BTECH 451: FINAL REPORT

INTERNATIONAL TELEMATICS LTD.

Nizamuddin Shaik
5695040
nsh726@aucklanduni.ac.nz

ACKNOWLEDGMENTS

I would first like to thank my academic supervisor Yun Sing Koh for all the help and motivation she provided me all throughout the year, without her guidance this project would have not been as successful as it was. I would also like to thank Dr. Manoharan for not only giving me with the opportunity to do this project but also for the valuable feedback he provided me. Finally, I would like to thank Greg Bacchus, for providing me the project and the sample data, which was very helpful.

1 TABLE OF CONTENTS

2 ABSTRACT

This report gives a detailed overview on my 4th year BTECH 451 project. For this project I am working with the International Telematics company and the project involves in an investigation and analysis of location based services. In section 3 of this report you will find a background introduction to the company, an overview about their current product they have for offer and the project they have proposed. In section 4, you will find an outline of the project goals and technologies that were used for the development of this project. Section 5 explains my initial research into the different data mining techniques from which I decided on one to implement for this project. Section 6 provides a basic overview of the standard version of the algorithm I have decided to use in my implementation, the *k-means* clustering algorithm. Sections 7 and 8 discus my research into similar areas of research and applications that involved the concepts used in this project. Then sections 9 and 10 give a detailed explanation of the techniques I have used in my implementation, and explained screenshots of my first initial application as well as showing its limitations. Section 11 explains the improvements made after analyzing and evaluating my initial application. Then sections 12 and 13 present the new and final application after the improvements have been made and also the final evaluation of this new application. Finally, the last 2 sections of this report provides some information on the future work of my project and a conclusive summary on my years' work.

3 INTRODUCTION

3.1 International Telematics

Telematics are solutions that allow users to integrate telecommunications and information processing systems to make the use of a vehicle much more efficient. These systems include GPS navigation, hands-free mobile calling, emergency warnings, and other automated driving assistances. *International Telematics* are a company who specialize in the design and manufacturing of such vehicle telematics. Their developed technologies were aimed for clients with off-site remote assets, specifically fleet managers of road vehicles which were mainly used for transportation. Their main goal was to provide these fleet managers with an environment allowing them to manage their fleet and assets with better efficiency which will save them money. The problem they have addressed is that, previously, fleet managers and owners were reliant on many different telematics solutions for their various vehicular types. *International Telematics* have introduced their *ibright* as a solution that provides all these solutions in a single platform with superior telematic features.

3.2 Current Solution

The *ibright* solution is an in-vehicle telematics computer which gives fleet managers and the drivers a powerful tool to collect various data and information about the vehicles and their journeys. The *ibright* hardware combines the use of wireless network technologies, accelerometer technologies, and machine-to-machine communications to gather critical fleet operational information and present it back to fleet controllers and drivers. This is all provided for through an interface called the *ibright* Enterprise. This user-friendly interface allows its users to control fleet operations from a central location and monitor assets in and out of specific locations. The main goal of the *ibright* solution is to dramatically improve performance and maximize productivity, reduce costs and optimize asset use to gain a clear competitive advantage

.

3.3 Project Proposal

The project *International telematics* have proposed is an expansion on this current technology they have for offer. As they have yet to have a system in place which lets them know how long a particular transportation vehicle will stop at a specific location, this project involves the development of an application which will predict this stop duration. This prediction of the stop duration will not only provide crucial timing information, it will also enable for planning of better routes for each vehicle's journey.

4 The Project

4.1 Project Goals

The goal of this project is to simply build an application that will predict the stop duration of particular trucks when they stop to make a pickup or a delivery at different sites. To do this, an analysis of previous recorded event information will need to be taken into consideration. This event information should encompass all the details of each individual truck journey. Such detailed information should include, details about the driver and the particular truck being driven, details about the location of the different pickup and delivery sites being visited, and most importantly, the currently recorded known duration of how long the driver and vehicle has stopped at these different pickup delivery sites.

4.2 Project Technologies

It has been advised by the company that their current project being developed is in *Node.js* and *MongoDB*. They have also suggested that the application being built by me should be compatible with such technologies so that they can easily use the components of it. So I decided to develop the whole application using *Node.js* and *MongoDB* itself.

With the fact that I was unfamiliar with both these technologies as they are relatively new to the industry, my first stage of this project was to do some initial research into them for some background understanding. I later found that these technologies are becoming very popular and their use is rapidly rising.

4.2.1 Node.js

Node.js is a platform built on Google Chrome's open source V8 JavaScript runtime engine. This means it enables for easy development of fast and scalable network applications. It uses an event-driven, non-blocking I/O model which makes it very lightweight and perfect for data-intensive real-time applications. So with the fact that this project involves analyzing numerous records of event information, the use of *Node.js* is very suitable for this application.

4.2.2 MongoDB

MongoDB is a NoSQL database which is also open source. It is a document database that provides high performance with the use of embedded read and write operations. This once again, with the integrated use of *Node.js,* makes *MongoDB* also a suitable technology for this application.

5 Data Mining Techniques

As the goals and requirements of this project invoked the use of data mining concepts, I knew initially it would involve an area in which I had no previous background knowledge. So the next stage of this project involved in researching such data mining concepts used for building prediction models. In my initial research I came across a couple data mining techniques that did such prediction models in machine learning applications. These techniques were *supervised* and *unsupervised learning*.

5.1 Supervised Learning

Supervised learning simply put, is the machine learning task of deducing a result from a labeled training dataset. A training dataset is usually a set of data used in discovering a possible relational prediction within the data found in the dataset. A labeled training dataset consists of training example pairs and in terms of machine learning, each pair consists of an input object and an output object (usually defined as a class label). A *supervised learning* method will train the labeled dataset using the pairs and then inferring a function that will match the input object with its respected class label. This can be formally defined as follows:

$$f_x = y \rightarrow \{(x_1, y_1), \dots (x_n, y_n)\}$$

where x_i and y_i are the input and output objects respectively, (x_n, y_n) are the training example pairs in the dataset where n is the total number of these pairs, and f is the target function which takes in an input object x and returns an output object (class label) y. From here it can be seen that for a new input object, x, that is not currently in the training dataset, a value for y can be obtained by applying the function f.

5.1.1 Classification

A *supervised learning method* that learns a target function which maps each attribute set to one of the predefined class labels is *classification*. In *classification* the target function is also known as a *classification model* and this model can be useful for two purposes, descriptive and predictive modeling. Predictive modeling is when the classification model is used to predict the class label of an unknown input value. For example, consider the following labeled dataset of vertebrates for which we have inferred a classification model:[7]

Name	**Blood**	**Skin**	**Legs**	**Aquatic**	**Hibernates**	***Class Label***
Human	Warm	Hair	Yes	No	No	*Mammal*
Python	Cold	Scales	No	No	Yes	*Reptile*
Salmon	Cold	Scales	No	Yes	No	*Fish*
Whale	Warm	Hair	No	Yes	No	*Mammal*
Frog	Cold	None	Yes	Semi	Yes	*Amphibian*
Komodo	Cold	Scales	Yes	No	No	*Reptile*
Bat	Warm	Hair	Yes	No	Yes	*Mammal*
Pigeon	Warm	Feathers	Yes	No	No	*Bird*
Shark	Cold	Scales	No	Yes	No	*Fish*
Penguin	Warm	Feathers	Yes	Semi	No	*Bird*
Salamander	Cold	None	Yes	Semi	Yes	*Amphibian*

Now suppose we are given the following new attribute set of another vertebrate which is not in the training dataset above:

Name	**Blood**	**Skin**	**Legs**	**Aquatic**	**Hibernates**	***Class Label***
Godzilla	Cold	Scales	Yes	Semi	Yes	*?*

We can now easily use the classification model we have built from the training dataset in Table 1 to determine the class label to which this new vertebrate belongs. This is a basic example of how *classification* works as a prediction technique.

Most *classification* techniques are best suitable for building a prediction model when working with datasets with categorical variables. A categorical variable is one that has more than one category, in which lies no specific ordering among them. An example of such variable is gender, which has two categories (male and female) however, there is no intrinsic ordering to these two categories where one category can be ranked higher than the other. Another factor influencing *classification* is that initially, a training dataset which contains records of which their class labels are known must be provided. The training dataset is crucial in building the classification model which is eventually applied to new attribute sets, containing records of unknown class labels.

5.2 Unsupervised Learning

In contrast to *supervised learning*, a machine learning task which infers a function from a labeled dataset, *unsupervised learning* is a task that aims to find unknown hidden data structure in a usually unlabeled data set. Thus, the training dataset will not consist of explicit training examples in which a target class label is to be deduced from an input attribute set. *Unsupervised learning* simply receives an input dataset $\{x_1, x_2, \dots x_n\}$ but does not obtain neither output object/class label, nor error and reward signals to evaluate an output object. Moreover, from this given input dataset, an *unsupervised* technique will aim to find patterns and relational links that lie within the input dataset objects x_i. Once these patterns are found, a representation of the inputs can be built for the use of predicting future inputs and decision making.

5.2.1 Cluster Analysis

An example of an *unsupervised learning* method is *cluster analysis*. This method involves dividing the input dataset into groups usually called clusters. Each input dataset object in a specific cluster is more similar to objects in the same cluster than to those in other clusters. Being an *unsupervised* technique means that the different data objects in the dataset are clustered based solely on the relational information found within the input dataset itself. For example, consider again the same vertebrate training dataset used previously, however, this time it is unlabeled without any class labels:

Name	**Blood**	**Skin**	**Legs**	**Aquatic**	**Hibernates**
Human	Warm	Hair	Yes	No	No
Python	Cold	Scales	No	No	Yes
Salmon	Cold	Scales	No	Yes	No
Whale	Warm	Hair	No	Yes	No
Frog	Cold	None	Yes	Semi	Yes
Komodo	Cold	Scales	Yes	No	No
Bat	Warm	Hair	Yes	No	Yes
Pigeon	Warm	Feathers	Yes	No	No
Shark	Cold	Scales	No	Yes	No
Penguin	Warm	Feathers	Yes	Semi	No
Salamander	Cold	None	Yes	Semi	Yes

Now the clustering method will group the data so all the similar data objects are clustered together as so:

Name	**Blood**	**Skin**	**Legs**	**Aquatic**	**Hibernates**
Cluster 1					
Human	Warm	Hair	Yes	No	No
Whale	Warm	Hair	No	Yes	No
Bat	Warm	Hair	Yes	No	Yes
Cluster 2					
Python	Cold	Scales	No	No	Yes
Salmon	Cold	Scales	No	Yes	No
Shark	Cold	Scales	No	Yes	No
Komodo	Cold	Scales	Yes	No	No
Cluster 3					
Pigeon	Warm	Feathers	Yes	No	No
Penguin	Warm	Feathers	Yes	Semi	No
Cluster 4					
Salamander	Cold	None	Yes	Semi	Yes
Frog	Cold	None	Yes	Semi	Yes

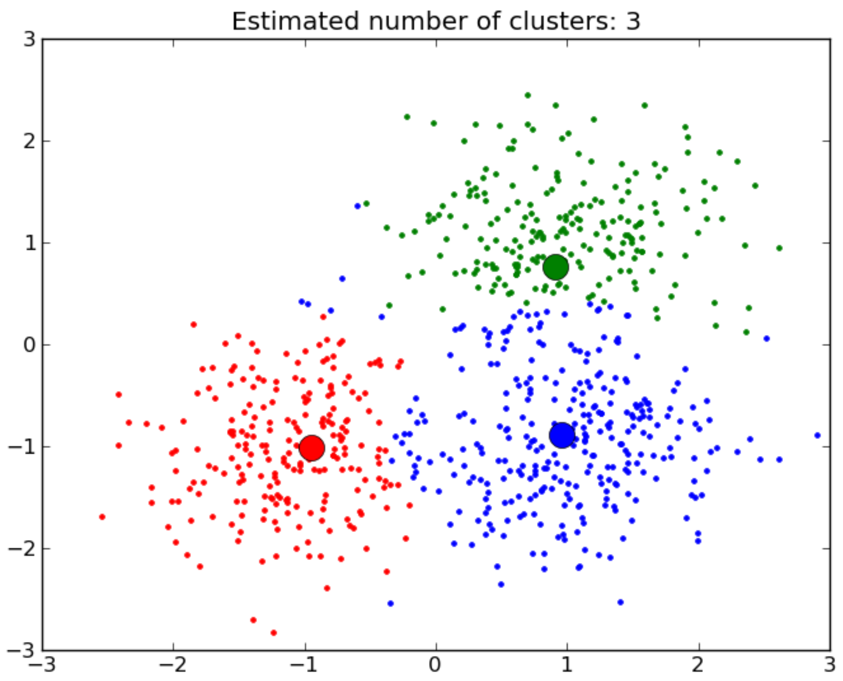
From here it can be seen that the initial input dataset has been grouped into four different clusters. What also can be seen is that within each of these clusters the vertebrate within them seem to have similar attributes with each other, specifically in terms of the "blood" and "skin" categories. Now by simply analyzing at the entries within the clusters a deduction can be made to state that each cluster represents a different vertebrate type. For instance, the clusters 1, 2, 3, and 4, can be used to represent mammals, reptiles, birds, and amphibians, respectfully. This is simple example of how *cluster analysis* can be used to predict desired outcomes with a method of grouping input dataset objects in terms of similar attributes.

