# Senior Thesis:

# Anomaly Detection in Transportation Networks Using Machine Learning Techniques

Boston University

*Lab: Network Optimization and Control Laboratory*

*Advisor: Ioannis Paschalidis*


Athanasios Tsiligkaridis

May 5, 2017

*I would like to sincerely thank my brother, my parents, my research advisor, my academic advisor, and all my professors for their continued support and assistance throughout my undergraduate years.*

# Contents

# List of Figures

# List of Tables

# 1 Abstract

Most large cities throughout the world exhibit traffic congestion; Boston is a perfect example of this reality. Congestion can be characterized by both usual traffic jams (due to morning/evening commutes) and unusual jams (due to accidents, inclement road conditions, etc.); these unusual jams must be identified to prevent congestion from spreading to adjacent roads. The goal is to identify these unusual jams and report them in order to ultimately build smart and efficient cities.

This thesis involved the design of an anomaly detection system and its application to real traffic jam data provided by the City of Boston. For this endeavor, a dataset containing traffic jam points was used for creating anomaly-free baseline models for a set of time intervals defined over different seasons, days, and hours. All jams become parametrized into a 4D feature space; clustering methods are used to measure jam persistence in each time interval and discard the jams that do not appear often. With defined models, new traffic jams are compared and classified as either anomalous or non-anomalous.

# 2 Introduction

This thesis project is done in collaboration with the City of Boston in support of making Boston a smarter and more efficient city. There has been a large amount of recent work on smart cities on problems such as improvement of transportation networks [1], detection and classification of roadway obstacles [2], and emergency response systems [3]. We focus on the problem of anomalous jam identification for transporation network improvement.

The Google-owned traffic and navigation application, Waze, continuously collects data regarding traffic jams throughout a day. Waze provides this data to the City for research purposes. Boston's Department of Innovation and Technology receives and processes the data and it then passes information to the city transportation management group that has full control of transportation-related infrastructure such as traffic lights, sensors, and cameras throughout the Boston Metro area. The management group then analyzes this information and takes appropriate action if needed. The information of focus to us and to the City is traffic jam alerts.

Traffic jams can be classified into two categories: 1) typical jams, and 2) anomalous jams. Typical traffic jams are jams that one can expect to happen and that are not caused by a specific event. For example, a jam on the Mass Pike at 9:00AM would be classified as a typical jam because we expect a jam to exist at that time. These traffic jams are due to regular congestion and no city intervention can fix the congestion, only time can. On the other

hand, anomalous jams are unexpected jams that might happen at atypical locations and/or times. These jams could be caused by on-road accidents, inclement road conditions, construction at inappropriate times, etc. The anomaly detection system is intended to be used as shown in Figure 1. The system will receive jam data and it will output jam listings along with a metric that represents the probability that a jam is non-anomalous. This list can be used by the City of Boston and critical jams can be selected and studied further using the city's infrastructure; next, the cause of the jam can be identified and appropriate action can be taken. As an example, there might be a jam on a normally empty road at 9:30pm whose cause might be a car accident due to icy roads that has caused lane blockage; this jam is unexpected, and will be identified as such by the system. The jam will be relayed to the city and the city management group will use their infrastructure to investigate the cause of the strange jam (more specifically, they will use cameras to locate the car accident), and they will take action by contacting nearby police to quickly remove the cars and free up the blocked lane. The transportation management group wants to receive alerts regarding anomalous, not typical, traffic jams because only these jams can be remedied through action from the city. Alerts about normal jams are not important because the City will only waste time and resources to investigate a normal jam that cannot be fixed by its intervention. The thesis work aims to develop a traffic jam anomaly detection system that will provide alerts only about anomalous jams in order to preserve resources and focus solely on jams that

can be remedied with help from the City.

Data feature extraction and machine learning methods will be chosen and applied to the jam dataset for anomalous jam identification. Training data will be used to create nominal and anomaly-free traffic models and testing data will be used to test system detection performance.

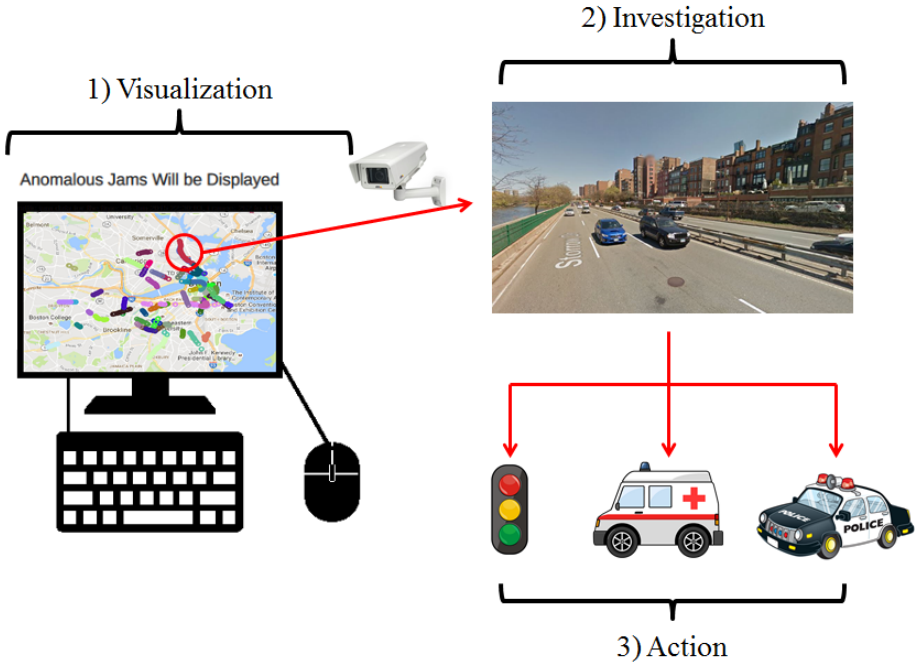



Figure 1: Anomaly Detection System Usage

# 3 Anomaly Detection System Design

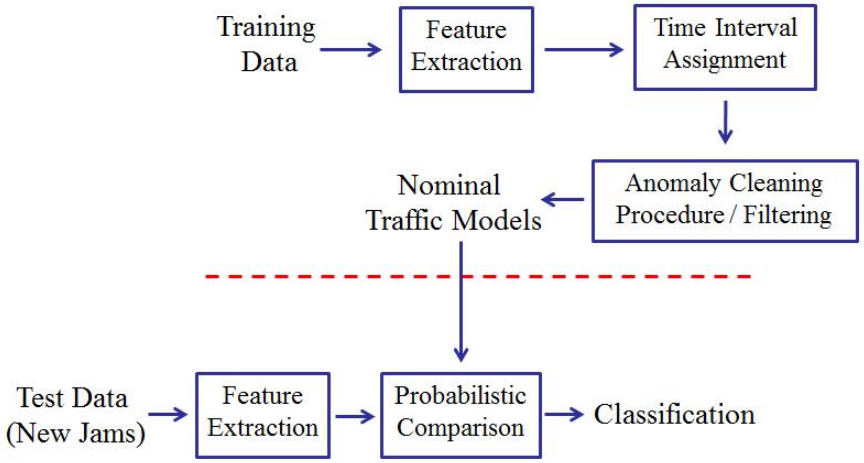
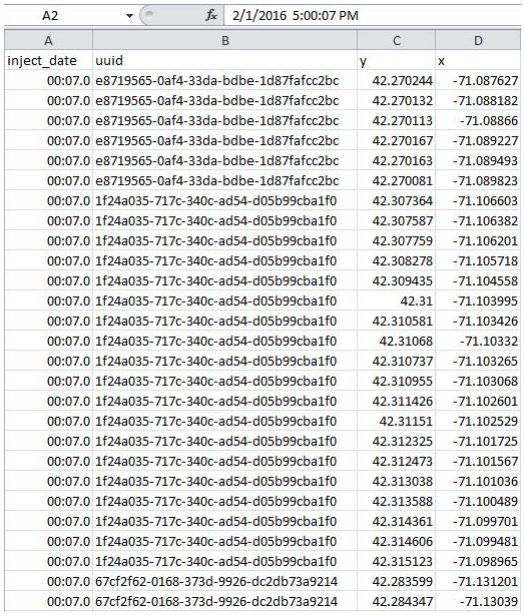



Figure 2: Anomaly Detection System Architecture

Figure 2 above represents the anomaly detection system block diagram. The system design has been designed relative to the structure of the data at hand. The massive jam dataset used in this endeavor contains jam points from 2014 to 2016 and it has the format as shown in Figure 3.

