vehicle-to-vehicle (V2V) communication allows wireless information exchange about the speed and position of nearby vehicles, offering great promise in accident prevention, congestion reduction, and environmental improvement. However, the full benefits of this technology can only be realized when all vehicles can communicate with each other. That's why the National Highway Traffic Safety Administration (NHTSA) has collaborated with the automotive industry and academic institutions for over a decade to make V2V communication's life-saving potential a reality.

In this work, we were provided with two initial datasets containing data from a simulation of a real vehicular network. These datasets represent two different types of messages: Cooperative Awareness Messages (CAM) and Decentralized Environmental Notification Messages (DENM). CAM messages represent vehicle movements, while DENM messages convey information about events such as accidents or traffic incidents that vehicles encounter along their routes.

The objective of this work was to observe the data, perform data preprocessing and clustering, introduce noise into the provided dataset to simulate potential malicious reports, and ultimately develop a system capable of labeling the data based on clustering and identifying potential outliers, which we consider as 'malicious.'

2. Background

2.1. Cooperative Awareness Message (CAM)

The Cooperative Awareness Message (CAM), defined by the European Telecommunications Standards Institute (ETSI) in 2011, is a crucial component of cooperative Intelligent Transportation Systems (ITS) networks. It provides a basic awareness service by periodically sending status data to nearby nodes in the network. CAM acts as an application support facility, distributing messages containing information about presence, location, and fundamental status.

CAM messages are broadcasted to nearby ITS stations, allowing them to be aware of each other's presence, positions, movements, and characteristics. The recipient evaluates the message's information to make informed decisions. The ETSI standard offers a predefined set of information for CAM messages, which can be used for various ITS applications. It also allows for the definition of new data elements to cater to specific application needs.

The CAM message format, according to ETSI TS 102 637-2, consists of a header and a body. The header contains message-specific details like version, identifier, and generation time. The CAM message body contains essential information about the sending ITS station, including:

1. A unique identifier for the sending ITS station.
2. The type of ITS station (e.g., mobile, public authority, private).
3. The reference position, which includes latitude, longitude, elevation, and heading.
4. An optional set of CAM parameters, following the standard's recommendations based on the ITS station type.

In addition to specifying the CAM message format, the ETSI standard outlines processes for message handling. It provides guidance on timing requirements for generating CAM messages periodically. The frequency of message generation can be adjusted based on the specific ITS application's needs. The standard also describes rules for determining when a CAM message should be sent. However, it acknowledges that these rules are general, and architects implementing CAM facilities may have some flexibility in making certain decisions.

2.2. Decentralized Environmental Notification Message (DENM)

DENM, according to the European Telecommunications Standards Institute (ETSI) in 2011, serves as another type of application support facility, primarily providing a notification service regarding road status. While it was designed by ETSI to support active road safety applications, its utility can extend to any ITS application interested in acquiring information about road traffic conditions.

A DENM transmission is initiated by an ITS application that detects a relevant driving environment or traffic event. This application requests the DENM messaging facility to transmit DENM messages to notify others about the event. As per standard specifications, an event is characterized by various parameters:

- Event Type: An identifier associated with the type of detected event (e.g., vehicle breakdown, traffic jam).
- Event Position: Describes the event's location, which can be either a specific point or a geographical area.
- Event Detection Time: Represents when the event is expected to conclude.
- Destination Area: Indicates the geographical area over which the DENM message needs to be distributed among ITS stations.
- Transmission Frequency: Specifies how often DENM messages are issued by the same ITS station.

The ITS application must provide this information to the DENM facility, which is responsible for periodically transmitting DENM messages within the specified destination area and frequency. When the event's status changes, the ITS application notifies the DENM facility to update the information in the DENM message. Once the expiration time is reached, the DENM facility ceases to send DENM messages. Alternatively, events can be explicitly canceled by sending DENM messages to inform about the situation.