5.3 Chosen Prediction Technique

The decision on which prediction technique will be best suited in the development of my application for predicting the stop duration of trucks at certain sites, depends heavily on the training dataset provided. However, I was not able to receive any sample data in the early stages of this project and had to wait until the request for some was granted. So, after looking back at the goals and requirements of this project, I led on to decide that an *unsupervised* technique such as *cluster analysis* was the best approach in moving forward. Another motivation for choosing this technique is that, since I had no initial training dataset to analyse, I was obviously unsure about the structure of the dataset I will be working with. And by just going by the project requirements, I believed there was a high possibility that there would be no class labels in the dataset. This meant there will be no training examples to enable for error/reward signalling to evaluate the output solutions. So the use of a *supervised technique* was withheld. This is where my decision for an *unsupervised* technique came into place. As a technique such as *cluster analysis*, tries to find structure in an unlabelled dataset, I will not be needing to worry about the structure of the given sample data. Also with the nature of clustering in *cluster analysis*, this technique is adaptable to changes in the future which is a very favourable aspect in the real world.

6 K-Means Clustering Algorithm

The *cluster analysis* method I have decided to use in my implementation is the well-known *K-means* clustering algorithm. This algorithm basically works by clustering the data objects within a dataset into a usually, user defined K number of total clusters. Each cluster is represented by a centroid point which is used as a reference to assign each of the data objects to the nearest cluster based on this centroid.



For example looking at the above 2-dimensonal graph, it can be seen with the number of selected clusters at 3, the dataset objects have been grouped into three different clusters. With each cluster having a centroid point, shown by the larger red, blue, and green dots, it can also be seen that most of the data objects around each centroid are assigned to their nearest corresponding centroid color. This shows here that the main objective of a *cluster analysis* method, where each data object is more similar to the data objects within its cluster than to the others outside its own cluster. The basic algorithm in how the *K-means* clustering algorithm performs this task, is explained below.

6.1 Basic Algorithm

Given a dataset $\{x_1, x_2, \dots x_n\}$, where x_i are the data objects where n is the last object and the number of desired clusters, K. The standard *K-means* algorithm is as follows:

1. Select K number of points from the dataset, to represent initial centroids $\{c_1, \dots c_K\}$.
2. Assign each dataset object x_i, to the nearest centroid point c_i, to form K clusters.
3. Re-compute each centroid point $\{c_1, \dots c_K\}$ of each cluster.
4. Repeat steps 2-3 until centroid points $\{c_1, \dots c_K\}$ do not change.

The details about how each step is performed is explained below.

6.1.1 Choosing Initial Centroids

After being given a dataset and a K number of desired clusters. The first step in the standard k*-means* algorithm is to select K number of objects within the dataset to represent to starting centroids of the clusters. This is the key fundamental step in the standard k-means algorithm as the initial centroids are very important. There are many ways in which these initial centroids can be selected but, most commonly, a random selection of dataset objects is used. However, the choosing of random initial centroids often leads to poor resulting clusters, as this process makes finding the natural clustering of the data very difficult. A technique that is usually used to avoid facing such problems, is to perform multiple runs of the algorithm, each with a different set of randomly chosen initial centroids and then the best resulting cluster obtained from these centroids set is chosen. The procedure of calculating the best cluster is explained later on in this report.

6.1.2 Assigning to Nearest Centroid

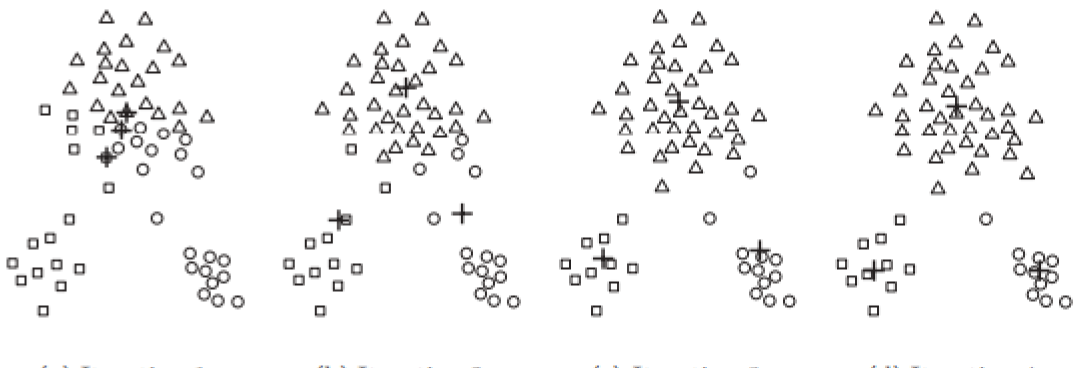
Once the initial centroids have been selected, the next step in the algorithm is to assign the data objects in the dataset to the nearest centroid. What this step means is that with each data object x_i the distance to each of the centroid points $\{c_1, \dots c_K\}$ is calculated. Then the data object x_i is assigned to the centroid c_i to which it calculated its smallest distance to. For this assignment to occur a measuring technique will need to be known that measures the distance between two data objects. The most commonly used technique is *Euclidean distance*. More on this technique is explained later on.

6.1.3 Re-computing Centroids

The next step in the k*-means* algorithm, after the data objects have been assigned to their respected clusters, is to re-compute the centroids of these clusters. Looking back at the goal of clustering, we need to ensure that the appropriate centroid is chosen so that the assigned data objects are closest to itself rather than to another centroid. So in obvious terms, the initial centroid of each cluster will need to be updated after all the data objects have been assigned. The purpose of this update is to make sure the centroid point is the mean of all the data objects within that centroid's cluster, so then the data objects are appropriately clustered within their respected clusters.

6.1.4 Repeating

The final step, rather the next action, is to repeat the data object assignment and centroid re-computation steps. This action is important because, when the data objects have been initially assigned to their nearest centroids, the next step is to re-compute these centroids and so, after this happens, the centroids are indeed updated but, the data objects remain assigned to their previous centroids. So the data objects, in a way, will need to be reassigned to the newly updated centroids. Then once this happens, there will obviously be a new mean of the cluster so, the centroids once again, have to be updated, hence the repeat of both these steps. These steps are repeated until the updating of the cluster centroids results in unchanged values. The following illustration shows why this repeat step is crucial in finding clusters in the data. From the initial random selection of centroids iteration, it can be seen in (a) that they do not identify all three groupings very effectively, and so the process is repeated. By iteration 4 in (d) it can be seen that the centroids are now a good representation of the groupings and so the *K-means* clustering algorithm now terminates.[7]


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

This is the basic procedure of how the standard K-*means* clustering algorithm performs its clustering task. Section 9 of this report will explain how I have decided to implement this procedure into my application.

7 Related Work

Before I got started with the implementation of my project, I first did some more research into related work to look into the similar research done in the context of my project. In this section you will find an overview of my findings revolving around different areas in my project.

7.1 Supervised vs Unsupervised

The first resource I came across in my research was a paper called *Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches*.[2] As the title suggests, this paper compares both *supervised* and *unsupervised* techniques that I have also mentioned in this report. The purpose of this paper is to investigate how these techniques can be used to classify movie reviews as either positive or negative. From a dataset of movie reviews, both *supervised* and *unsupervised* techniques were performed separately and the separate results were compared. The results of the *supervised* techniques showed that it was able to accurately recall 85.54% of the movie review classification, whereas the *unsupervised* technique had 77% classification accuracy.[2] These results do indeed suggest that the *supervised* technique is more accurate however, to obtain this accuracy the procedure took a lot more time in processing and training the model. In comparison, the *unsupervised* technique is only slightly less accurate and was stated to be much more efficient to use in real-time applications.[2] With this statement, a justification can be made on my decision in using an *unsupervised* technique in my implementation.

7.2 Clustering for Recommender Systems

The next couple of papers I came across in my research included the concepts of using clustering for *recommender systems*. The first paper, *A Two-stage Recommendation Algorithm Based on K-means Clustering in Mobile E-commerce*[9], looks into location-based personalized information services in the mobile e-commerce world. It provides an argument on why the accuracy of such services are poor if only the location of users is used as the primary standard for the recommendations. The paper suggests that a better recommendation result will be obtained with the use of a *k-means* clustering algorithm, as this can cluster the profiles of neighboring users. Instead of using just the location of a single user to target personalized information, the users profile can be clustered with the profiles of neighboring users to find a common preference in their desired information services. The methods used and explained in this paper seem to be very similar to what I should consider in my implementation. As I am using driver and vehicle details as well as details about the location of different stops, this is comparative with the user details clustering in a location as implemented in this paper. So ultimately, I can use this paper as a reference to refine and tune my implementation wherever necessary.

The second paper, *An Effective Recommendation Algorithm for Clustering-Based Recommender Systems*[5], shows the effectiveness of using clustering as a recommendation algorithm. The algorithm presented in this paper uses a refined neighbor selection method with the utilization of item attributes, which is again in direct correlation to what I purpose to do in my implementation. This paper can be used to show the effectiveness of finding valuable neighbors using clustering and manipulating attribute information of each item. So once again, I can use this paper as a reference point when implementing my algorithm in my application. This paper also explains how clustering techniques can often lead to bad prediction accuracy however, it also agrees with the fact that performance on a large-scale environment is dramatically better.

7.3 Time Prediction Models

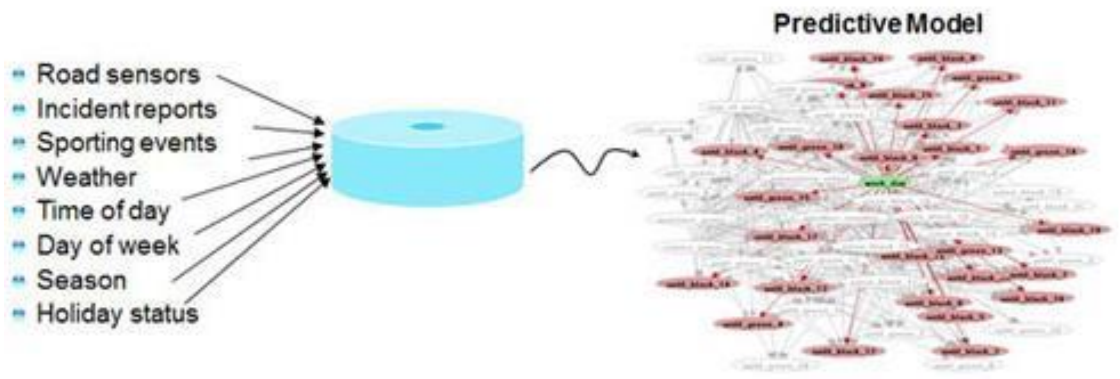
Next, in my research, I decided to look into different applications where the use of time prediction models have been implemented, so that I can familiarize myself with such applications before I begin my implementation. I came across a resource which describe real world applications that use the concepts of time prediction as a benefit. In the second chapter of *Clinical Prediction Models*[8], the value of prediction models for applications for medical practices and research purposes is explained. It explains how time prediction models can be useful for the planning of remaining life-time of patients with terminal disease. The prediction model will take into consideration the historic details of both similar patients and similar disease and/or similar symptoms of a different disease. This type of application can be used to show how time prediction can be effectively used in the public health sector. From this resource it can be seen that time prediction is a useful model in applications that are in different field area in my project.

8 Related Applications

Some final pieces of research I decided to do before my implementation was to look into other applications relating to the proposed one in my project. This section of the report provides a couple examples of current applications I came across in my research. They both worked in a similar field of predicting time and other factors with the use of historic data analysis.

8.1 Traffic Prediction

The first of such applications I came across was the *Predictive Analytics for Traffic* application that was introduced by the Microsoft Research team.[3] Their application which is in the midst of development addressed the issues of large amounts of traffic flow in major cities all around the globe. Their aim was to infer and predict this flow of traffic at specific locations at different times in the future. To do so, they took a related approach in building a machine learning tool which encompasses the use of both live streams of input information and analyzing large amounts of heterogeneous historical data.[3]



This data included various information about different factors and influences involved in the outcome of traffic flow. Such information included weather reports informing the weather conditions on specific traffic sites; major event information informing about near-by large crowd events such as concerts and sporting events; and also information about current accident reports on specific roads. These and many other pieces of data are used by the machine learning tool to build a predictive model which outputs information including a prediction on the duration of a traffic jam in a certain location and/or when a certain traffic location will began to clutter together. In this research they have developed a number of successful prototype applications and it also has been incorporated into Bing Maps which feature such machine learning and prediction features.[3]

8.2 Bus Arrival Time Prediction

The next example application I came across in my research was the *Bus Arrival Time Prediction Method for ITS Application*.[1] The motivation of this application arose from the implication that the stop duration of busses at traffic lights was the cause of the biggest errors in predicting bus arrival times. With this implication they introduced a prediction method which will incorporate various information about the different traffic lights in each of the bus routes to obtain a better prediction of the bus arrival time. To do so they first decided to build a predictive method to model the stop duration at different traffic lights. This involved in analyzing the different information about each traffic light stop. For example its location and environment, intersection type, state of the traffic lights, and its previously recorded length of stop duration. Their proposed solution was that with the prediction of the stop duration at signalized intersections incorporated with other bus arrival time prediction methods, a more accurate bus arrival time prediction is resulted. After field surveys and in-lab simulations, this proposed solution was tested and found to be very successful, averaging more than a 200% increase in the accuracy of the final prediction.[1]

From these two example applications of time predicting applications it can be quite evident that the methods and techniques they have used are very effective in this field. As my implementation is going to be very similar to what these examples have implemented, to make sure my implementation is also effective, using these examples as reference will be very beneficial. Also the successfulness of these applications can not only justify the various decisions made in my implementation process but also, direct them in the better path.