The data has 4 columns: 1) Time Stamp, 2) Jam ID, 3) Latitude, 4) Longitude. These elements can be used for traffic jam visualization at a specific time instance; this visualization is displayed in Figure 4. Overall, all the data at hand takes up at least 10GB of data, so working directly with this data would not be effective or efficient. It follows that a feature extraction must be performed to obtain data that will be more amenable to analysis. The feature extraction procedure is displayed in Figure 5. For every traffic jam, 4 parameters are extracted: 1) Mean Latitude ($\overline{lat}$), 2)



A2 2/1/2016 5:00:07 PM

| A | B | C | D |
|---|---|---|---|
| inject_date | uuid | y | x |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270244 | -71.087627 |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270132 | -71.088182 |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270113 | -71.08866 |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270167 | -71.089227 |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270163 | -71.089493 |
| 00:07.0 | e8719565-0af4-33da-bdbe-1d87fafcc2bc | 42.270081 | -71.089823 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.307364 | -71.106603 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.307587 | -71.106382 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.307759 | -71.106201 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.308278 | -71.105718 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.309435 | -71.104558 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.31 | -71.103995 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.310581 | -71.103426 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.31068 | -71.10332 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.310737 | -71.103265 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.310955 | -71.103068 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.311426 | -71.102601 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.31151 | -71.102529 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.312325 | -71.101725 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.312473 | -71.101567 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.313038 | -71.101036 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.313588 | -71.100489 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.314361 | -71.099701 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.314606 | -71.099481 |
| 00:07.0 | 1f24a035-717c-340c-ad54-d05b99cba1f0 | 42.315123 | -71.098965 |
| 00:07.0 | 67cf2f62-0168-373d-9926-dc2db73a9214 | 42.283599 | -71.131201 |
| 00:07.0 | 67cf2f62-0168-373d-9926-dc2db73a9214 | 42.284347 | -71.13039 |

Figure 3: Traffic Jam Data Format

Mean Longitude ($\overline{long}$), 3) Length [in km] ($l$), and 4) Directionality relative to a horizontal line ($\theta$). Thus, each jam gets represented by a feature vector of the form: $\overrightarrow{f} = \left[\overline{lat}, \overline{long}, l, \theta\right]$. With this, we form a simpler dataset that can be analyzed efficiently.

With the feature extraction, we implicitly assume that traffic jams can be approximated as lines; to verify this, we take all traffic jams, carry out a linear regression, and look at the distribution of $R^2$ values over all jams. Figure 6 displays the sorted distribution and we see that over 85% of jams have $R^2 > 0.7$; this verifies the validity of our assumption.

With the new and simplified dataset, we take all the data and form a training set; we will synthesize test data for use in the anomaly identification

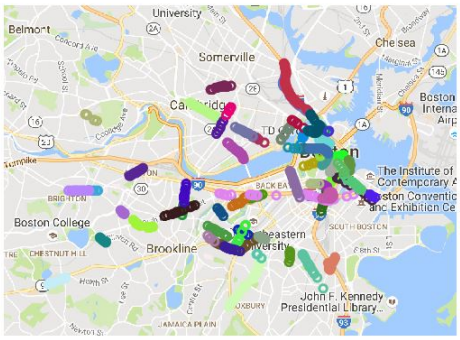



Figure 4: Traffic Jam Visualization @ 2:00PM, January 8th, 2015

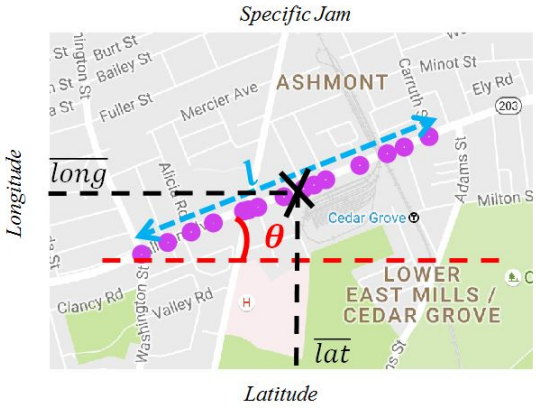



Figure 5: Jam Feature Extraction

stage. The simplified training dataset, again, contains 4 parameters and also jam occurence times. We want to take the training data and form baseline models for comparison with new data in the anomaly identification testing stage. To form baseline models, we must assume that traffic is constant throughout the information used in creating the model; but, since our data varies over years, seasons, days, and hours, there is a large amount of traffic variability so our assumption will definitely not hold. To remedy this, we

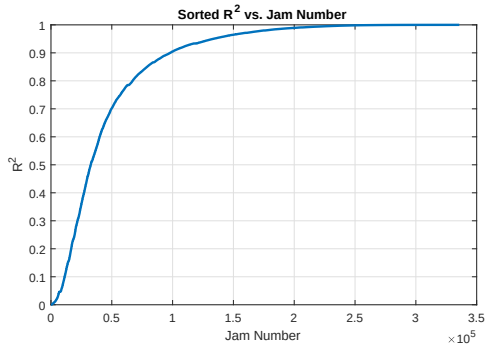



Figure 6: Distribution of Jam $R^2$ Values

must split the training data into time intervals over seasons, days of the week, and hours in a day to form partitions where traffic could safely be assumed to be constant. Figure 7 displays how the partitioning of the data is done. Basically, we split the data using each jam's time stamp over 4 seasons, over 7 days, and over 5 different hour intervals per day.

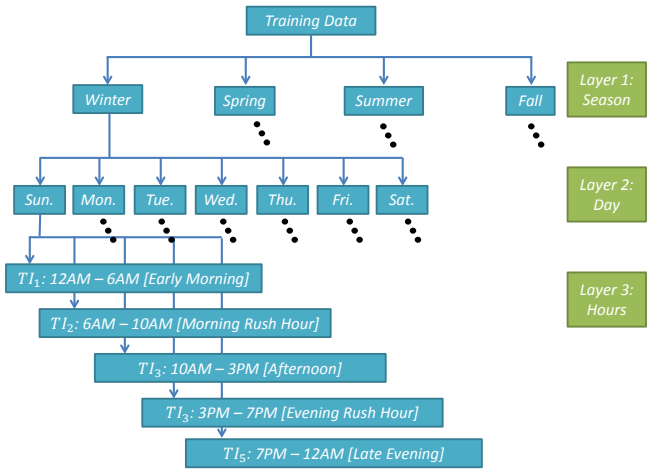



Figure 7: Time Interval Creation Scheme

With this, we form 140 time intervals, and a baseline traffic model needs

to be formed for each time interval. Each baseline model will consist of sets of traffic jams that will be referenced in testing to see whether or not a new jam resembles any jam in the baseline model. If an identical jam exists, the test point will be labeled as *non-anomalous* but if no identical jam exists, it will be labeled as *anomalous*. This comparison assumes that the baseline model is fully made up of non-anomalous jams; but this might not be the case since the jam dataset contains raw and unprocessed data. A cleaning stage must be applied to the data in each time interval in order to get rid of the jams that seldomly occur. Clustering methods will be used for this stage and the procedures and testing results are described in detail in Section 4.

After cleaning, baseline models can be formed and the testing stage can begin. In testing, raw jam data will be obtained, the feature extraction used in training will be carried out to obtain the feature vector $\overrightarrow{f}(test) = \left[\overline{lat}_{test}, \overline{long}_{test}, l_{test}, \theta_{test}\right]$, the time stamp of the test jam will be used to reference the appropriate time interval and thus baseline model, and a probabilistic comparison will be done to determine whether a jam is anomalous or not by using a probability measure. The methodology for the probabilistic comparison will be discussed in detail in Section 5.

# 4 Anomaly Cleaning Procedures

In this section, we endeavor to find an appropriate cleaning method in order to remove anomalous jams from each time interval; clustering methods will be used to achieve the task. We want to group together related jams in 4 dimensional space and locate the clusters with very few members. These clusters contain jams that seldomly show up and can thus be labeled as anomalous. These clusters with few members must be identified and removed to ultimately create baseline models of non-anomalous jams. In this work, we looked in depth at 2 different clustering methods: 1) DP-Means, and 2) DB SCAN. From our experimental analysis, we decide that clustering can happen more accurately by using cascaded clustering techniques. We look at the following 2 cascaded clustering methods: 1) DP Means + DP Means, and 2) DP Means + DB SCAN.