When an ITS station receives a DENM message, it assesses the information's relevance and determines the necessary actions (e.g., notifying the driver). It may also forward the message to neighboring ITS stations, whether vehicles or roadside units, to disseminate the information in the destination area specified by the originating ITS station. Unlike CAM messages, DENMs can be forwarded multiple hops away from the sending ITS station, covering longer distances. Roadside ITS stations, in particular, play a crucial role in collecting broadcasted information from vehicle ITS stations, processing it, and forwarding it to a central ITS station, benefiting traffic efficiency and management.

The ETSI TS 102 637-2 standard defines the structure of a DENM message, which consists of a header (similar to the CAM message) and a body containing event information. The event information is categorized into three sections:

- Management: Includes general event information, event source identification, event version, expiration time, sending frequency, event reliability, and an indicator of whether the event is false.
- Situation: Provides specific event details, including the event's cause, sub-cause, and severity.
- Location: Contains event location data, including hazard coordinates and extra trace location data.

In addition to message format and data elements, the ETSI standard offers guidance on DENM handling processes, including rules for forwarding DENM messages, detecting outdated messages, and managing situations where multiple ITS stations notify the same event. However, certain aspects, such as the communication path between central and roadside ITS stations for transmitting DENM information, remain undefined in the standard.

2.3. Clustering: The X-Means Algorithm

The K-Means algorithm is widely used for clustering and requires specifying the desired number of clusters in advance. This can be a challenging task as the correct number of clusters may not be known beforehand. This is where X-Means comes into play.
The objective of the X-Means algorithm is to automatically determine the optimal number of clusters in the data without requiring a predefined specification. It is based on the concept of recursion, introducing an evaluation of clustering quality during execution.

Initially, X-Means performs a standard version of the K-Means algorithm with an initial number of clusters. It then calculates a quality measure, such as the Sum of Squared Errors (SSE), to assess the goodness of the obtained clustering.

Subsequently, X-Means checks if dividing a cluster into two sub-clusters could improve the overall quality of clustering. It uses the Akaike Information Criterion (AIC) or another similarity criterion to determine if the division would lead to a significant improvement. If the division is deemed advantageous, the cluster is split into two sub-clusters using the K-Means algorithm.

This process of division and evaluation is iterated for all existing clusters, allowing for the potential creation of new sub-clusters and overall clustering improvement. However, if the division is not advantageous according to the criterion used, the cluster remains intact.

The X-Means algorithm terminates when it is no longer possible to further divide the clusters or when the division does not yield a significant improvement compared to the added complexity. At the end, the final clustering with the automatically determined optimal number of clusters is returned.

The use of the X-Means algorithm has the advantage of avoiding the need to specify the number of clusters in advance, allowing for automatic determination. However, it is important to note that the X-Means algorithm may require more time compared to traditional K-Means, as it involves iterating through multiple divisions and evaluations.
We opted for the utilization of X-Means as we believe it is well-suited for our specific objectives. The X-Means algorithm is part of ***pyclustering***, an open-source library for clustering, classification, and data analysis in Python.

2.4. DBSCAN

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm is a clustering algorithm that groups points based on the data density in space. It operates by defining two primary parameters: ϵ (epsilon), which represents the maximum distance between two points to consider them part of the same cluster, and *minPoints*, the minimum number of points required to form a cluster.

The ϵ parameter controls the radius around each data point that is considered for cluster expansion. A larger ϵ value results in larger clusters, potentially merging smaller clusters into a single larger one. On the other hand, a smaller ϵ value may lead to the formation of more compact and isolated clusters.

The *minPoints* parameter influences the minimum number of data points required to form a cluster. Setting *minPoints* higher can lead to more stringent cluster definitions, ignoring smaller, less dense regions. Conversely, setting *minPoints* lower allows the algorithm to identify smaller clusters with fewer members.

One notable feature of DBSCAN is its ability to identify outliers or noise in the dataset, as points that do not meet the density criteria for any cluster are labeled as noise. This makes DBSCAN particularly robust in handling datasets with varying cluster shapes and sizes, as well as datasets containing irregularities or outliers.