9 IMPLEMENTATION

After I had familiarized myself with a sound understanding of the different techniques used in prediction models and applications, I decided to begin my implementation. For my implementation of the *K-means* clustering algorithm, I first made an analysis on the sample data which I was eventually given. From this analysis, I decided on the method and the techniques that I planned to use in my implementation. A snippet of this sample data is given below.

Data Entry	VehicleID	GeofenceID	Category	Arrival	Departure	Duration
1	A91E0C9	983055C6	Load/Un.	1/6/13	1/6/13	689

				7:50	8:01	
2	D544532	798A02BE	Load/Un.	4/6/13 5:07	4/06/13 6:46	5952
3	545B1611	9E839BCD	Load/Un.	4/6/13 5:47	4/6/13 6:12	1490
4	D544532	8D1CF87C	Queue	4/6/13 6:46	4/6/13 7:00	853
5	1CE8212F	16683F4A	Service Centre	4/6/13 7:28	4/6/13 7:31	196
6	A91E0C9	8D1CF87C	Queue	4/6/13 8:02	4/6/13 8:30	1647
7	B7A70A11	8D1CF87C	Queue	4/6/13 8:38	4/6/13 11:59	12068

The first aspect of the sample data I noticed, was the absence of a driver attribute for any of the data entries. As mentioned in the goals of this project, the provided training data will need to have details about the vehicles as well as the drivers that drive them. However, due to possible reasons of confidentiality, I believe the driver details have been purposefully withheld. So my implementation method will need to be factored without the use of driver details. What follows in this report, is a breakdown of this method and an explanation of the techniques I have used in my implementation.

9.1 Method

9.1.1 Dataset Partitioning

After my analysis of the given sample data, I found that each data entry in the dataset has a *GeofenceID* attribute, as shown above. Each geofence ID is basically the identifier of the region location of a specific site a vehicle stops. Now I realized the goal of my application was to predict the stop duration of a vehicle within these different geofences. Also what I soon realized was that between each attribute there seemed to be no correlation between them and the duration recorded for that entry and only the *GeofenceID* had some correlation with it. This meant out of all the other attributes (*VehicleID, Category, Arrival*) the geofence location had the most impact on the duration of the stop. So my first decision was to partition the whole dataset into groups where each data entry is grouped by its *GeofenceID* value. This will leave me with partitions where the data entries within them all have the same stop region location. This will simplify the *K-means* algorithm as now I can obtain clusters within each partition to find the stop duration. This is without the need of worrying about which stop region location a data entry belongs to as the partition the data entry already belongs to, identifies which stop region location it is.

9.1.2 Initial Centroid Selection

After the whole dataset has been partitioned by *GeofenceID*, within each partition I have randomly selected five data entries to represent the initial centroids. This means the user defined value for K has been selected to be five. I have decided to initially go with the commonly used random initial centroid selection for now however, if that proves to result in poor clusters, I will look into a different technique in finding more accurate initial centroids. Also, I may need to tinker around with the value of K, to find the optimal number of clusters per geofence partition.

9.1.3 Cluster Assignments

From the selection of 5 random centroids within each partition, the next step in my method was to assign each data entry to the nearest centroid within its own geofence partition. As explained previously in the basic algorithm of the standard *K-means*, I have used the *Euclidean distance* technique to measure the closet distance between each data entry to its nearest centroid. The details of how I have used this technique is explained in the following sub-section.

9.1.4 Stop Duration Prediction

After the accurate clusters have successfully been obtained, the algorithm is now ready to predict the stop duration for a new data entry for which the stop duration is unknown. This is done in a similar way to how all the data entry objects were assigned to clusters, by finding its nearest centroid, in the cluster assignment stage. The only difference is that the duration attribute will not be present in the new data entry so the duration attribute will be left out in this calculation. For example, the following data entry is a new data entry being inputted into the algorithm to find its stop duration.

VehicleID	**GeofenceID**	**Category**	**Arrival**
A91E0C9	983055C6	Service Centre	22/10/14 11:10

The first thing that happens is that the algorithm checks to see if the *GeofenceID* in this new entry has already been recorded and thus already having a partition in the dataset. If it does in fact have a previously recorded *GeofenceID* then this entry is only compared to other data entry objects within this geofence partition.

If it is a new *GeofenceID* then it is compared with all other data entry objects in all partitions. Once the algorithm has decided on the geofence it now performs the similarity measure between itself and all other centroids (within its partition or all of them in total) and finds the one which it is most 'similar' to. Then the recorded stop duration of this centroid is selected to be the predicted stop duration for this new data entry. So this is how, given different sets of input data, a stop duration can be predictive from my algorithm.

9.2 Techniques

This sub-section of this report will give a detailed explanation of the techniques I have used in the implementation and evaluation of the K-*means* clustering algorithm in my application. Each technique is explained with a walkthrough example for better understanding.

9.2.1 Similarity Measure

The notion of *distance* between two data entries can be seen as how *similar* they are with each other. In other words, the similarity in the attributes of the data entries being compared. *Euclidean distance* can be used to measure this similarity. The formula for this can be defined as,

$$dist(A, B) = \sqrt{w_1(A_1 - B_1)^2 + w_2(A_2 - B_2)^2 \ldots + w_n(A_n - B_n)^2}$$

where the distance between data entry A and B, $dist(A, B)$, can be seen as the square root of the sum of the squared differences between the different attributes in each data entry, A_i and B_i. n is the total number of attributes each data entry contains and this value must be the same for A and B for this formula to work. w_i is the weight value given to each specific attribute, where a higher value means that attribute is more important in terms of similarity. The sum of all w_i to w_n should equal to 1.

By looking at the above formula for the *Euclidean distance*, it can be seen that the difference between each attribute can only be calculated with numerical attributes. However, the given sample data contains heterogeneous data, meaning it contains more than one data type. For instance, if the sample data does not contain all numerical attributes, other than the last duration attribute, the rest of the attributes are all categorical. (Categorical variables have been defined previously in this report). So now I will need to find a way in which to measure the difference between the categorical attributes. I found that the use of a Boolean result can be used to do such feat. For instance, if the comparing of two categorical attributes results in them being the same (true), the value 0 is returned, indicating that they are similar. Else if they are different (false) the value 1 is returned, indicating that they are different. Once the difference in the attributes has been found, each attribute is multiplied by its weight value and then the sum of result is square rooted to get the final distance between the two data entries.

An example of the first two entries in the sample data is a follows:

Data Entry	VehicleID	GeofenceID	Category	Arrival	Departure	Duration
1	A91E0C9	983055C6	Load/Un.	1/6/13 7:50	1/6/13 8:01	689
2	D544532	798A02BE	Load/Un.	4/6/13 5:07	4/06/13 6:46	5952

$$dist(1,2) = \sqrt{\begin{matrix} w_1(1_1 - 2_1)^2 + w_2(1_2 - 2_2)^2 + w_3(1_3 - 2_3)^2 \\ +w_4(1_4 - 2_4)^2 \end{matrix}}$$

$$= \sqrt{\begin{matrix} 0.15(\text{A91E0C96} == \text{D544532})^2 + 0.15(\text{Load/Un.} == \text{Load/Un.})^2 + \\ 0.15(June == \text{June})^2 + 0.15(\text{AM} == \text{AM})^2 + 0.4(0.06 - 0.49)^2 \end{matrix}}$$

$$= \sqrt{0.15(1)^2 + 0.15(0)^2 + 0.15(0)^2 \ + 0.15(0)^2 + 0.4(-0.43)^2}$$

$$= \sqrt{0.0225 + 0 + 0 \ + 0 + 0.1849}$$

$= 0.46 =$ the similarity distance between data entry 1 and 2

As shown by this above example I have not directly used the *Arrival* attribute as it is recorded in the sample data, in my *Euclidean distance* calculation. Instead, I decided to extract the month and time period from the *Arrival* attribute in each data entry. I decided to do this as the *Arrival* attribute initially had a date as well as a time value associated with it. And time is on a very small continuous scale and using this for the measure of similarity is obviously unfeasible. Thus, a morning (AM) or afternoon (PM) value was used. Also shown in the above calculations is the weight I have decided to give to each attribute in my implementation. The duration attribute is given a weight of 0.4 and the rest are given a weight of 0.15 each. The weights have been distributed this way as I believed the duration attribute should be more important, as the end goal is to ultimately predict this value. However, since the duration values recorded in the sample data are large numbers, I have decided to normalize them so that the calculations are not too weighted on one side. To do this normalizing, I have divided each duration value being used for calculation, by the maximum duration value in the recorded sample data. In the above example, the maximum duration is 12068. In the end of the calculation a value is obtained to represent the similarity distance between the two data entries. The lower this value is, the more similar the data entries are with each other. So now with my implementation of this *Euclidean distance* measure my application contains a method in finding the nearest centroid to a data entry object.

9.2.2 Internal Cluster Validation

Once I have used my implementation of the *Euclidean distance* to find the nearest centroid for each data entry, the assignment of each data entry was done to form the clusters within each partition. At this current stage, I decided to have a measure to check the accuracy of these obtained clusters. To do this measure I have used an *internal cluster validation* method called the *silhouette measure*. The purpose each individual cluster. This is done by the following formula:

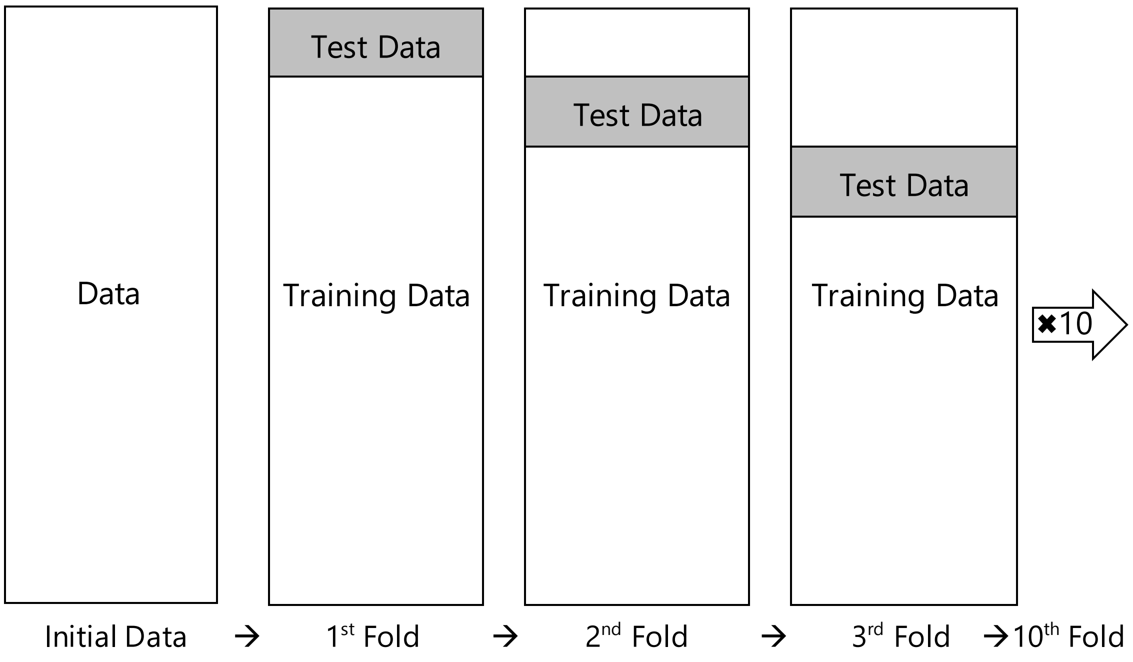
$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \ , \ -1 \leq s_i \leq 1$$

Where a_i is the average distance of data entry i to all other data entries within its own cluster, and b_i is the distance between data entry i and its closest neighbouring cluster. In the final result, the value of s_i will be a number between -1 and 1. What this value means is that if a positive value is obtained for s_i, then the data entry i is appropriately clustered to its cluster. If the obtained value is negative, this indicates that the data entry i is not clustered appropriately and possibly belongs in another cluster.

This measure is done for each of the five clusters within their respected geofence partitions. The results of this is seen in the next section.

9.2.3 Final Prediction Validation

Once my algorithm is able to produce accurate enough clusters, with the use of the internal cluster validation method, it is ready to produce an output prediction. However before the solution is finalized this final output prediction will need to be validated in terms of its accuracy. So the next stage in my development process is to implement a final prediction validation feature. The method I have chosen to do so is the *10x10-fold cross validation* method. The following information gives an illustrated explanation of this method:



This illustration shows how the basic process of this method is done. Starting from the left, it shows that with a given dataset the first step is to part the data into 10 parts, each with 10% of the data. Now 1 out of these 10 parts is chosen to be the test data, and the remaining 9 parts are chosen as the training data. The test data is kept aside and the training data is inputted into the clustering algorithm. After this 9 part, 90% of data is clustered into clusters represented by their according centroids, the entries from the test data is now inputted into the algorithm and each one is treated as a new data entry for which the stop duration is required. Remembering that the test data has been taking out from the initial dataset, each entry will already have a recorded duration value, so this value will need to be taken out of the similarity measure for this final prediction validation method to be effective. Once an entry from the test data has been predicted with a stop duration derived from the nearest centroid, this predicted stop duration value is compared to the actual duration value which was recorded in this entry. Then the difference between these two values is stored. Now looking at the neighboring centroid, in other terms, the second most similar centroid to the entry, the difference between the duration value of this centroid and the actual recorded duration value of the entry is also stored. Now with the two differences stored, one with the closest assigned centroid and one with the second closest neighbor centroid, both these differences are now compared together. If the difference between the actual duration for the entry and the predicted duration of the closest assigned centroid is smaller than the distance between the actual duration for the entry and the second closest centroid, then it can be seen that, for this entry the stop duration is predicted to its best value. This process is now repeated for all other entries in the test data and then once all 10% has been processed the average accuracy for this test data, also known as a fold, is recorded. Now this whole process is repeated 10 times, each time selecting a new 10%

from the data, and each time recording the average accuracy for each fold, hence the term *10x10-fold cross validation*.