## 4.1 DP-Means

DP Means is a fast and scalable nonparametric extension of k-means where clusters can grow as a function of data. It was first introduced in [4] and this unsupervised learning method improves upon k-means by removing the necessity of knowing $K$, the number of desired clusters, and replacing it with a cluster penalty parameter, $\lambda$, that dictates the amount of clusters formed.

$$\arg\min_{\{l_c\}_{c=1}^{k}} \sum_{c=1}^{k} \sum_{x \in l_c} ||x - \mu_c||^2 + \lambda k \tag{1}$$

For a given set of data points $x_1, x_2, ..., x_n$, the DP Means objective function in (1) finds the best set of $k$ clusters $l_1, l_2, ..., l_k$ that appropriately groups together points and minimizes the objective function where $k$ is the number of created clusters, $n_c$ is the number of points in cluster $c$, and $\mu_c = (\sum_{x \in l_c} x)/n_c$ is the mean of cluster $c$.

The intuition of (1) is as follows: $k$ clusters are created, and for each cluster, we look at its constituent points and add up the Euclidean distance from each point to the cluster center to form a metric for each cluster. These metrics are then summed along with a term that penalizes $k$ to form a single value for each cluster set. We want to minimize this and thus find the optimal cluster set. To evaluate this metric, clusters need to be formed initially; this is done through the DP Means algorithm in 1.

Algorithm 1 successfully implements the nonparametric extension of k-means where the appropriate clusters are created based on a user-defined

**Algorithm 1**: DP Means

**Input:** $x_1, ..., x_n$ [Input Data] ; $\lambda$ [Penalty Parameter]
**Output:** $l_1, ..., l_k$ [Clusters] ; $k$ [# Of Clusters]
Initialize: $k = 1, l_1 = \{x_1, ..., x_n\}, \mu_1 = (\sum_{i=1}^{n} x_i)/n$
Initialize: Set cluster indicators: $z_i = 1, \ \forall i \in \{1, ..., n\}$
**repeat**
    For each point $x_i$:
        Compute $d_{ic} = ||x_i - \mu_c||^2$, for $c = 1, ..., k$
        If $\min_c d_{ic} > \lambda$, set $k = k + 1, z_i = k, \mu_k = x_i$
        Otherwise, set $z_i = \arg\min_c d_{ic}$
    Generate clusters $l_i, ..., l_k$ based on $z_i, ..., z_k : l_j = x_i | z_i = j$
    For each cluster $l_j$, compute: $\mu_j = (\sum_{x \in l_j} x)/n_j$, $\Sigma_j = \text{Cov}(x \in l_j)$
**until** convergence

penalty parameter, $\lambda$. The larger the $\lambda$, the fewer the amount of clusters that will be formed; the smaller the $\lambda$, the more plentiful the amount of clusters will become. For our problem, we would like accurate clusters but we do not want a large amount of them in order to make the testing portion of the project as computationally efficient as possible. The more clusters that exist, the more calculations that must be done in testing to determine whether or not points belong to any cluster. Figure 8 displays the results of DP Means on the simple case of 2 2D Gaussian clusters with various cluster penalty parameters. We see that the larger the $\lambda$, the fewer the created clusters and vice versa.

We see that the appropriate choice of $\lambda$ is very important; to accurately choose its best value, we must use cross validation.

Cross validation is an eminent and frequently used method of optimizing parameters to ensure best algorithm performance as presented in [5]. In

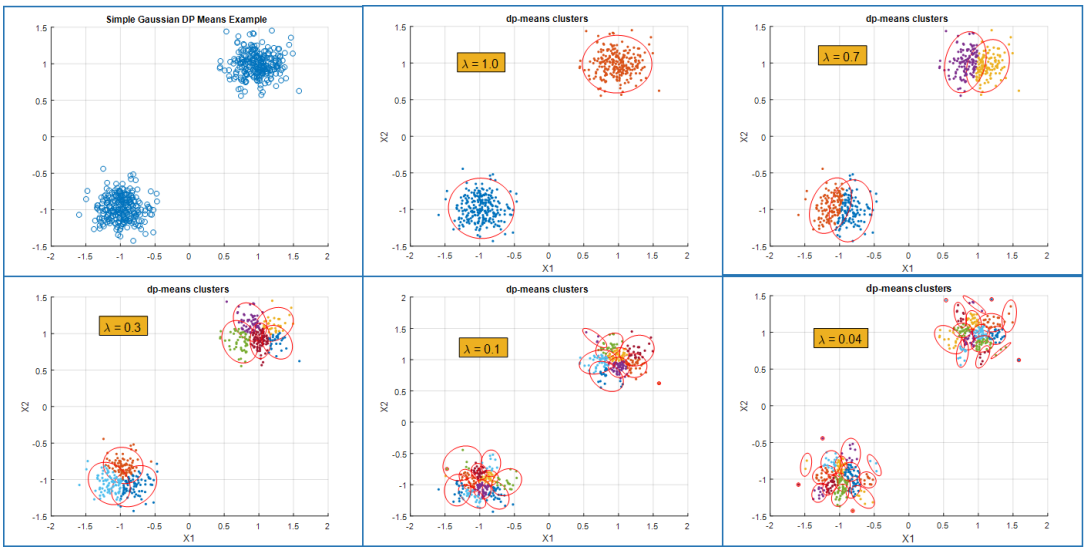



Figure 8: DP Means on Simple 2D Gaussian Clusters With Various Penalties

this case, we want to choose the best $\lambda$ by using $k$-fold Cross Validation; the procedure is displayed in 2 and a visual representation of the data partitioning is shown in Figure 9.

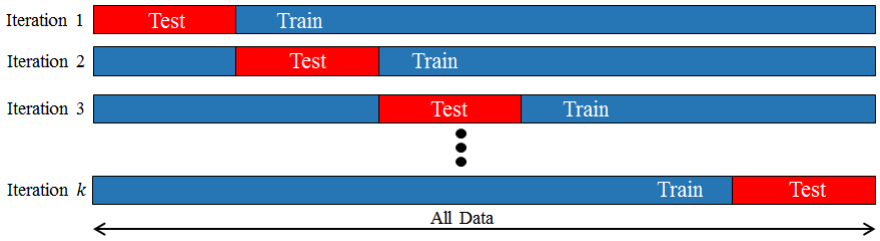



Figure 9: Visualization of $k$-fold Cross Validation

Procedure 1 allows us to locate the best penalty parameter in a given set; the larger the fold value used, the more computation time that is required. For our experiments, we use a fold value $k = 2$ since good performance was observed with this. Now, we look at some more complex examples of

*Procedure 1*: $k$-Fold Cross Validation for DP-Means

1: **Input:** $x_1, ..., x_n$ [Input Data] ; $\lambda_{set}$ [Set of $\lambda$'s] ; $k$ [Fold #]
2: **Output:** $\lambda_{best}$ [Best Lambda]
3: Partition the $n$-length dataset into $k$ $(n/k)$-length sized intervals $T_1, ..., T_k$
4: Loop through all penalty parameters; for a specific $\lambda_i \in \lambda_{set} = \lambda_1, ..., \lambda_M$:
5: Loop through the $k$ data blocks; for iteration $j$ where $j \in 1, ..., k$:
6: Set the testing set as $T_j$, training set as all remaining data (of length: $n - (n/k)$)
7: Run DP Means on the training set with $\lambda_i$ and obtain a set of clusters $l_1, ..., l_C$
8: For each test point, find closest cluster $l_{closest}$ and do the following:
9: Obtain weighted distance from test point to closest cluster: $d_{weighted,i} = \sqrt{(x_i - \mu_{closest})^T(\Sigma^{-1}_{closest})(x_i - \mu_{closest})}$ $\forall i \in 1, ..., n_{test}$ where $n_{test} = (n/k)$
10: For each cluster, add up the weighted distances from the points that have the specific cluster as their closest cluster.
11: Obtain the final evaluation metric for a specific $\lambda$ by taking the variance of all the per cluster summed up weighted distances; add an optional penalty $\beta K$ to the metric to penalize on the amount of clusters formed, $K$.
12: Choose the $\lambda_{best}$ that minimizes the penalized variance metric.

DP Means where the penalty parameter was obtained through 2-fold Cross Validation. As a reminder, our data has 4 features so we must cluster in 4-dimensional space. Unfortunately, we cannot visualize 4D information easily so the following examples are chosen to represent some features of the 4D data but in fewer dimensions so we can visualize the clustering results.