3. Implementation

3.1. Pre-Processing

In the initial phase of our work, our focus was on preprocessing the data to prepare it for optimal utilization with the subsequent clustering algorithms. Both datasets share a similar structure, with semicolons separating fields within each row. We found it most convenient to manage these datasets using a pandas DataFrame.

By employing the DataFrame, we were able to handle the data effectively. Regarding the clustering process, we encountered the need to convert certain fields in the dataset into float data types. This was necessary because the chosen clustering algorithm, X-Means, explained in the previous section, requires float values as inputs.

Within the dataset messages, there were some fields represented in hexadecimal values. We addressed this by converting hexadecimal values into float format. Additionally, we encountered numerous fields with NaN (Not-a-Number) values, which we subsequently dropped from the dataset.

For fields that originally contained letters, we adopted an approach to convert them into binary format and then further into float values. This was done to retain the underlying meaning of the fields while ensuring compatibility with the clustering algorithm.

Through these preprocessing steps, we made it possible to analyze the entire datasets using the chosen clustering algorithm.

3.2. Dataset Analysis and Clustering based on Latitude and Longitude Parameters

In this section, our primary focus is the analysis of the CAM and DENM datasets using latitude and longitude parameters. The main objective is to identify messages originating from the same source, which show contradictory trajectories between the DENM and CAM datasets. By carefully comparing these trajectories, our goal is to detect any discrepancies that may indicate the presence of malicious messages.
In general, when a vehicle follows a particular trajectory as indicated by CAM messages, any DENM message it sends should originate from a location that aligns with the CAM trajectory. If a report deviates significantly beyond a predetermined threshold,it will be considered as malicious. By carefully observing the provided data, we may gain a better understanding of the trajectories followed by the vehicles. These trajectories were recorded using CAM messages, as well as how these vehicles reported simulated events through DENM messages. To this end, we extracted from both datasets only the columns indicating the source and the position coordinates, namely latitude, longitude, and altitude, into two Pandas Dataframes. As evident, the two trajectories described by the messages exhibit a high degree of similarity. This is, of course, because the data is clean, and the reports do not contain any potential malicious information.

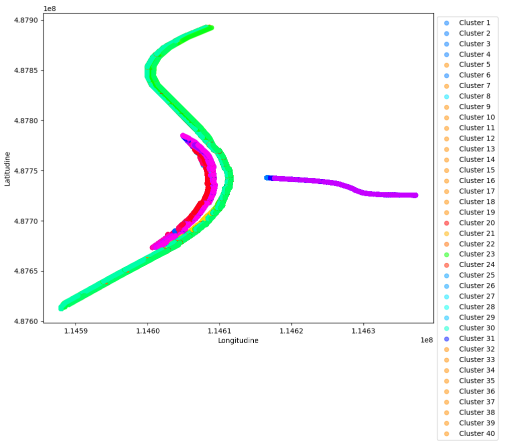

Figure 1: Clustering of the provided CAM dataset

As mentioned before, we employed the X-Means algorithm. Firstly, we calculated the “initial_centers” using the “kmeans_plusplus_initializer” method. This method takes the data and an initial number of centers and applies the k-means++

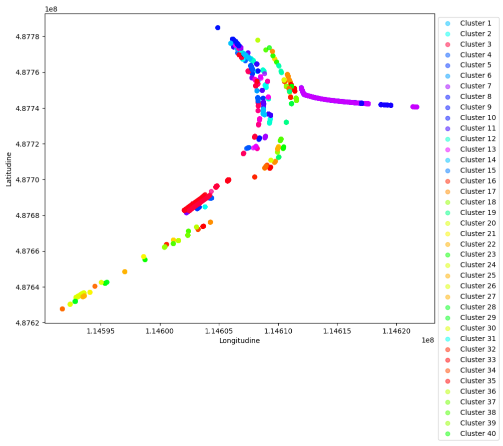

Figure 2: Clustering of the provided DENM dataset

algorithm to determine the initial set of centers, which enhances the efficiency and effectiveness of the k-means algorithm.

Finally, we called the "*xmeans*" method, which takes the data, initial centers, and a maximum number of centers (40, in our case) as input. This step initiates the X-Means algorithm and performs the clustering process on the provided data.