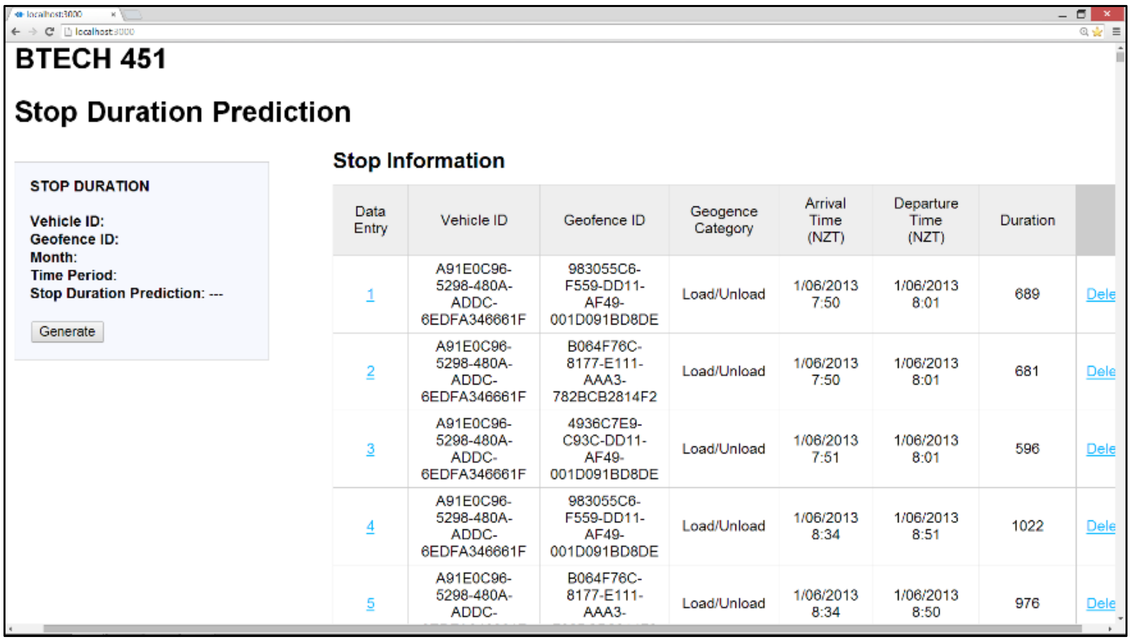
One drawback in this method is that if the initial dataset is somehow already grouped with its adjacent entries, dividing the dataset into 10 parts from top to bottom will not produce an appropriate test dataset and so will yield erroneous accuracy validation. To address this drawback I have decided that a 10% fold is selected with random entries from the dataset, and so each time a new fold is selected a random 10% will be used as the test data for more variety and so a more reliable test data is obtained. My application now as a method in testing the validation of a final stop duration prediction.

10 PREVIOUS APPLICATION

The following section of this report, gives an overview of how I decided to build my initial application. It provides screenshots of my previous application, an explanation on the design decisions I made, and then I describe the limitations of this version and then in the next section explain what I did to address these limitations.

10.1 Building the Application

Using *Node.js* and *MongoDB* meant that the application is meant to be a web application. The webpage templating engine I choice is *Jade*, as this is very clean and also very compatible with *Node.js*. As the purpose of my application was to work with a given dataset of information and process this information and then display results, I realized my application can be done on a single webpage. Since my application is used for statistical reasons and not used by an end-user, I do not need to worry much about the actual appearance of my application. So the layout of this webpage is kept very simple. The first step was to initialize a table in *Jade*, which will hold all the information obtained from the dataset, derived from the given sample data. This sample data was given in *Microsoft Excel* format and so I first saved this data as a CSV (Comma Separated Values) file then through *MongoDB* imported this CSV file into a database that my application is linked to. Then in my application, I loaded the data from the CSV file into the table I have previously initialized. The following screenshot shows the how this table looks when the data is loaded from the database. The information on the left side of the dataset table I have put in there is to show the final stop prediction of a certain vehicle at a certain geofence location, month, and time period. There is also a *Generate* button there that is used to being the processing.



BTECH 451

Stop Duration Prediction

STOP DURATION

Vehicle ID:
Geofence ID:
Month:
Time Period:
Stop Duration Prediction: ---

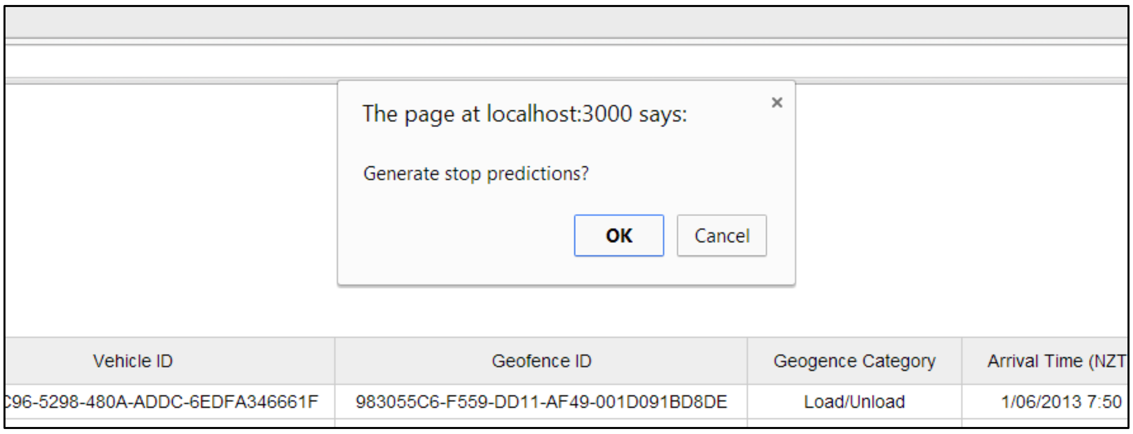
Generate

Stop Information

Data Entry	Vehicle ID	Geofence ID	Geogence Category	Arrival Time (NZT)	Departure Time (NZT)	Duration	
1	A91E0C96-5298-480A-ADDC-6EDFA346661F	983055C6-F559-DD11-AF49-001D091BD8DE	Load/Unload	1/06/2013 7:50	1/06/2013 8:01	689	Dele
2	A91E0C96-5298-480A-ADDC-6EDFA346661F	B064F76C-8177-E111-AAA3-782BCB2814F2	Load/Unload	1/06/2013 7:50	1/06/2013 8:01	681	Dele
3	A91E0C96-5298-480A-ADDC-6EDFA346661F	4936C7E9-C93C-DD11-AF49-001D091BD8DE	Load/Unload	1/06/2013 7:51	1/06/2013 8:01	596	Dele
4	A91E0C96-5298-480A-ADDC-6EDFA346661F	983055C6-F559-DD11-AF49-001D091BD8DE	Load/Unload	1/06/2013 8:34	1/06/2013 8:51	1022	Dele
5	A91E0C96-5298-480A-ADDC-	B064F76C-8177-E111-AAA3-	Load/Unload	1/06/2013 8:34	1/06/2013 8:50	976	Dele

10.2 Displaying Results

The next step for my application was to decide how I was going to display the many different results that are obtained throughout my implementation. As these were results which were just used for statistical reasons, I decided a simple alert box that pops up when each result is calculated can be used for the time being. The following screenshots shows this alert box displaying the different results:



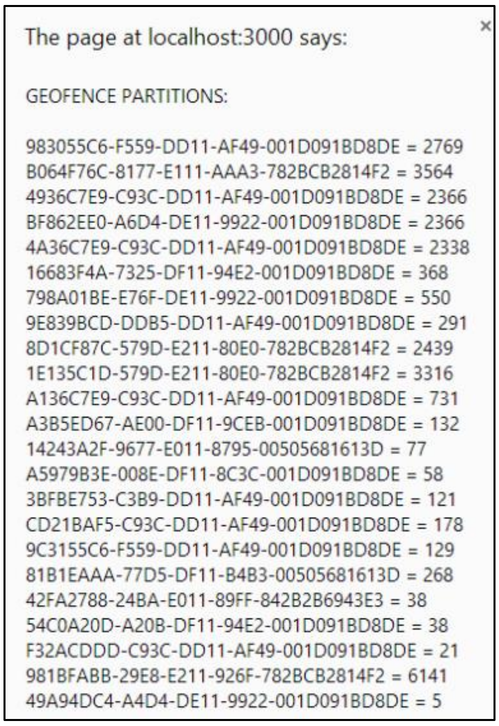
The page at localhost:3000 says:

Generate stop predictions?

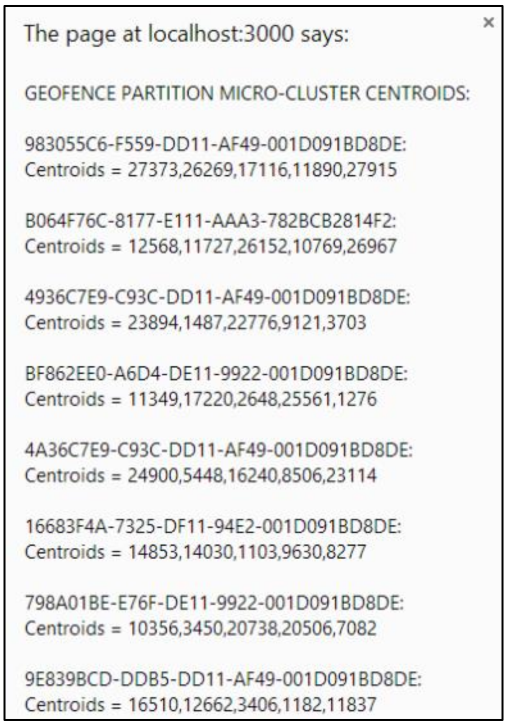
OK Cancel

Vehicle ID	Geofence ID	Geogence Category	Arrival Time (NZT
C96-5298-480A-ADDC-6EDFA346661F	983055C6-F559-DD11-AF49-001D091BD8DE	Load/Unload	1/06/2013 7:50

The above screenshot shows the confirmation box when the *Generate* button is first clicked. Once a user says *OK* the data processing starts.

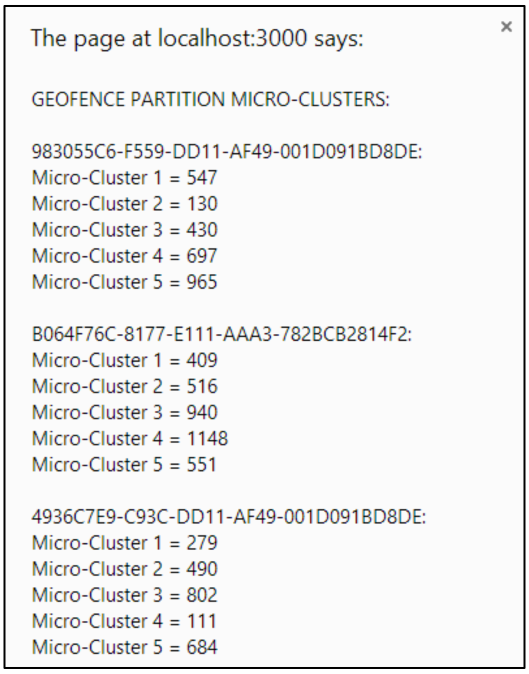


```
The page at localhost:3000 says:

GEOFENCE PARTITIONS:

983055C6-F559-DD11-AF49-001D091BD8DE = 2769
B064F76C-8177-E111-AAA3-782BCB2814F2 = 3564
4936C7E9-C93C-DD11-AF49-001D091BD8DE = 2366
BF862EE0-A6D4-DE11-9922-001D091BD8DE = 2366
4A36C7E9-C93C-DD11-AF49-001D091BD8DE = 2338
16683F4A-7325-DF11-94E2-001D091BD8DE = 368
798A01BE-E76F-DE11-9922-001D091BD8DE = 550
9E839BCD-DDB5-DD11-AF49-001D091BD8DE = 291
8D1CF87C-579D-E211-80E0-782BCB2814F2 = 2439
1E135C1D-579D-E211-80E0-782BCB2814F2 = 3316
A136C7E9-C93C-DD11-AF49-001D091BD8DE = 731
A3B5ED67-AE00-DF11-9CEB-001D091BD8DE = 132
14243A2F-9677-E011-8795-00505681613D = 77
A5979B3E-008E-DF11-8C3C-001D091BD8DE = 58
3BFBE753-C3B9-DD11-AF49-001D091BD8DE = 121
CD21BAF5-C93C-DD11-AF49-001D091BD8DE = 178
9C3155C6-F559-DD11-AF49-001D091BD8DE = 129
81B1EAAA-77D5-DF11-B4B3-00505681613D = 268
42FA2788-24BA-E011-89FF-842B2B6943E3 = 38
54C0A20D-A20B-DF11-94E2-001D091BD8DE = 38
F32ACDDD-C93C-DD11-AF49-001D091BD8DE = 21
981BFABB-29E8-E211-926F-782BCB2814F2 = 6141
49A94DC4-A4D4-DE11-9922-001D091BD8DE = 5
```