Figure 10 displays the results of successful DP Means clustering on 2 noisey lines; we look at these lines because jams are represented as lines along the 2 dimensions of our jam data, latitude and longitude. In addition, we look at clustering results on 6 slightly-noisey lines. Figure 11 shows the accurate clusters. Next, we want to show that clustering will work in higher dimensions, so we attempt to cluster 216 point masses in 3D in Figure 12. All masses are clustered correctly and 216 multi-colored clusters are formed. Finally, we test clustering performance on the latitude and longitude features of the jam data. Clusters are appropriately formed but the clusters in the Downtown Boston area are very dense and not fine enough as those of the outside suburban areas. This problem can be averted through multistage clustering which is discussed in 4.3.1 and 4.3.2.

Ultimately, we see that DP Means is a clustering method that could scale up well to our 4D data but the only bottleneck (which *can* be avoided) is the variation of cluster density that could occur as seen in Figure 13. We now transition to study another clustering method, DB SCAN, in 4.2.

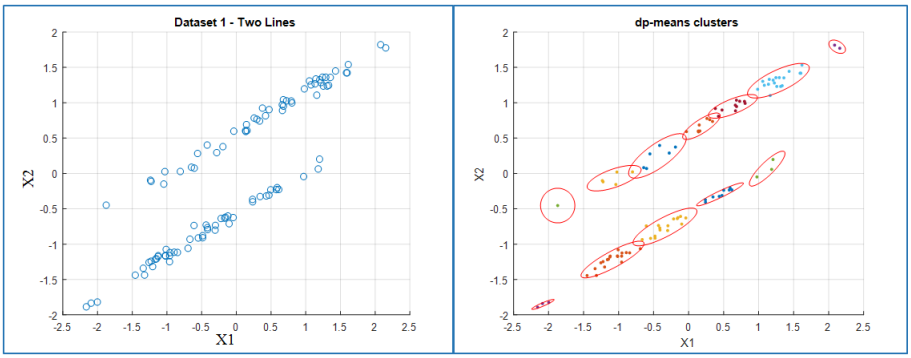



Figure 10: DP Means - Test on 2 Lines

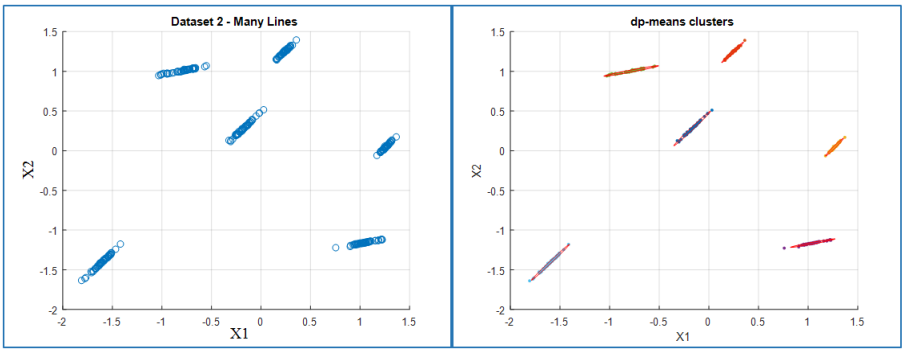



Figure 11: DP Means - Test on 6 Lines

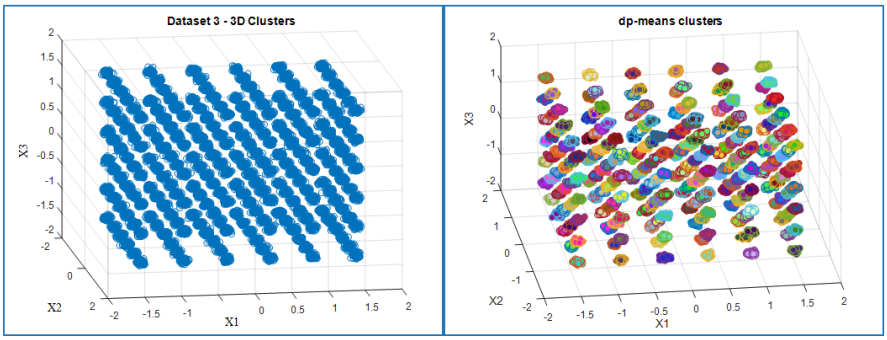



Figure 12: DP Means - Test on 3D Data

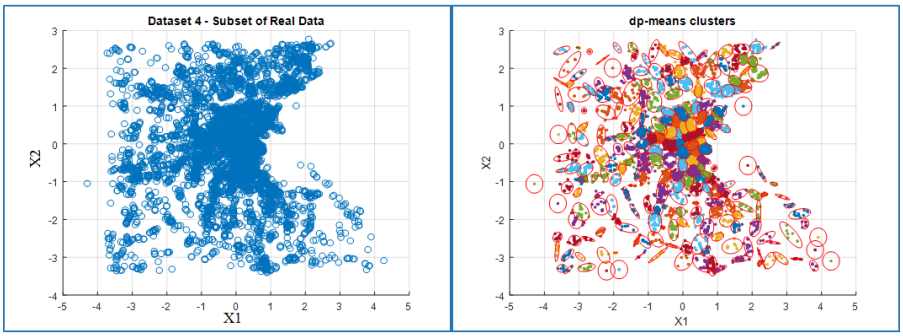



Figure 13: DP Means - Test on 2D Real Data

## 4.2 DB SCAN

DB SCAN is a very well-known density clustering method presented in [6] that groups together points that are closely packed together (points with many neighbors) and it marks as outliers points that lie alone in low density regions (whose nearest neighbors are too far away). This method is resistant to noise and it works well with clusters of various sizes and shapes [7]; but, just like with DP Means, this method does not work very well with data of varying density. This method requires two parameters: $\epsilon$ (distance threshold for connecting points) and $minPts$ (number of points required to form a dense region).

The conventional algorithm presented in [6] identifies each point as one of the following three types: 1) Outlier, 2) Core, and 3) Density Reachable; labeling occurs by forming an $\epsilon$-Neighborhood around each point and searching for neighboring points whose distances to the current pont are less than $\epsilon$. Outliers are points that have no nearest neighbors in their $\epsilon$-Neighborhood, core points have at least $minPts$ points in their $\epsilon$-Neighborhood, and density reachable points have at least one neighbor inside their $\epsilon$-Neighborhood. With these label identifications, the next step is to create clusters. Outliers form single member clusters while linked core and density reachable points form their own clusters. This method seems promising because the outliers that are identified are the anomalies that we must remove in creating baseline traffic models so we can just remove the outliers from the data and thus clean up each time interval for testing.

*Procedure 2*: DB SCAN

1: **Input:** $x_1, ..., x_n$ [Input Data] ; $\epsilon$ [Threshold] ; $minPts = 2$

2: **Output:** $l_1, ..., l_k$ [Clusters]

3: Loop through all points.

4: Create an $\epsilon$-Neighborhood for each point.

5: Connect each point with all other points in its $\epsilon$-neighborhood.

6: Create single member clusters for the points with no other points in their $\epsilon$-neighborhoods.

7: Create clusters using the inter-point links.

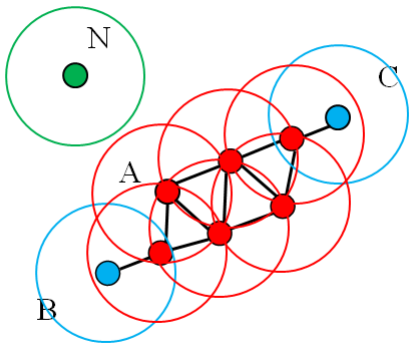



Figure 14: DB SCAN - Simple 2D Example

For our application, we solely care about forming clusters, not labeling them; thus, an intuitive procedure for applying DB SCAN is displayed in Procedure 2 and is used for our tests. A simple example of DB SCAN in 2D is also shown in Figure 14 in order to visualize cluster formations. Basically, points with no close neighbors form single member clusters but those with

close neighbors connect and form clusters.

To successfully implement DB SCAN, we must have knowledge of the 2 necessary parameters, $\epsilon$ and $minPts$. In our case, $minPts$ is not very important since we solely care about forming clusters (setting $minPts = 2$ results in good performance), not labeling points; so, it is imperative to find an appropriate value for $\epsilon$ and we do so using the $k$-fold cross validation method shown in Procedure 3 which endeavors to find the best $\epsilon$ that minimizes an aggregate cluster density metric.