We followed the same process for both the CAM and DENM datasets.

3.3. Data Contamination in the DENM Dataset

The decision was made to introduce noise into this dataset because the primary objective is to simulate a dataset where potential malicious event reports, such as accidents, traffic disruptions, etc., are present. To accurately simulate the possible actions of attackers, it was initially decided to contaminate the dataset in the most realistic manner possible.

First, a function was implemented to determine new coordinates. We chose to vary the coordinates using a Gaussian distribution with a mean of 0 and a standard deviation of 1, multiplied by a factor of 100. This approach allows us to stay within a variation ranging from approximately 100 to 900 meters on the map. Since our dataset's coordinates are expressed in degrees, appropriate conversions were applied to enable the calculation of position changes.

Once the coordinates were calculated, a decision was made to contaminate data based on the number of sources.

Specifically, the choice was made to contaminate the data by selecting certain sources and contaminating only those, thus simulating malicious vehicles. Therefore, the decision was taken to contaminate primarily the sources with a *situation_eventType* value of 97, as that event possesses the most significant cluster. Additionally, of this type of source, only 20% are contaminated, resulting in a contamination of 8 sources.
Subsequently, by duplicating the initial DENM dataset and replacing the coordinate values with the newly calculated ones, we saved the contaminated dataset into a CSV file named *'dataset_denm_dirty.csv.'*

After the data contamination phase, we naturally applied the clustering algorithm to the contaminated data as well. Upon immediate observation of the generated graph, it is apparent that some points have been shifted from the previously observed trajectory. However, these points are not significantly distant from the event. Therefore, we may consider the contamination to be realistic and successful.

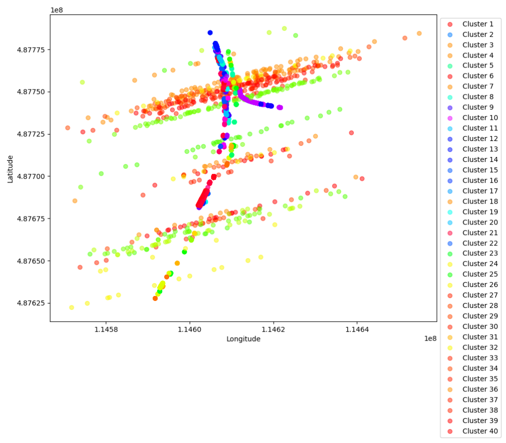

Figure 3: Clustering following data contamination

3.4. Event Type Analysis Based on Time and Space Parameters

A crucial parameter to observe within the dataset is the type of events being reported, specifically indicated by the 'situation_eventType' field. Although DENM messages can encompass various event types, our simulation focuses on three specific event types:

- CauseCodeType_dangerousEndOfQueue = 27;
- CauseCodeType_collisionRisk = 97;
- CauseCodeType_trafficCondition = 1.

To monitor the progression of these reports over time, we leverage two parameters provided by the DENM dataset:

- *'simulation_time,'* which represents the simulation time when the DENM message is received by an ITS-S, starting from 0.
- *'detection_time,'* indicating the time at which the event is detected by the originating ITS-S.

So, with these parameters, we proceeded to calculate the time difference between the simulation time and the detection time. It's worth noting that the simulation time required an initial conversion for this operation. Unlike the detection time, which starts from 0, we needed to adjust the value to express it in UTC, similar to the detection time. Once this conversion was applied, we were able to calculate the time differences between when a vehicle perceives an event and when it reports it.

Another critical parameter involves the spatial variation between the CAM and DENM messages emitted from the same source, occurring within a minimal time gap.

Understanding this spatial difference is vital as it helps ascertain if a vehicle sent a message from a similar physical distance compared to other DENMs related to the same event, with a similar CAM position. The spatial difference was computed by converting coordinates from degrees to radians, applying the Haversine formula to ascertain the angular distance on Earth's surface, and then multiplying the result by the Earth's radius to obtain the spatial difference.