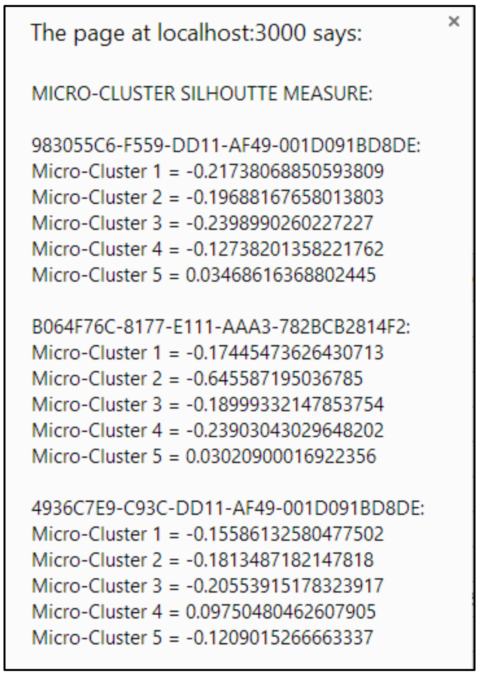


```
The page at localhost:3000 says:

GEOFENCE PARTITION MICRO-CLUSTER CENTROIDS:

983055C6-F559-DD11-AF49-001D091BD8DE:
Centroids = 27373,26269,17116,11890,27915

B064F76C-8177-E111-AAA3-782BCB2814F2:
Centroids = 12568,11727,26152,10769,26967

4936C7E9-C93C-DD11-AF49-001D091BD8DE:
Centroids = 23894,1487,22776,9121,3703

BF862EE0-A6D4-DE11-9922-001D091BD8DE:
Centroids = 11349,17220,2648,25561,1276

4A36C7E9-C93C-DD11-AF49-001D091BD8DE:
Centroids = 24900,5448,16240,8506,23114

16683F4A-7325-DF11-94E2-001D091BD8DE:
Centroids = 14853,14030,1103,9630,8277

798A01BE-E76F-DE11-9922-001D091BD8DE:
Centroids = 10356,3450,20738,20506,7082

9E839BCD-DDB5-DD11-AF49-001D091BD8DE:
Centroids = 16510,12662,3406,1182,11837
```

The alert box on the left shows the first step in my implementation. It shows a list of all the GeofenceID's recorded in the dataset, followed by the number of data entries with that GeofenceID. The alert box on the right shows the next step where within each geofence partition, five random data entries have been selected from it to represent the initial centroids. The numbers shown there are the *Data Entry* ID's of the selected data entries.



```
The page at localhost:3000 says:

GEOFENCE PARTITION MICRO-CLUSTERS:

983055C6-F559-DD11-AF49-001D091BD8DE:
Micro-Cluster 1 = 547
Micro-Cluster 2 = 130
Micro-Cluster 3 = 430
Micro-Cluster 4 = 697
Micro-Cluster 5 = 965

B064F76C-8177-E111-AAA3-782BCB2814F2:
Micro-Cluster 1 = 409
Micro-Cluster 2 = 516
Micro-Cluster 3 = 940
Micro-Cluster 4 = 1148
Micro-Cluster 5 = 551

4936C7E9-C93C-DD11-AF49-001D091BD8DE:
Micro-Cluster 1 = 279
Micro-Cluster 2 = 490
Micro-Cluster 3 = 802
Micro-Cluster 4 = 111
Micro-Cluster 5 = 684
```



```
The page at localhost:3000 says:

MICRO-CLUSTER SILHOUTTE MEASURE:

983055C6-F559-DD11-AF49-001D091BD8DE:
Micro-Cluster 1 = -0.21738068850593809
Micro-Cluster 2 = -0.19688167658013803
Micro-Cluster 3 = -0.2398990260227227
Micro-Cluster 4 = -0.12738201358221762
Micro-Cluster 5 = 0.03468616368802445

B064F76C-8177-E111-AAA3-782BCB2814F2:
Micro-Cluster 1 = -0.17445473626430713
Micro-Cluster 2 = -0.645587195036785
Micro-Cluster 3 = -0.18999332147853754
Micro-Cluster 4 = -0.23903043029648202
Micro-Cluster 5 = 0.03020900016922356

4936C7E9-C93C-DD11-AF49-001D091BD8DE:
Micro-Cluster 1 = -0.15586132580477502
Micro-Cluster 2 = -0.1813487182147818
Micro-Cluster 3 = -0.20553915178323917
Micro-Cluster 4 = 0.09750480462607905
Micro-Cluster 5 = -0.1209015266663337
```

The above two alert boxes show the final clustering results. The left box shows that in each geofence partition, with the 5 random centroids previously selected within them, the data entries have been assigned to form 5 clusters. The number beside each cluster is the number of data entries that have been assigned to that specific cluster. The alert box on the right shows the results of the internal cluster validation method of the silhouette measure. It shows the average silhouette value for each cluster within each partition.

10.3 Limitations

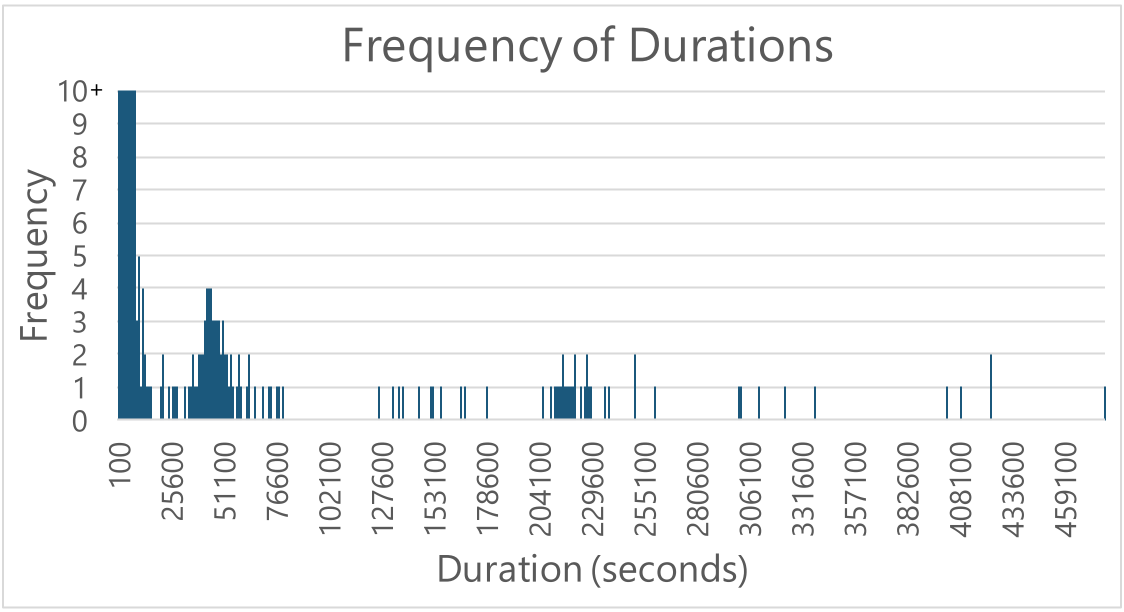
Applying the *silhouette measure* to validate the accuracy of my clusters, proved to show that the final clusters I have obtained, are mostly inaccurate. As seen by the silhouette values in the previous section most values are negative, indicating that most entries are not appropriately clustered and belong to different cluster than the one they have been assigned to. I have looked over my implementation to find possible factors that have influenced this outcome of my clusters. The first factor that came to mind was the fact of using a random selection to initialize the centroids, as this method is known to produce poor clusters. To test how much this factor has actually influenced the outcome I have performed the method mentioned previously, of running the clustering algorithm multiple times. However, even after this, applying the *silhouette measure* once again proved to result in negative values, with only a slight deviation from the original values. So this made me believe that there were more factors that were influencing my clustering algorithm. These factors and how I improved on them are explained in the next section of this report.

11 IMPROVEMENTS MADE

As explained in the previous section, the accuracy of my final clusters were initially very limited. Before I moved on to the next steps of my implementation, I needed to make sure my clusters were actually accurate enough. This is important as without accurate clusters to work with, the final prediction will be very inaccurate as well. To increase the accuracy of my clusters I looked deeper into my implementation to find the factors that were influencing poor clusters. This section looks into these factors and their influence on the accuracy of the clusters obtained.

11.1 Removing of Outliers

One possible factor included the involvement of a number of outliers within the dataset. Even though the *Duration* attribute has been normalized to avoid the very large outlier numbers, that whole data entry is still being used in the calculations. This is a known issue as the *k-means* clustering algorithm is very sensitive to outliers, so my next action was to try my implementation after removing these outliers that were outside the bounds of a certain threshold. The following shows a graph of the frequencies of the different durations recorded in the given sample data.



It can be clearly seen that most of the recorded durations are below around the 80000 seconds mark. After that there is a wide dispersion of frequencies with four large outliers after the 400000 seconds mark. So what I decided to do in my implementation was to assume that the wide dispersion entries were due to other factors (for example the vehicle breaking down) and so I ended up removing all entries that had a duration of larger than 80000 seconds. So now my algorithm is working with a cleaner dataset with no outlier entries to hinder the accuracy of the clusters.

11.2 Increasing Number of Runs

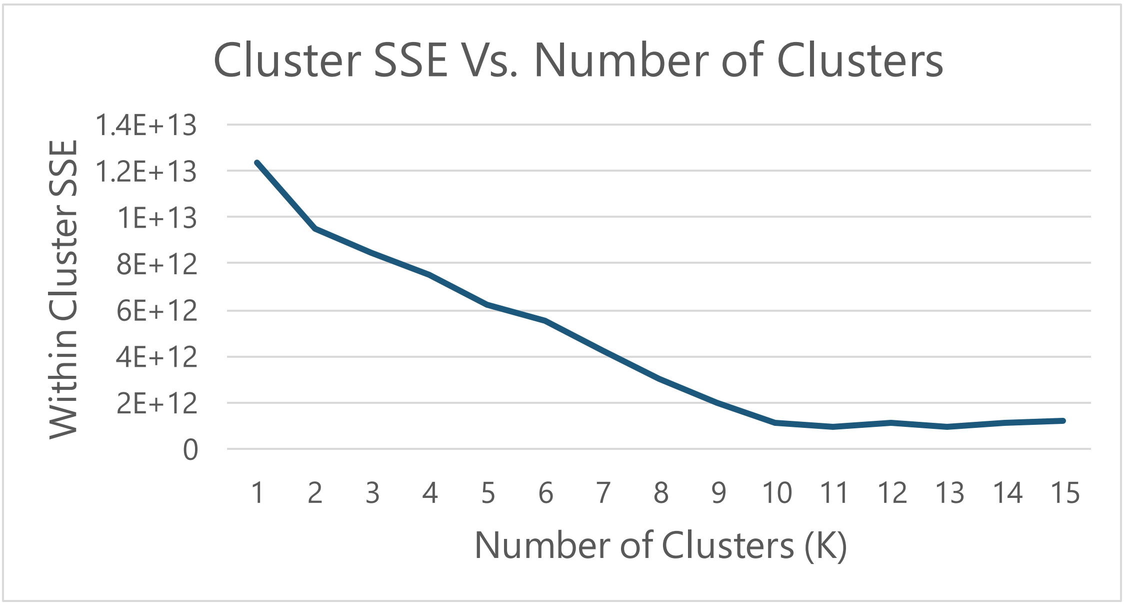
Another method of increasing the accuracy of my clusters involved in increasing the number of runs my algorithm takes. The advantages of this method has been explained previously in this report however in simple terms, the number of runs my algorithm takes directly corresponds to how accurate the final clusters are. The basic *K-means* clustering algorithm describes the number of runs being the amount needed until no change to the centroids happen, however due to performance and other possible issues of the dataset being uncorrelated and slightly random, my algorithm is not able to run that many times and so, my initial run number was chosen to be 25. I decided to increase this number to 50 as this had no major effect on performance but will still produce more accurate clusters.

11.3 Finding Optimal Number of Clusters

One last method I have used to increase the accuracy of my clusters was to find the optimal number of clusters to be used in the algorithm process. This method invloved doing more work than the previous two. It involved introducing a new technique which bassically checks the compactness of the data entry objects within each of the final clusters obtained. This technique is called the *within-cluster sum of squared errors* calculation and is formally defined as so:

$$SSE = \sum_{p=1}^{Part.} \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$$

Where p is the number of partitions, i is the number of clusters, C_i is the ith cluster, c_i is the centroid of the cluster C_i, x is a data entry object and finally $dist$ is the similary measure, in this case *Euclidean distance*. What this technique does is that with each object in a cluster, it gets the distance between that object and the centroid of that cluster. Then all the distances for each obejct in a cluster is sumed together and then all the cluster sum distances are sumed together and then finally all the partition sums are sumed together, in the end getting the SSE value which is the total distances between data entires within the clusters. Obviously a smaller SSE value means that the data entry objects are closer together in their respectfull clusters where a larger values means they are further apart. In other terms, the smaller the SSE the more approprate the data entries have been clustered. Now with this technique in hand the next thing to do was to run my algorithm with different values fo *K*, meaning different number of clusters and then observing the SSE value obtained in the final stage of each run. The following is a graph showing the results of this observation.



From this graph it can be quite evidently seen that with the increase of the number of clusters the SSE value does in fact decrease. However, after a certain number of clusters, the SSE value does not seem to have any dramatic improvements. This is seen by the "elbow" in the line on the graph, which is around the 10 number of clusters mark. This means that after 10, any additional increases of the number of clusters will yield no decrease in the SSE and so will only be redundant changes. This means the optimal number of clusters for clustering my dataset will be is 10. This is twice the amount I initially decided with and so the total SSE should be significantly lower and so in turn should increase the accuracy of the final obtained clusters.