*Procedure 3*: Cross Validation for DB SCAN

1: **Input:** $x_1, ..., x_n$ [Input Data] ; $\epsilon_{set}$ [Set of $\epsilon$'s]

2: **Output:** $\epsilon_{best}$ [Best $\epsilon$]

3: Loop through all $\epsilon$'s; for a specific $\epsilon_i \in \epsilon_{set} = \epsilon_1, ..., \epsilon_M$:

4: Run DB SCAN with $\epsilon_i$ and obtain a set of clusters $I_1, ..., I_C$:

5: Loop through all clusters:

6: For each cluster, do the following to obtain a density metric: Take all data in the cluster, form an adjacency matrix where each element is a euclidean distance, form a vector $\mathbf{v}$ containing all the distances in the upper triangular portion of the adjacency matrix, and define a cluster density metric as the *fraction of the sum of the values in* $\boldsymbol{v}$ *that are less than* $\epsilon$ *divided by the entire sum of* $\boldsymbol{v}$.

7: Obtain overall density metric for a specific $\epsilon$ by adding up the cluster densities over all clusters.

8: Choose the $\epsilon_{best}$ that minimizes the overall density metric.

Just like we did with DP Means, we test DB SCAN on a few representative examples that give us confidence that it will work appropriately with our 4D jam data. First, we apply the method on a 2-line dataset as shown in Figure 15; here, $minPts = 2$ and $\epsilon$ is obtained through cross validation. We see that points are clustered correctly and outliers are correctly identified. Next, Figure 16 shows nice clustering results for similar data that makes up 6 low-noise lines. Now, we look at clustering results of 216 3D masses in Figure 17; each mass was successfully identified as a cluster. Finally, in Figure 18, we test the clustering on the latitude and longitude features from a random time interval in our real data. Due to the density variation of the latitude and longitude features, the formed clusters are appropriately made outside of the Center/Downtown Boston area that is represented as one large cluster. For this section of the data, a smaller $\epsilon$ must be used; to test this, we take the entire center cluster and cluster it again as shown in Figure 19 using DB SCAN but with a smaller $\epsilon$. This cascaded clustering seems to be able to solve the density dilemmas we have had with both DP Means and DB SCAN; using these two methods, we form cascaded clustering schemes that we will use to form anomaly-free baseline models for use in testing.