By combining the time difference and spatial difference parameters, we constructed a graph illustrating the time it took to signal the event against the spatial difference. This graph facilitates the identification of outliers.

Before expalining how is possible to detect outliers from this kind of graph, me must focus on the results in the picture below (4).

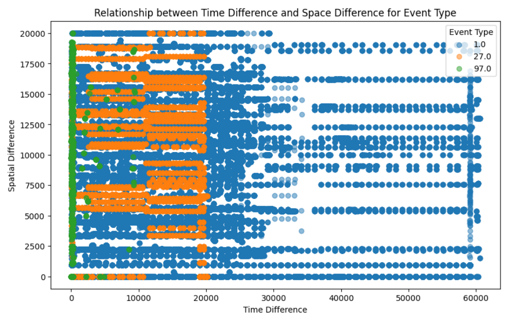

Figure 4: Relationship between Time Difference and Space Difference for Event Type

We observe distinct behaviors among the three different event types from the graph. To gain a clearer picture, let's promptly apply the Clustering algorithm to these data, grouping them by event type and the same source.

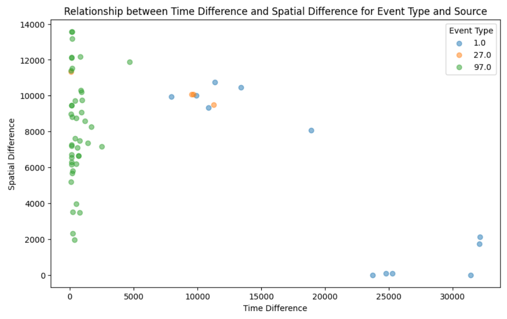

Figure 5: Relationship between Time Difference and Spatial Difference for Event Type and Source

Upon immediate observation from Figure 5, we notice that event type 97 represents the most significant cluster. Therefore, we proceed to work solely on the data associated with event type 97. This reasoning guided our decision in the preceding paragraph to exclusively manipulate data related to event type 97.

At this stage, focusing solely on the selected data with event type = 97, we reapply clustering exclusively to this subset. Particularly, we opt to primarily work with the mean of this data as we believe it better encapsulates the vehicles' behavior, offering a clearer representation of the scenario.

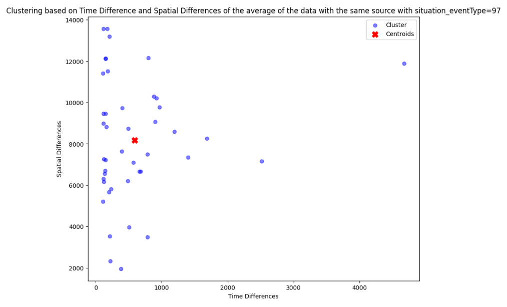

Figure 6: Clustering based on Time Difference and Spatial Differences of the average of the data with the same source with situation_eventType=97

As we see in Figure 6 it's evident that some messages took a very short time and space to be sent, likely indicating good messages. On the other hand, messages that took longer to be delivered since the CAM appeared are fewer, possibly raising suspicion of outliers in a real-life scenario. However, it's important to note that in this section, we are still working with clean data.

3.5. *Identifying Outliers Using Clean Data*

In our quest to identify outliers, we initiate from clean data and apply the DBSCAN algorithm. We experiment with varying epsilon values to find the optimal configuration. The choice of the best epsilon relies on selecting the one that identifies the fewest possible outliers, considering that the data we are working with is clean, and ideally, none of them should be considered an outlier.

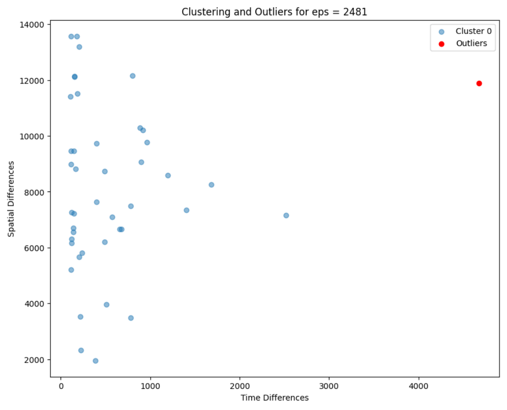

Figure 7: Clustering and Outliers for epsilon = 2481

As seen in Figure 7, the value chosen for epsilon is 2481. Now, our next step is to reapply this same epsilon value to the dirty data, as depicted in Figure 8. It's important to note that some points have been identified as outliers. Remember, we are consistently working with the mean of these points.