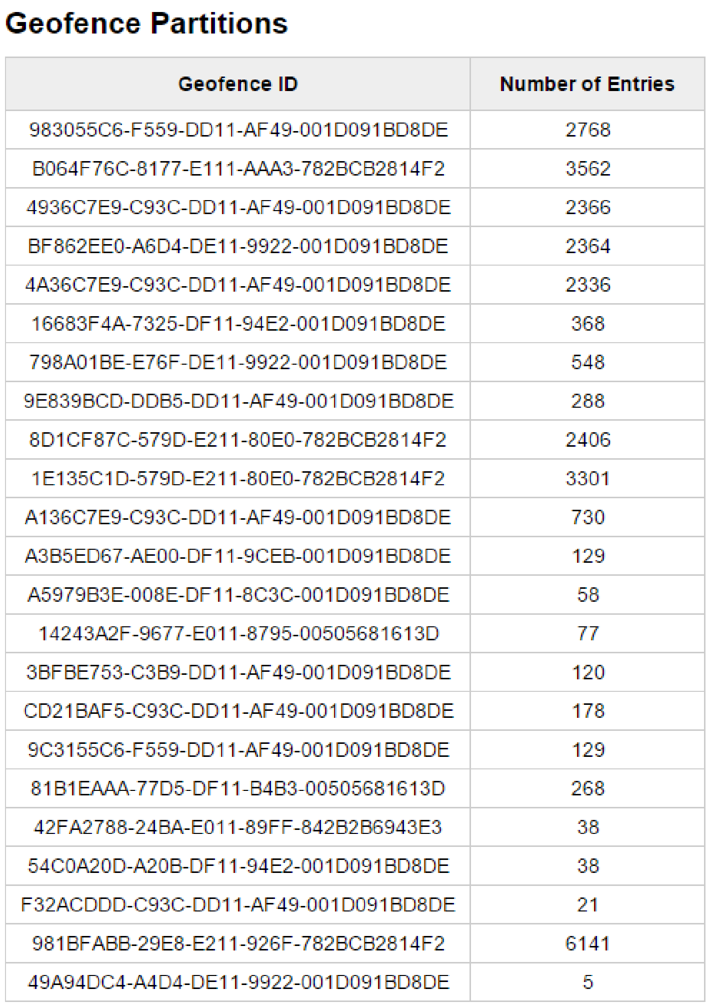
These are the improvements I made after the initial implementation of my application. The next section of this report shows the new application after the improvements and other changes were made.

12 FINAL APPLICATION

This section of the report shows the final application that was developed to address the issues and limitations found in the initial version. This section provides screenshots of the changes made to the interface of the application, when displaying results, and also presents the improvements made in the clustering algorithm.

12.1 Interface Changes

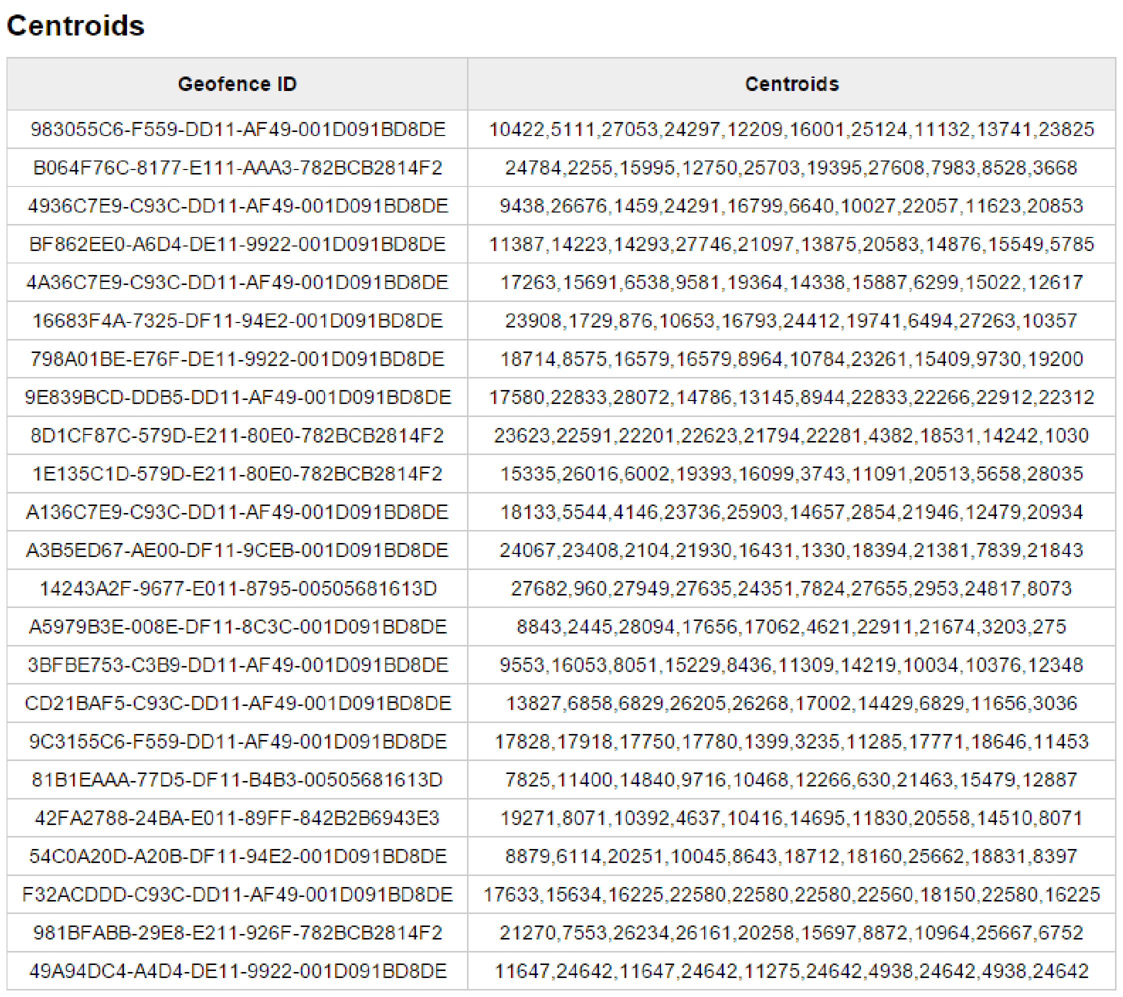
The following screenshot is a basic representation of the change I made in displaying the results produced in my application. Instead of alert boxes that popped up one after another I decided to put all results in tables all on the single webpage. This not only allowed more of results of the data to be shown but also, it provided a better indication of what each value represented, with the use of table column headings.



Geofence Partitions

Geofence ID	Number of Entries
983055C6-F559-DD11-AF49-001D091BD8DE	2768
B064F76C-8177-E111-AAA3-782BCB2814F2	3562
4936C7E9-C93C-DD11-AF49-001D091BD8DE	2366
BF862EE0-A6D4-DE11-9922-001D091BD8DE	2364
4A36C7E9-C93C-DD11-AF49-001D091BD8DE	2336
16683F4A-7325-DF11-94E2-001D091BD8DE	368
798A01BE-E76F-DE11-9922-001D091BD8DE	548
9E839BCD-DDB5-DD11-AF49-001D091BD8DE	288
8D1CF87C-579D-E211-80E0-782BCB2814F2	2406
1E135C1D-579D-E211-80E0-782BCB2814F2	3301
A136C7E9-C93C-DD11-AF49-001D091BD8DE	730
A3B5ED67-AE00-DF11-9CEB-001D091BD8DE	129
A5979B3E-008E-DF11-8C3C-001D091BD8DE	58
14243A2F-9677-E011-8795-00505681613D	77
3BFBE753-C3B9-DD11-AF49-001D091BD8DE	120
CD21BAF5-C93C-DD11-AF49-001D091BD8DE	178
9C3155C6-F559-DD11-AF49-001D091BD8DE	129
81B1EAAA-77D5-DF11-B4B3-00505681613D	268
42FA2788-24BA-E011-89FF-842B2B6943E3	38
54C0A20D-A20B-DF11-94E2-001D091BD8DE	38
F32ACDDD-C93C-DD11-AF49-001D091BD8DE	21
981BFABB-29E8-E211-926F-782BCB2814F2	6141
49A94DC4-A4D4-DE11-9922-001D091BD8DE	5

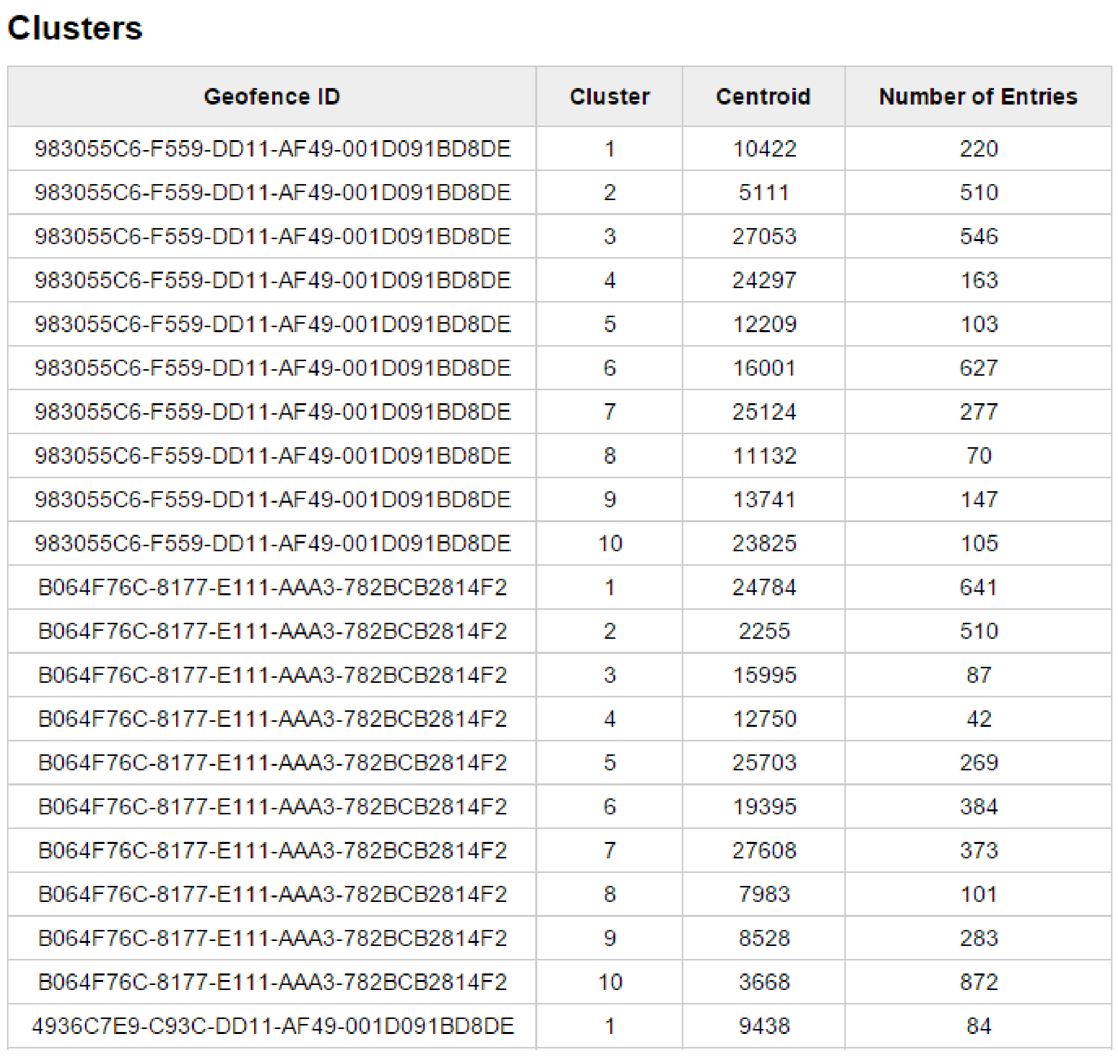
The above screenshot is a snippet showing the geofence partition table. It essentially is providing the same information that my previous application provided; a list of all the geofences recorded in the dataset with the number of entries with that particular geofence attribute.