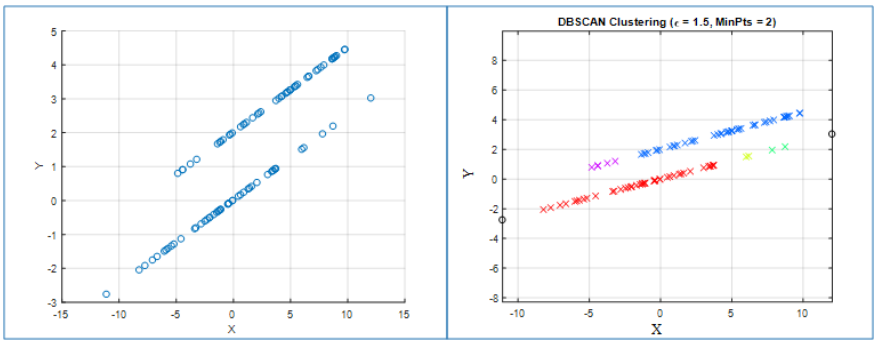



Figure 15: DB SCAN - 2 Lines

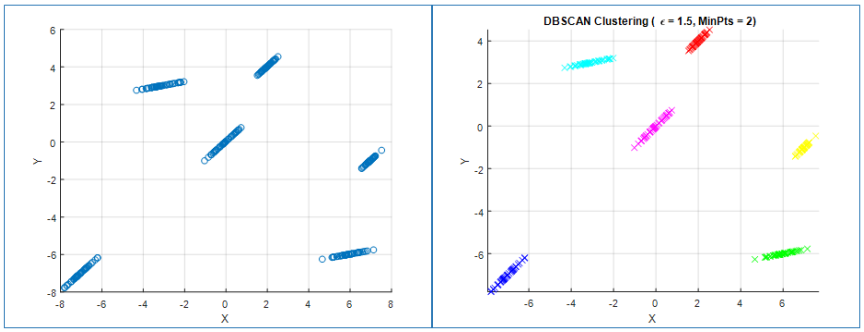



Figure 16: DB SCAN - 6 Lines

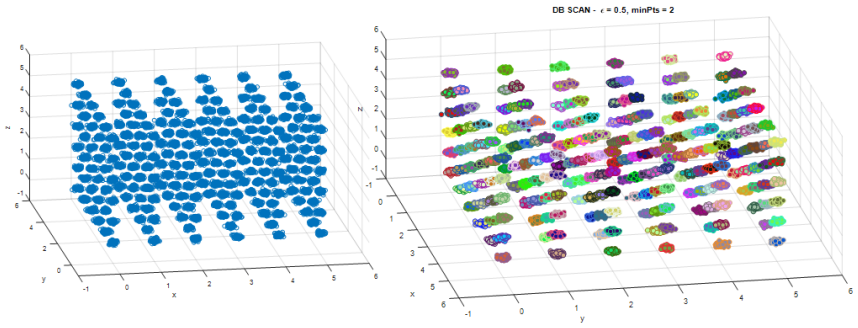



Figure 17: DB SCAN - 3D Data

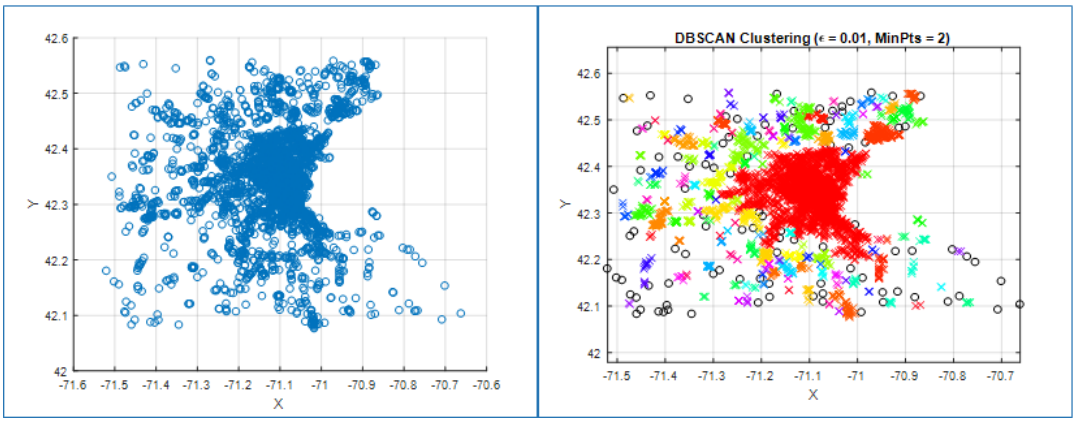



Figure 18: DB SCAN - Latitude and Longitude Data

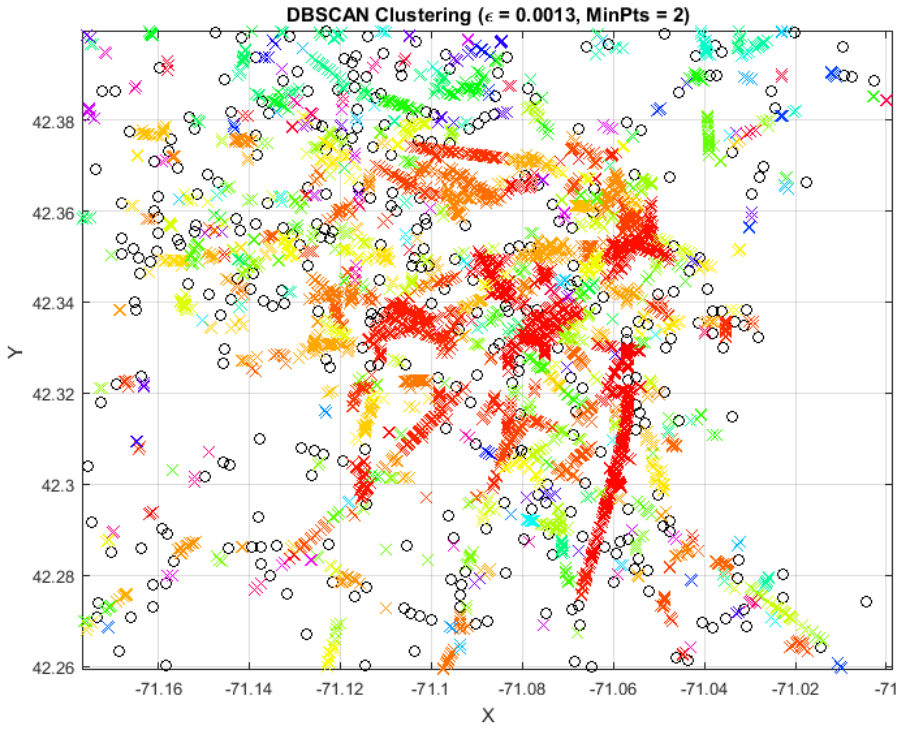



Figure 19: DB SCAN - Clustering On Dense 2D Data

## 4.3 Cascaded Clustering

Cascaded clustering is a means of circumventing the problem of varying data density that arises with single stage clustering. We ultimately want to form dense and accurate clusters that group together similar points and remove the clusters with very few members; the process for Cascaded Clustering is as follows: *We first take a dataset, apply one clustering method to it, get a set of clusters, and then apply another clustering method to the initial clusters that need further clustering.* We look at 2 cascaded methods: 1) DP Means + DP Means, 2) DP Means + DB SCAN and we test them out on simple examples to show their effectiveness in clustering data of varying density. For each time interval, these two methods will be applied and clusters with few members will be removed in the creation of baseline models. With this, we will have sets of clusters that we will use in 5 to determine whether or not a new jam is anomalous.

### 4.3.1 DP Means + DP Means

Cascading DP Means allows us to form dense and precise clusters; the clustering procedure is as follows: *Given a dataset, apply DP Means with a small penalty $\lambda$, and for the clusters that have a large amount of members, apply DP Means again with a $\lambda$ that is obtained from k-fold Cross Validation.* The procedure results are first shown in Figure 20 where we cluster a 2 line dataset. The resulting clusters from the first stage of the sequence are shown by the point colors while the even finer clusters from the second stage are displayed by the ellipses. Single member clusters are displayed without any ellipse around them. For our anomaly cleaning, these point clusters would be removed from the dataset. With this, we obtain finer and more accurate clusters for analysis. As another example, we look at a random time interval's latitude and longitude data, cluster once with DP Means, find a dense cluster in the Downtown Area, and then cluster it again to get finer and more accurate groupings. This is indeed the case as we see in Figure 21 where identical jams are grouped nicely. In addition to Cascaded DP Means, we look at DP Means + DB SCAN in 4.3.2.

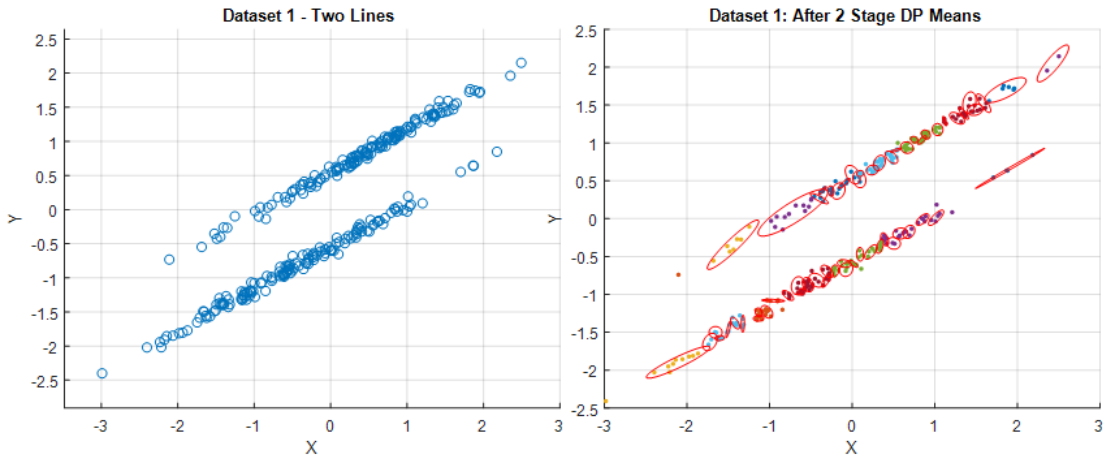



Figure 20: DP Means + DP Means - Clustering 2 Lines

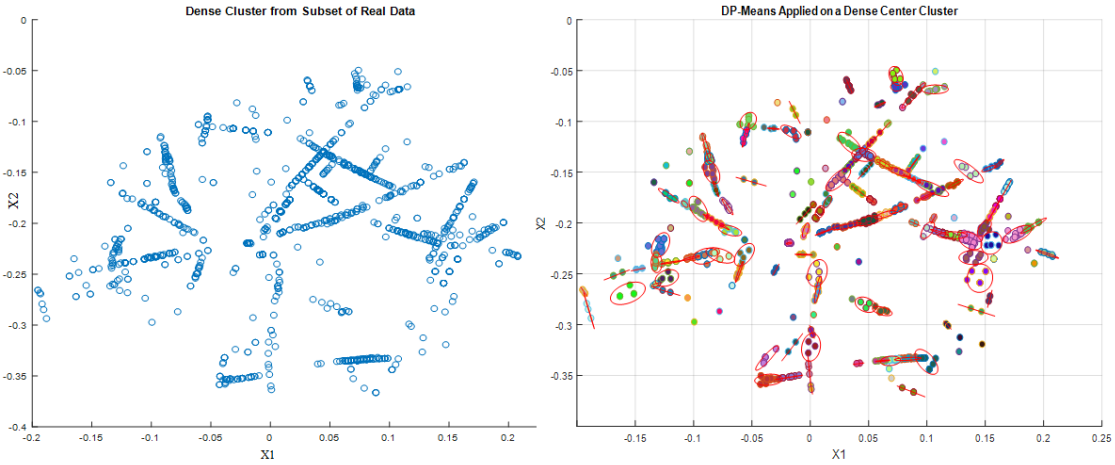



Figure 21: DP Means + DP Means - Clustering Downtown Boston Cluster

### 4.3.2 DP Means + DB SCAN

Here, we look at how clustering works with DP Means + DB SCAN. The procedure is as follows: *Apply DP Means to the entire dataset to form preliminary clusters, locate the clusters that require further clustering, and then apply DB SCAN to them where* $minPts = 2$ *and* $\epsilon$ *is obtained through k-fold cross validation.* As done with DP Means, we test this procedure on a 2-line dataset as shown in Figure 22; we see that the initial clusters from DP Means became more fine after the DB SCAN application. Next, we look at a dense cluster that was acquired from applying DP Means to latitude and longitude data and apply DB SCAN to it to obtain the clusters displayed in Figure 23. Similar jams are clustered correctly and those that do not have close neighbors are also correctly identified as outliers. For our anomaly cleaning, we can just remove these outliers directly from the dataset.