3.6. *Labeling*

As a final step, we add a column to our dirty dataset to label our data as "Malicious" or not. We set the attribute = True

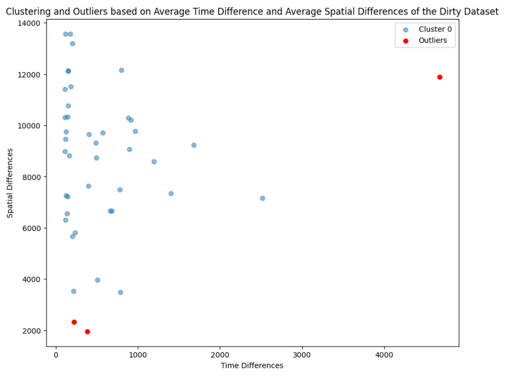

Figure 8: Clustering and Outliers for epsilon = 2481 with the dirty dataset

if the point, according to DBSCAN, is an outlier, and = False otherwise.

4. Experiments and Results

Finally, Figure 9 presents the trajectories plotted based on the DENM messages. All points identified as outliers by our algorithm are marked with ”x”. The results, as shown in Figure 9, may appear unsatisfactory; therefore, additional experiments were conducted to further investigate them.

Initially, the decision was made to assess the output results when selecting the best epsilon value as 1281. This value served as the starting point for the iteration in the search for the best epsilon, as it was identified as the first acceptable value. The outcomes of this experiment are illustrated in Figure 10. In this case, a significant improvement in results is observed, capturing a greater number of outliers, closer to the actual count of approximately 690 outliers.

An additional experiment was conducted to explore whether considering the time variable introduced any inaccuracies. Consequently, the clustering process was applied, and outliers were sought using DBSCAN with only spatial difference and source as parameters. With this approach, as evident from Figure 11, the detection significantly improves, capturing around 580 outliers.

5. Discussion

The objective of the project was to start with a dataset containing all CAM and DENM messages related to events occurring at a specific moment. From these data, the goal was to perform preprocessing and manipulate the data, followed by clustering operations. Through these clustering methods, the aim was to identify outliers successfully. Two algorithms, X-Means and DBSCAN, were involved in the clustering process. X-Means proved optimal for data analysis, while DBSCAN excelled in analyzing and detecting outliers.

The obtained results, while not perfect, are nonetheless satisfactory and provide insights for various analyses. As illustrated in Figure 9, challenges arise in achieving anticipated positive outcomes due to inherent limitations in this method. Potential