Centroids

Geofence ID	Centroids
983055C6-F559-DD11-AF49-001D091BD8DE	10422,5111,27053,24297,12209,16001,25124,11132,13741,23825
B064F76C-8177-E111-AAA3-782BCB2814F2	24784,2255,15995,12750,25703,19395,27608,7983,8528,3668
4936C7E9-C93C-DD11-AF49-001D091BD8DE	9438,26676,1459,24291,16799,6640,10027,22057,11623,20853
BF862EE0-A6D4-DE11-9922-001D091BD8DE	11387,14223,14293,27746,21097,13875,20583,14876,15549,5785
4A36C7E9-C93C-DD11-AF49-001D091BD8DE	17263,15691,6538,9581,19364,14338,15887,6299,15022,12617
16683F4A-7325-DF11-94E2-001D091BD8DE	23908,1729,876,10653,16793,24412,19741,6494,27263,10357
798A01BE-E76F-DE11-9922-001D091BD8DE	18714,8575,16579,16579,8964,10784,23261,15409,9730,19200
9E839BCD-DDB5-DD11-AF49-001D091BD8DE	17580,22833,28072,14786,13145,8944,22833,22266,22912,22312
8D1CF87C-579D-E211-80E0-782BCB2814F2	23623,22591,22201,22623,21794,22281,4382,18531,14242,1030
1E135C1D-579D-E211-80E0-782BCB2814F2	15335,26016,6002,19393,16099,3743,11091,20513,5658,28035
A136C7E9-C93C-DD11-AF49-001D091BD8DE	18133,5544,4146,23736,25903,14657,2854,21946,12479,20934
A3B5ED67-AE00-DF11-9CEB-001D091BD8DE	24067,23408,2104,21930,16431,1330,18394,21381,7839,21843
14243A2F-9677-E011-8795-00505681613D	27682,960,27949,27635,24351,7824,27655,2953,24817,8073
A5979B3E-008E-DF11-8C3C-001D091BD8DE	8843,2445,28094,17656,17062,4621,22911,21674,3203,275
3BFBE753-C3B9-DD11-AF49-001D091BD8DE	9553,16053,8051,15229,8436,11309,14219,10034,10376,12348
CD21BAF5-C93C-DD11-AF49-001D091BD8DE	13827,6858,6829,26205,26268,17002,14429,6829,11656,3036
9C3155C6-F559-DD11-AF49-001D091BD8DE	17828,17918,17750,17780,1399,3235,11285,17771,18646,11453
81B1EAAA-77D5-DF11-B4B3-00505681613D	7825,11400,14840,9716,10468,12266,630,21463,15479,12887
42FA2788-24BA-E011-89FF-842B2B6943E3	19271,8071,10392,4637,10416,14695,11830,20558,14510,8071
54C0A20D-A20B-DF11-94E2-001D091BD8DE	8879,6114,20251,10045,8643,18712,18160,25662,18831,8397
F32ACDDD-C93C-DD11-AF49-001D091BD8DE	17633,15634,16225,22580,22580,22580,22560,18150,22580,16225
981BFABB-29E8-E211-926F-782BCB2814F2	21270,7553,26234,26161,20258,15697,8872,10964,25667,6752
49A94DC4-A4D4-DE11-9922-001D091BD8DE	11647,24642,11647,24642,11275,24642,4938,24642,4938,24642

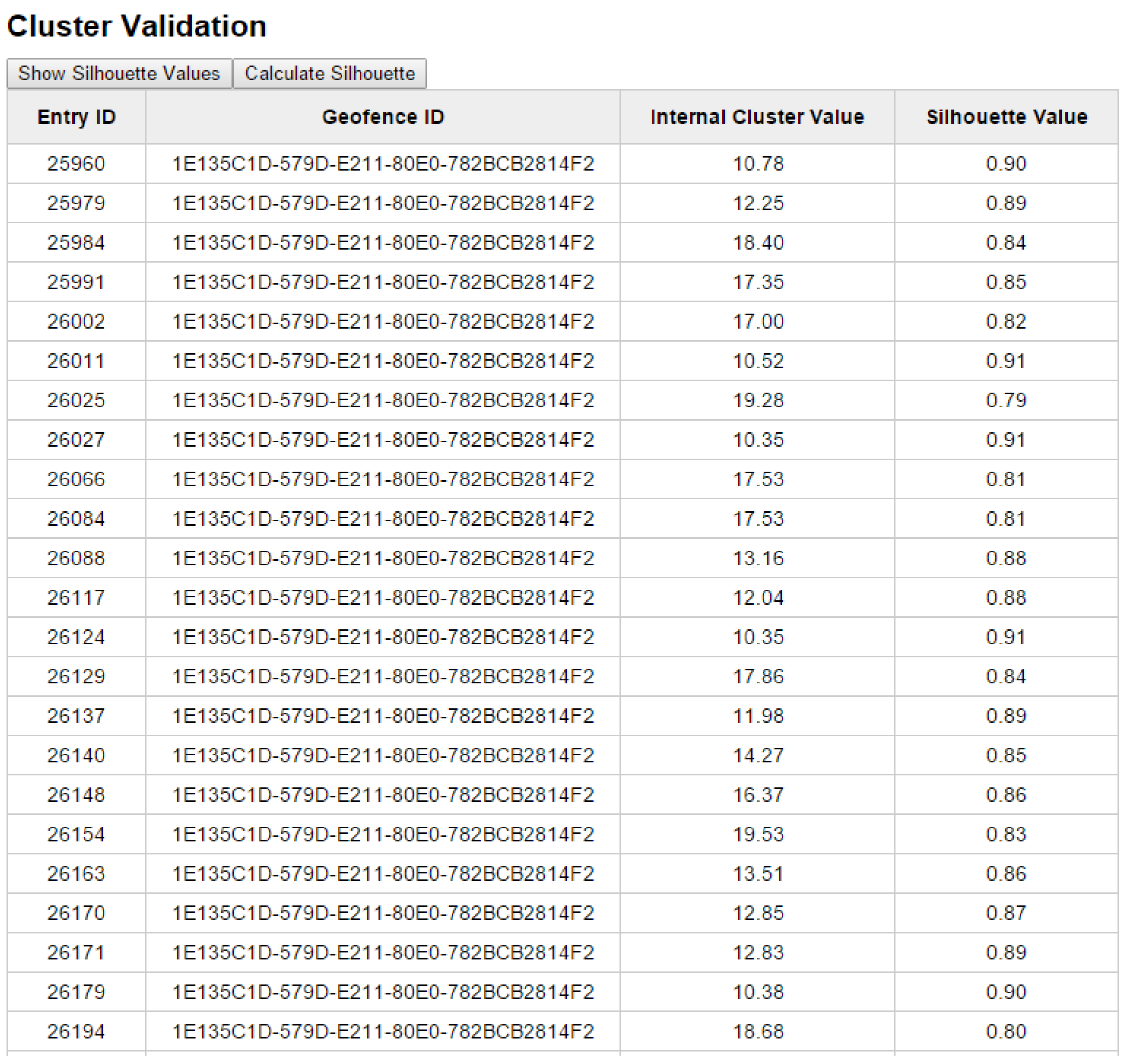
The above screenshot is a snippet showing the centroids table. Again, this is essentially showing the same information as my previous application (list of all the geofences with their 10 entry IDs chosen to be their final centroids) however, making it a table rather than an alert box allows for all pieces of the information to be shown with a cleaner presentation. Since the assignment of these centroids is random, the final displayed entry IDs in the *Centroids* column will be different each the algorithm is run.



Clusters

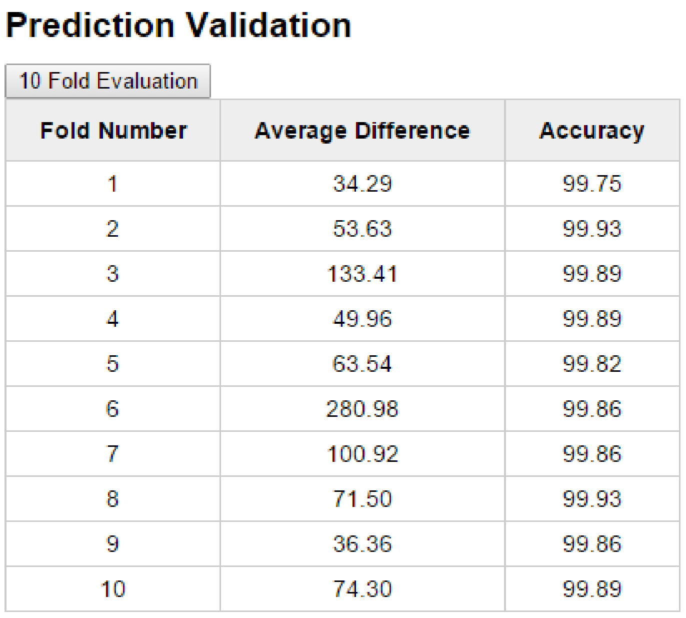
Geofence ID	Cluster	Centroid	Number of Entries
983055C6-F559-DD11-AF49-001D091BD8DE	1	10422	220
983055C6-F559-DD11-AF49-001D091BD8DE	2	5111	510
983055C6-F559-DD11-AF49-001D091BD8DE	3	27053	546
983055C6-F559-DD11-AF49-001D091BD8DE	4	24297	163
983055C6-F559-DD11-AF49-001D091BD8DE	5	12209	103
983055C6-F559-DD11-AF49-001D091BD8DE	6	16001	627
983055C6-F559-DD11-AF49-001D091BD8DE	7	25124	277
983055C6-F559-DD11-AF49-001D091BD8DE	8	11132	70
983055C6-F559-DD11-AF49-001D091BD8DE	9	13741	147
983055C6-F559-DD11-AF49-001D091BD8DE	10	23825	105
B064F76C-8177-E111-AAA3-782BCB2814F2	1	24784	641
B064F76C-8177-E111-AAA3-782BCB2814F2	2	2255	510
B064F76C-8177-E111-AAA3-782BCB2814F2	3	15995	87
B064F76C-8177-E111-AAA3-782BCB2814F2	4	12750	42
B064F76C-8177-E111-AAA3-782BCB2814F2	5	25703	269
B064F76C-8177-E111-AAA3-782BCB2814F2	6	19395	384
B064F76C-8177-E111-AAA3-782BCB2814F2	7	27608	373
B064F76C-8177-E111-AAA3-782BCB2814F2	8	7983	101
B064F76C-8177-E111-AAA3-782BCB2814F2	9	8528	283
B064F76C-8177-E111-AAA3-782BCB2814F2	10	3668	872
4936C7E9-C93C-DD11-AF49-001D091BD8DE	1	9438	84

The above screenshot is showing a snippet of the clusters table. This version is showing slightly more information than the alert box in its previous version. This table is showing the number of entries assigned to each of the 10 clusters for each geofence ID. The entry ID of the centroid of each cluster is also shown. Once again the displayed entry ID of the centroid will change with different runs of the clustering algorithm which will in turn change the values of the number of corresponding entries for that cluster. After this step of my implementation, I was now able to see that some clusters in particular geofence partitions where have very low number of entries assigned to them, sometimes even none. To deal with this I decided to not include all the clusters with less than 10 entries assigned to it. This ensures that the clustering algorithm will not assign a data entry to an inaccurately represented cluster.


Cluster Validation

Entry ID	Geofence ID	Internal Cluster Value	Silhouette Value
25960	1E135C1D-579D-E211-80E0-782BCB2814F2	10.78	0.90
25979	1E135C1D-579D-E211-80E0-782BCB2814F2	12.25	0.89
25984	1E135C1D-579D-E211-80E0-782BCB2814F2	18.40	0.84
25991	1E135C1D-579D-E211-80E0-782BCB2814F2	17.35	0.85
26002	1E135C1D-579D-E211-80E0-782BCB2814F2	17.00	0.82
26011	1E135C1D-579D-E211-80E0-782BCB2814F2	10.52	0.91
26025	1E135C1D-579D-E211-80E0-782BCB2814F2	19.28	0.79
26027	1E135C1D-579D-E211-80E0-782BCB2814F2	10.35	0.91
26066	1E135C1D-579D-E211-80E0-782BCB2814F2	17.53	0.81
26084	1E135C1D-579D-E211-80E0-782BCB2814F2	17.53	0.81
26088	1E135C1D-579D-E211-80E0-782BCB2814F2	13.16	0.88
26117	1E135C1D-579D-E211-80E0-782BCB2814F2	12.04	0.88
26124	1E135C1D-579D-E211-80E0-782BCB2814F2	10.35	0.91
26129	1E135C1D-579D-E211-80E0-782BCB2814F2	17.86	0.84
26137	1E135C1D-579D-E211-80E0-782BCB2814F2	11.98	0.89
26140	1E135C1D-579D-E211-80E0-782BCB2814F2	14.27	0.85
26148	1E135C1D-579D-E211-80E0-782BCB2814F2	16.37	0.86
26154	1E135C1D-579D-E211-80E0-782BCB2814F2	19.53	0.83
26163	1E135C1D-579D-E211-80E0-782BCB2814F2	13.51	0.86
26170	1E135C1D-579D-E211-80E0-782BCB2814F2	12.85	0.87
26171	1E135C1D-579D-E211-80E0-782BCB2814F2	12.83	0.89
26179	1E135C1D-579D-E211-80E0-782BCB2814F2	10.38	0.90
26194	1E135C1D-579D-E211-80E0-782BCB2814F2	18.68	0.80

The above screenshot is showing a snippet of the internal cluster validation table. This is slightly different to how the results of this validation was shown in the previous version of the application. Instead of showing cluster averages for the silhouette measure values, I decided it will be more appropriate to show the silhouette measure value for each data entry itself. This way it is a lot easier to identify the exact entry in which produces a negative silhouette value and so allows easy evaluation of why it produced the negative value. The *Internal Cluster Value* column represents the average distance that particular data entry as to all other entries within its cluster. The *Calculate Silhouette* button above the table simply runs the silhouette measure algorithm after the clusters have been updated. The *Show Silhouette Values* button, simply loads the table from the database and then displays it onto the webpage. The evaluation of the silhouette values will be looked over in the next section of this report.