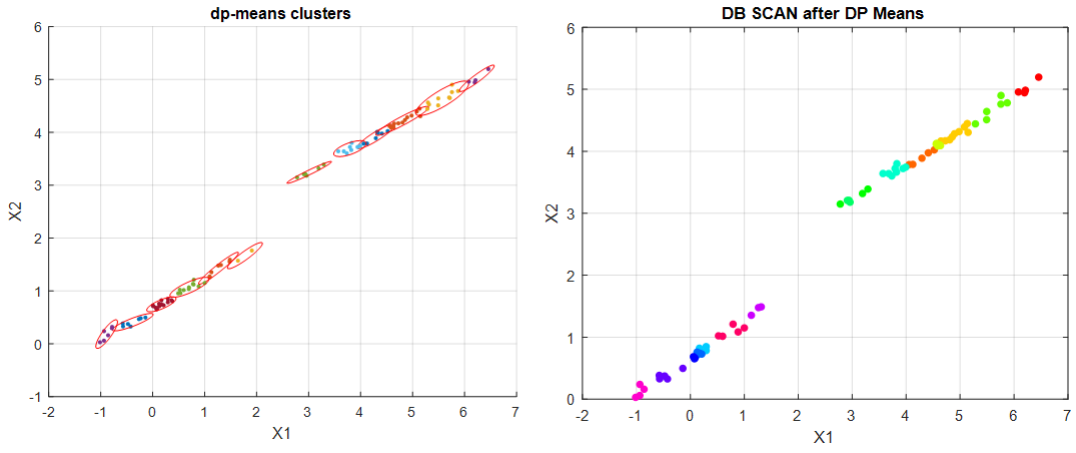



Figure 22: DP Means + DB SCAN - Clustering 2 Lines

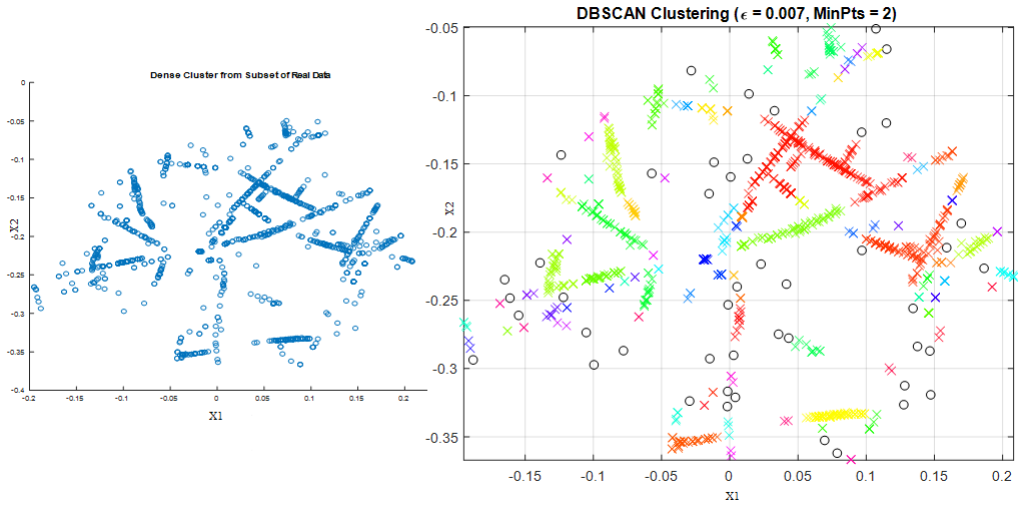



Figure 23: DP Means + DB SCAN - Clustering Downtown Boston Cluster

# 5 Anomaly Detection Testing Methodology

At this point of the work, we have been able to complete the training phase of the project where we formed anomaly-free baseline traffic models for different time intervals that could be used for testing whether or not a new jam is anomalous. Now, we want to find a means of comparing new jams to the data in a specific time interval; the way we do so is by forming a probability of association of a new jam to the closest jam cluster. As shown in Figure 2, to study a test point, we first parameterize the jam into 4D space, we reference the correct time interval based on *season, day,* and *hour* information, and then apply some comparison. The comparison process is depicted in Procedure 4; here, since we output a probability of cluster association, we present a soft decision where an anomaly can be defined by a range of small association values. Basically, small $p_{association}$ values show that specific jams cannot be associated to any cluster so they can be called anomalous and vice versa. Regarding the association metric, the $\alpha$ parameter is used to scale the squared Euclidean distance and its choice affects the resulting association value. More specifically, a large $\alpha$ will not give very large association values if a test point is very close to a cluster; on the other hand, a small $\alpha$ will not give very small association values if the test point is far away from a specific cluster. Thus, it is important to choose a scaling value that is not very large or small; an appropriate value can be experimentally found for a given dataset.

We start off by testing this Procedure on a simple 2 line dataset as shown in Figure 24 where we choose $\alpha = 1/8$. On the left, we have a test point that is very close to its closest cluster and the association value is 0.9360, a large value. On the right, the test point is farther away and the association value is smaller, 0.7813. From this, we see that the developed metric performs well. Next, we look at a more complicated example with a small subset of our real 4D feature data. The subset contains the points depicted in Table 1. We apply both cascaded clustering schemes (CC1 = DP Means + DP Means, CC2 = DP Means + DB SCAN) and the created clusters are the same and are grouped as shown in Columns 6 and 7 of Table 1. With these created clusters, we want to obtain association values of the test points in Table 2 (all of which are closest to Cluster 3) where feature vectors 2-5 are variations of feature vector 1 where only one feature value is changed; the rest of the jams vary multiple features, again, relative to feature vector 1. The changed features are shown in bold. Here, $\alpha = 3/16$ (experimentally chosen) for the testing metric and before any analysis, we normalize the test points to $z$-scores using the mean and standard deviations of the dataset in Table 1. The association values in Column 6 show that when you change a feature even slightly, the metric will correctly decrease to show that the adjusted jam has some characteristic that has not been seen in a specific time interval. When multiple features are changed, even smaller association values result, as we expect. With this, we see the effectiveness of our testing methodology and we can output a list of association values (which can be

represented as probabilities of belonging to the closest cluster in the time interval) that can be used by the City of Boston.

Now, we look at testing performance on synthetic jams during a random time interval for both cascaded clustering methods. The testing association metric is defined as: $\alpha = 5/8$; we look at the following time interval: *Season: Spring, Day: Friday, Hour: Evening Rush Hour* and the test points we use are displayed in Tables 3 and 4. More specifically, Table 3 displays test points and their probabilities of association to their nearest clusters using Cascaded DP Means; Table 4 shows the same elements using DP Means + DB SCAN.

The first test point occurs on Commonwealth Avenue, right on the BU Campus, and the chosen angle and length values represent the structure of the road and a common jam length. With CC1, we see that the association value is 0.8010, which says that this current test point belongs to the closest cluster and thus it should not be labeled as anomalous. CC2 results in a larger value, 0.8570, which shows even more that the point should not be anomalous. Next, we change the length of the same jam and we see that CC1 and CC2 give 0.7622 and 0.6725, respectively. Since the jam length has been increased to a large value, this jam can safely be labeled as anomalous since it represents heavy and unusual traffic; CC2 does the best in defining the point as such with a smaller association value. Now, we look at a jam on Massachusetts Avenue, a road that experiences traffic during rush hour. For some normal parameters, CC1 and CC2 correctly give large values of 0.8648 and 0.8707, respectively. After this, the angle and length parameters are

kept the same and the latitude and longitude locations are slightly shifted to a nearby road, Beacon Street, that does not experience traffic at the chosen spot. We want the resulting association metric to be small to show that the new jam is anomalous in comparison to the historical data; CC1 and CC2 give values of 0.7348 and 0.6699, respectively. We see that CC2 gives the more appropriate value. The fifth jam in located on Brookline Avenue, in front of the Beth Israel Deaconess Hospital, a place where traffic is common. CC1 and CC2 give associations of 0.8729 and 0.8616, respectively, and we see that both return correct values. The sixth jam occurs on South Huntington Avenue, a location close to the fifth jam but in a quiet neighborhood; we want the resulting probability value to be small. CC1 and CC2 give acceptable values of 0.7054 and 0.6979, respectively. Finally, the last 2 jams occur in the suburbs where not too many jams have been recorded. We want the association values for both these methods to be small; we get just that with both methods. More analysis of this kind can be done with different time intervals but the same performance trends will be observed as we have done more tests and ended up with the same conclusions.

With our testing, we conclude that both cascaded clustering methods perform very well but CC2 performs slightly better as seen with the preceding comparisons.

| 4D Subset of Real Data - 3 Clusters | | | | | | |
|---|---|---|---|---|---|---|
| True Labels | Longitude | Latitude | Angle ($\theta$) | Length (km) | CC1 Clusters | CC2 Clusters |
| 1 | $-71.0686$ | 42.3370 | 71 | .44 | 1 | 1 |
| 1 | $-71.0687$ | 42.3371 | 68 | .35 | 1 | 1 |
| 1 | $-71.0687$ | 42.3370 | 69 | .38 | 1 | 1 |
| 2 | $-71.0852$ | 42.3425 | $-13$ | .13 | 2 | 2 |
| 2 | $-71.0850$ | 42.3426 | $-11$ | .20 | 2 | 2 |
| 2 | $-71.0851$ | 42.3423 | $-12$ | .14 | 2 | 2 |
| 2 | $-71.0851$ | 42.3423 | $-12.4$ | .11 | 2 | 2 |
| 3 | $-71.0437$ | 42.3355 | $-55$ | .66 | 3 | 3 |
| 3 | $-71.0435$ | 42.3353 | $-50$ | .67 | 3 | 3 |
| 3 | $-71.0434$ | 42.3353 | $-52$ | .63 | 3 | 3 |
| 3 | $-71.0433$ | 42.3354 | $-48$ | .70 | 3 | 3 |