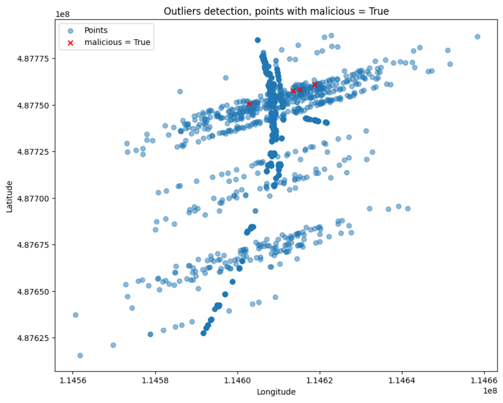

Figure 9: Outliers detection for epsilon = 2481 with the dirty dataset

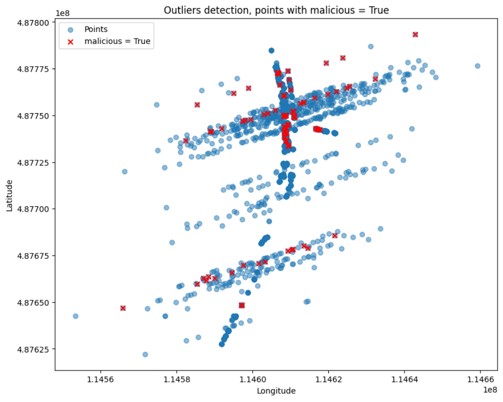

Figure 10: Outliers detection for epsilon = 1281 with the dirty dataset

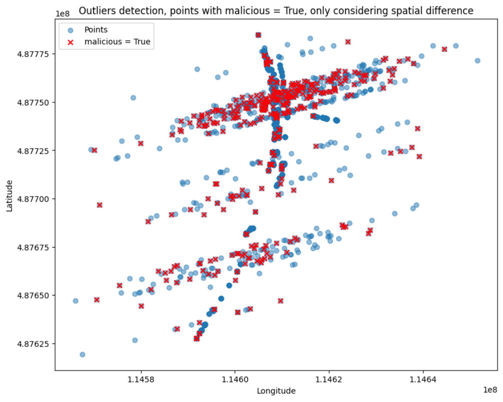

Figure 11: Outliers detection, points with malicious = True, only considering spatial difference

malicious actors within the system might strategically position themselves in both space and time to disseminate malevolent messages.

Considering the mean scenario, which we consider most indicative for an overarching description, slight shifts in our initial points occur. These shifts, though insufficiently substantial, fail to fully demonstrate the algorithm's effectiveness. However, appreciation for its functionality is evident when selecting the optimal epsilon, as depicted in Figure 10, identifying a greater number of outliers from clean data. Consequently, the final graph showcases heightened outlier detection, albeit at the expense of encountering false positives.

The algorithm's true efficacy becomes apparent in its ability to discern spatiotemporal differences between CAM and DENM messages. It can distinguish potential malicious messages based on those sent within similar time and space. In a real-world application, this tool would prove efficient, flagging messages transmitted from a distant space or time for a specific event, thus preventing potential threats.

In Figure 11, eliminating the time difference value results in even more satisfactory outcomes. Almost all outliers within the data are successfully identified.

6. Conclusions

This report introduces a model that, starting from clean data in a dataset of CAM and DENM messages, derives a method for outlier recognition even in contaminated datasets. The obtained results demonstrate that the system effectively identifies outliers; however, it lacks precision, particularly when dealing with points closely situated to trajectories, making outlier identification more challenging. Undoubtedly, the system serves as an excellent starting point for advancing this work. Future developments certainly involve the exploration of more advanced machine learning techniques for outlier detection. Additionally, considering a broader dataset with diverse event types for analysis could enhance the robustness of the model.

References

[1] Zhaojun Lu, Qian Wang, Gang Qu, and Zhenglin Liu; BARS: a Blockchain-based Anonymous Reputation System for Trust Management in VANETs

[2] Canhuang Dai, Xingyu Xiao, Yuzhen Ding, Liang Xiao, Yuliang Tang, Sheng Zhou; Learning Based Security for VANET with Blockchain

[6] Abid Ali, Muhammad Munwar Iqbal, Sohail Jabbar, Muhammad Nabeel Asghar,Umar Raza, Fadi Al-Turjman; VABLOCK: A blockchain-based secure communication in V2V network using icn network support technology

[4] anagiotis Papadimitratos, EPFL Levente Buttyan and Tamás Holczer, Budapest University of Technology and Economics, Elmar Schoch, Ulm University, Julien Freudiger and Maxim Raya, EPFL, Zhendong Ma and Frank Kargl, Ulm University, Antonio Kung, Trialog, Jean-Pierre Hubaux, EPFL; Secure Vehicular Communication Systems: Design and Architecture

[5] Xuanxia Yao, Xinlei Zhang, Huansheng Ning, Pengjian Li; Ad Hoc Networks: Using trust model to ensure reliable data acquisition in VANETs

[6] Hichem Sedjelmaci, Sidi Mohammed Senouci; An accurate and efficient collaborative intrusion detection framework to secure vehicular networks