Table 1: Table of Data and Cluster Labels for 4D Subset

*Procedure 4*: Determining If A Jam Is Anomalous

1: **Input:** $I_1, ..., I_C$ [Time Interval Clusters]; $\mathbf{x}_{test}$ [4D Feature Vector]

2: **Output:** $p_{association}$ [Degree of Association to Closest Cluster]

3: Calculate the association metric for each cluster where $n_i$ = # of points in cluster $i \in 1, ..., C$, $\alpha =$ user-defined scaling parameter, and $d_{j,test} =$ euclidean distance from a cluster point to test point :

$$p_{association} = (\sum_{j=1}^{n_i} \exp(-\alpha d_{j,test}^2))/(n_i)$$

4: Locate the maximum association value and output it.

| Test Points for 4D Subset of Real Data | | | | | |
|---|---|---|---|---|---|
| Test Jam # | Longitude | Latitude | Angle ($\theta$) | Length (km) | $p_{association}$ |
| 1 | $-71.0430$ | 42.3340 | $-47$ | .72 | 0.9104 |
| 2 | **-71.0030** | 42.3340 | $-47$ | .72 | 0.6596 |
| 3 | $-71.0430$ | **42.3490** | $-47$ | .72 | 0.4773 |
| 4 | $-71.0430$ | 42.3340 | **35** | .72 | 0.7160 |
| 5 | $-71.0430$ | 42.3340 | $-47$ | **1.12** | 0.6895 |
| 6 | **-71.0030** | **42.3490** | $-47$ | .72 | 0.4144 |
| 7 | $-71.0430$ | 42.3340 | **35** | **1.12** | 0.6120 |
| 8 | **-71.0030** | **42.3490** | **12** | .72 | 0.3916 |
| 8 | **-71.0030** | **42.3490** | **12** | **1.12** | 0.3663 |

Table 2: Table of Test Jams for 4D Subset

| Test Results for Real 4D Data - DP Means + DP Means | | | | | |
|---|---|---|---|---|---|
| Road | Longitude | Latitude | Angle ($\theta$) | Length (km) | $p_{association}$ |
| Comm. Ave. | $-71.1022$ | 42.3494 | $-35$ | .6 | 0.8010 |
| Comm. Ave. | $-71.1022$ | 42.3494 | $-35$ | 1.5 | 0.7622 |
| Mass. Ave. | $-71.0873$ | 42.3464 | $-60$ | .3 | 0.8648 |
| Beacon St. | $-71.1715$ | 42.3331 | $-60$ | .3 | 0.7348 |
| Brookline Ave. | $-71.1072$ | 42.3389 | 45 | .48 | 0.8729 |
| S. Hunt. Ave. | $-71.1122$ | 42.3239 | $-79$ | .28 | 0.7054 |
| Trapelo Rd. | $-71.2056$ | 42.3947 | $-40$ | 1.1 | 0.6154 |
| Hartford St. | $-71.2524$ | 42.2127 | 20 | .34 | 0.4617 |

Table 3: Table of Test Jams for Real Data - *Spring, Friday, Evening Rush Hour* - DP Means + DP Means

| Test Results for Real 4D Data - DP Means + DB SCAN | | | | | |
|---|---|---|---|---|---|
| Road | Longitude | Latitude | Angle ($\theta$) | Length (km) | $p_{association}$ |
| Comm. Ave. | $-71.1022$ | 42.3494 | $-35$ | .6 | 0.8570 |
| Comm. Ave. | $-71.1022$ | 42.3494 | $-35$ | 1.5 | 0.6725 |
| Mass. Ave. | $-71.0873$ | 42.3464 | $-60$ | .3 | 0.8707 |
| Beacon St. | $-71.1715$ | 42.3331 | $-60$ | .3 | 0.6699 |
| Brookline Ave. | $-71.1072$ | 42.3389 | 45 | .48 | 0.8616 |
| S. Hunt. Ave. | $-71.1122$ | 42.3239 | $-79$ | .28 | 0.6979 |
| Trapelo Rd. | $-71.2056$ | 42.3947 | $-40$ | 1.1 | 0.5806 |
| Hartford St. | $-71.2524$ | 42.2127 | 20 | .34 | 0.4625 |

Table 4: Table of Test Jams for Real Data - *Spring, Friday, Evening Rush Hour* - DP Means + DB SCAN

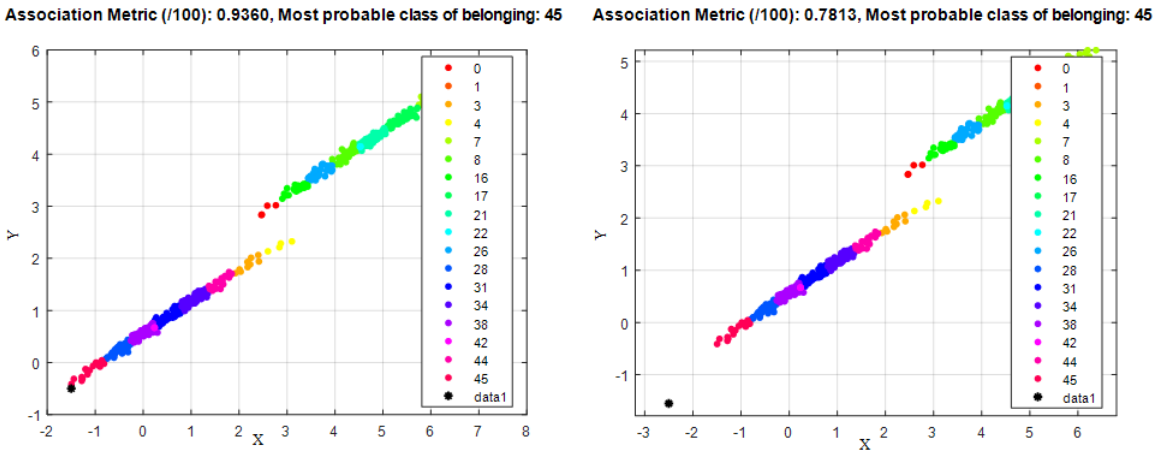



Figure 24: Testing: Simple 2 Line Example

# 6 Conclusion

In this work, we endeavored to develop a means of identifying anomalous traffic jams by using historical data; we achieved this by using machine learning techniques. We first began with latitude and longitude jam data and we extracted 4 features from each jam to create a less complex and more efficient dataset. Next, we partitioned the data into 140 time intervals grouped by season, day, and hour interval. With this, we studied different clustering methods and applied two cascaded clustering schemes to each time interval to group together similar points; we also cleaned some anomalous data from our time intervals by removing clusters with very few members. With the training phase complete, we then developed a means for testing whether or not a new jam is anomalous; we did so with a soft assignment method where we basically calculated, for a specific jam, a probability of it belonging to the closest cluster. Values close to 1 show that the jam is very much associated with its closest cluster so it is not anomalous while values close to 0 show that the jam is not associated with the closest cluster so it can be labeled as anomalous.

We successfully created a system that can interpret new data quickly and with a great amount of accuracy. In the future, more clustering methods can be studied for improved anomaly cleaning, more/different data can be used, and/or a different association metric can be used.

# 7 References


[1] J. Zhang, S. Pourazarm, C. G. Cassandras, and I. C. Paschalidis, "The price of anarchy in transportation networks by estimating user cost functions from actual traffic data," in *Proceedings of the 55th Conference on Decision and Control (CDC)*, (Las Vegas, NV, USA), December 2016.

[2] T. Brisimi, C. G. Cassandras, C. Osgood, I. C. Paschalidis, and Y. Zhang, "Sensing and Classifying Roadway Obstacles in Smart Cities: The Street Bump System," *IEEE Access*, vol. 4, pp. 1301–1312, 2016.

[3] A. Ataei and I. C. Paschalidis, "Quadrotor deployment for emergency response in smart cities: A robust mpc approach," in *Proceedings of the 54th Conference on Decision and Control (CDC)*, (Osaka, Japan), December 2015.

[4] B. Kulis and M. I. Jordan, "Revisiting k-means: New Algorithms via Bayesian Nonparametrics," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, (Scotland, Europe), June 2012.

[5] R. Gutierrez-Osuna, "Lecture 13: Validation." University Lecture, 2002.

[6] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge, Discovery, and Data Mining (KDD)*, (Portland, OR, USA), August 1996.

[7] J. Gao, "Clustering, lecture 4: Density-based methods." University Lecture, 2012.