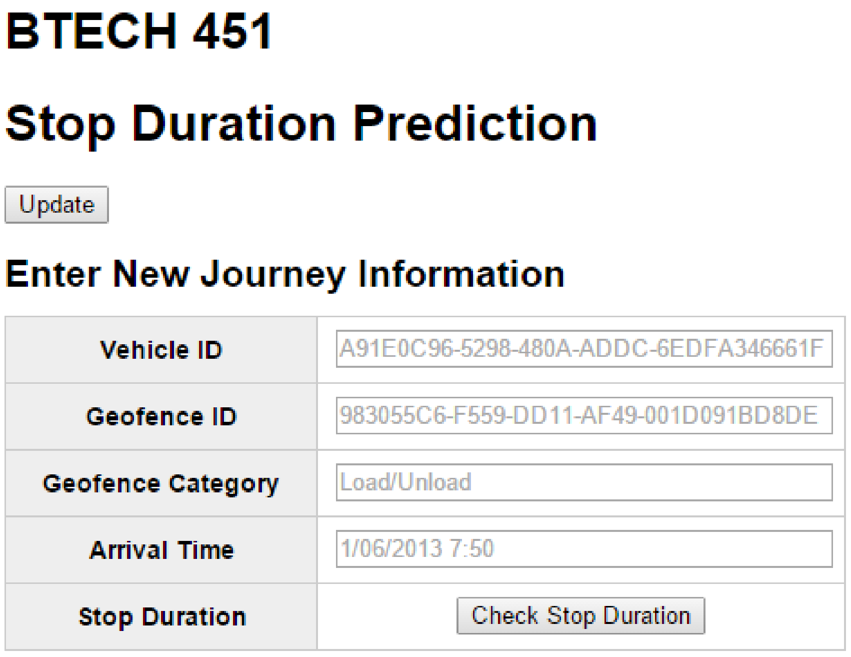

Prediction Validation

Fold Number	Average Difference	Accuracy
1	34.29	99.75
2	53.63	99.93
3	133.41	99.89
4	49.96	99.89
5	63.54	99.82
6	280.98	99.86
7	100.92	99.86
8	71.50	99.93
9	36.36	99.86
10	74.30	99.89

Finally, the above screenshot is showing the table displaying the results of the final prediction validation method. The *Fold Number* column displays the fold number and each of the random 10% test data used, the *Average Difference* column displays the average difference in the actual recorded stop duration to the predicted stop duration, and the *Accuracy* column displays the accuracy of the prediction (whether the predicted stop duration is more accurate than any other possible prediction). The *10 Fold Evaluation* button simply runs the 10 x 10-Fold Cross Validation method on the recently updated clusters. Again, the results of this method are explained in the next section.

12.2 New Data Entry Input

After I had implemented the final prediction evaluation method, and its results were satisfactory (explained in next section), I now was nearly done with the development of this project. This final crucial feature left to add was a way to let a user input a new data entry for which they will want to know the stop duration for. To do this, I decided an input form would be the most simple and effective way to perform this task. The following screenshots show this form and how it works:



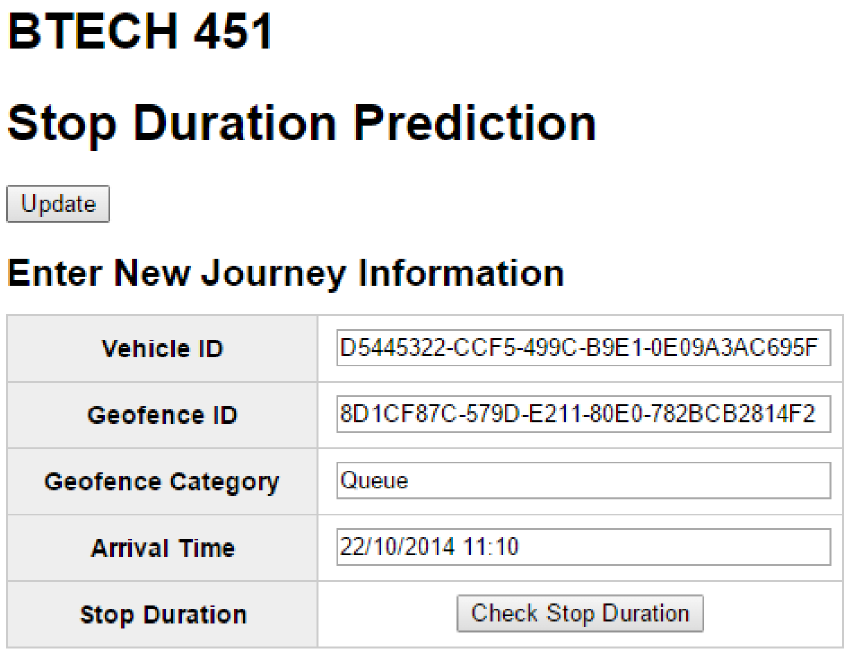
BTECH 451

Stop Duration Prediction

Enter New Journey Information

Vehicle ID	A91E0C96-5298-480A-ADDC-6EDFA346661F
Geofence ID	983055C6-F559-DD11-AF49-001D091BD8DE
Geofence Category	Load/Unload
Arrival Time	1/06/2013 7:50
Stop Duration	Check Stop Duration

This first screenshot shows how the form looks initially. As it is shown, the form consists of five fields, *Vehicle ID, Geofence ID, Geofence Category*, *Arrival Time*, and *Stop Duration*. The first four fields are the attributes of the new data entry the user will input and the last field consists of the *Check Stop Duration* button which will, once pressed, produce the stop duration prediction of this new entry. The *Update* button at the top is the main button used to run the clustering algorithm on the data loaded from the database to form the clusters. This is the same as the *Generate* button in the initial application.



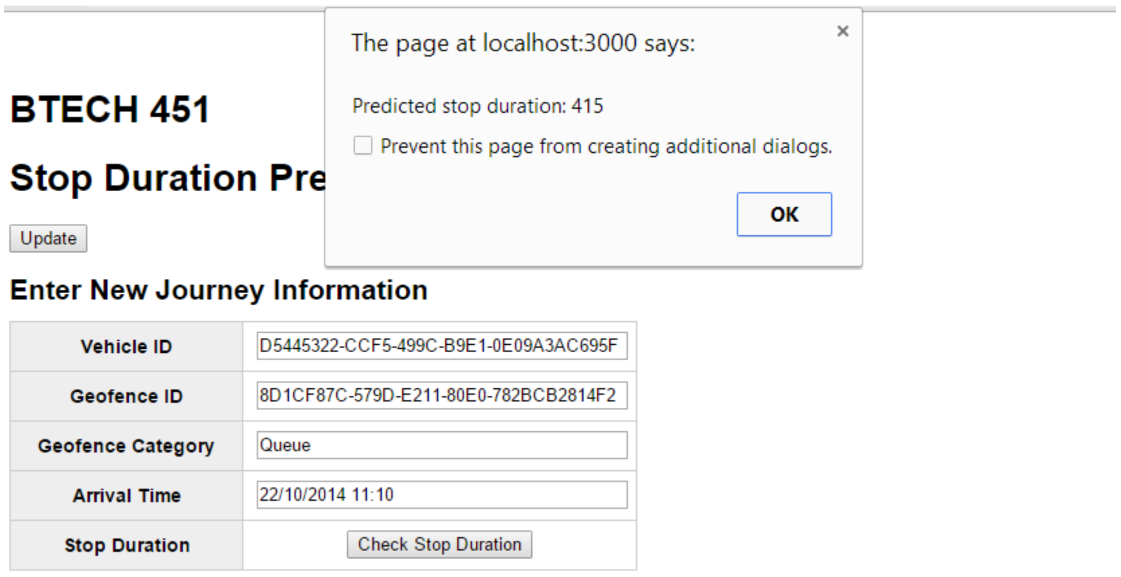
BTECH 451

Stop Duration Prediction

Enter New Journey Information

Vehicle ID	D5445322-CCF5-499C-B9E1-0E09A3AC695F
Geofence ID	8D1CF87C-579D-E211-80E0-782BCB2814F2
Geofence Category	Queue
Arrival Time	22/10/2014 11:10
Stop Duration	Check Stop Duration

The second screenshot above shows a basic example of a new data entry being inputted into the form. Now once the *Check Stop Duration* button is pressed, the inputted attributes of this new entry will now be reading to be compared with. If the newly inputted geofence ID is already in the dataset then this new entry will be compared to only the centroids of the clusters within that same geofence ID, else it will be compared to all other centroids of all other clusters in all partitions. The result of this is shown in the following screenshot:



The page at localhost:3000 says:

Predicted stop duration: 415

Prevent this page from creating additional dialogs.

OK

BTECH 451

Stop Duration Pre

Update

Enter New Journey Information

Vehicle ID	D5445322-CCF5-499C-B9E1-0E09A3AC695F
Geofence ID	8D1CF87C-579D-E211-80E0-782BCB2814F2
Geofence Category	Queue
Arrival Time	22/10/2014 11:10
Stop Duration	Check Stop Duration

This screenshot shows the final result after the new data entry has found its nearest centroid. What happens is the duration attribute recorded to the data entry representing this centroid will be used as the prediction for the new data entry's stop duration. A simple alert box is used to inform the user of this prediction. In this case, *415 seconds*.

13 EVALUATION

This section of the report provides an overview on the final evaluation done to my final application. It presents the results obtained from both of the different evaluation techniques implemented in this project. These techniques included the internal cluster validation, using the *silhouette measure* method and the final prediction evaluation, using the *10x10-fold cross validation* method. As a simple reminder on what the purpose of the techniques are, before going over the final results; the *silhouette measure* is a method in checking the accuracy of the obtained clusters and how well each data entry object is clustered; the *10x10-*

fold cross validation is a method for checking the accuracy of the final prediction resulted from the clustering algorithm.

13.1 Silhouette Measure Results

After implementing the needed improvements from my initial application, applying the *silhouette measure* to the new application proved to produce significantly more accurate silhouette values. This can be seen by the screenshot snippet of a set of silhouette values in the previous section of this report. In this snippet it can be shown that for each of the displayed data entries the silhouette values are around 0.8. Recall that a silhouette value is between -1 and 1 and being closer to 1 means that the certain data entry is appropriately clustered. So having 0.8 indicates a fairly good clustering. It is not shown in the previous screenshot snippet however, most of the data entries now produce a silhouette value all above around 0.75, with only a couple negative values. The following shows an example comparison between the average cluster silhouette values produced by my initial application with the values produced by my final application. As shown in the screenshot of the silhouette values alert box in my initial application section, looking at the partition with *983055C6-F559-DD11-AF49-001D091BD8DE* as the geofence ID, each of the five clusters had the corresponding average silhouette values of *-0.22, -0.20, -0.24, -0.13, 0.03*. Now looking at the same geofence partition, each of the now 10 clusters have the following silhouette values, *0.74, 0.65, 0.91, 0.92, 0.82, 0.88, 0.77, 0.69, 0.90 and 0.90*. This example shows that the improvements I had made after the initial application, were in fact successful in obtaining more accurate final clusters. I am no able to move onto the next stage in my implementation.

13.2 10x10-Fold Cross Validation Results

Once my implantation of the clustering algorithm was able to produce accurate clusters, I implemented the final prediction evaluation method called, *10x10-fold cross validation*. The screenshot of the prediction validation table in the previous section showed that all 10 folds had an average accuracy of over 99%. Recalling that the average accuracy in this validation method correlates to if a data entry is predicted to have the closest stop duration compared to the actual duration it was recorded with. So if each fold is producing an average accuracy of over 99% then it is safe to say that my implementation will produce a very accurate final prediction of stop durations.

14 FUTURE WORK

As seen in this report, more specifically the final evaluation of my implementation of the clustering algorithm used in my application, it can be shown that my implementation is fairly successful in performing the intended task – predicting the stop duration of vehicles at different locations. However, there are many areas in which my implementation will need a deeper analysis, to not only improve the implementation but also, make it more efficient in the future. This section highlights a few of these areas.

14.1 Choosing Initial Number for K

As explained in my implementation the chosen number of initial centroids was determined by graphing the within cluster SSE against the number of different centroids. However, this involved in using a pre-processing step of running the algorithm with different number of clusters each to find the most optimal number. This can not only be time consuming, but also requires for the actual clustering algorithm to be implemented before a final number of clusters is decided upon. And so, different methods of choosing the number of clusters can be looked into, mainly in terms of dynamically finding the optimal number of initial centroids by simply analyzing the dataset. The difficulty with this, is that it is sometimes almost impossible to find the optimal number of centroids in an unstructured dataset.

14.2 Updating Centroids Incrementally

In my implementation the final centroids were chosen to be the most accurate ones in the many iterations of running the clustering algorithm. Another way of finding these centroids can be done by not selection but by update. After the initial random centroids are selected, in the assignment stage, when each data entry is assigned to its nearest centroid, the mean between this data entry and the centroid can now be used as the new centroid of the cluster. This is repeated until all the data entries have been processed and assigned, each time creating a new centroid. This in turn will not result in the final centroids being actual entries in the dataset but, the actual mean value of the cluster. This may or may not be more accurate depending on the assignment stage. A possible disadvantage of this method is that if each data entry being assigned to the nearest centroid is coming from an ordered dataset, the produced clusters will be a representation of this order and so will not be as accurate. Also this method of incrementally updating the centroid after each assignment means that the whole process will be much longer to finalize.

14.3 Handling Low Entry Clusters

As mentioned previously in this report, in my implementation I decided to simply not include all clusters with a low number of entries, in my clustering algorithm. Another approach I can look into however, is instead of removing these clusters, when a data entry is being assigned to one of these clusters, a point which is most farthest away from any other centroid can be chosen to be that entries nearest centroid. This may not be that effective however, it will incorporate more of the data. Another approach in handling low entry clusters, is to combine neighboring clusters into one. Since they were neighbors their attributes should not differ dramatically and so an assignment to this cluster can be fairly accurate.

15 CONCLUSION

In conclusion, this report gives a detailed description about the methodology and techniques I have used in the implementation of my project by providing justifications and reasons for choosing them. The report gives, an overview on *International Telematics* and where my project fits into their current work and why the need for this project arose. This report explains the advantages and disadvantages of how different data mining techniques can be used to build prediction models, with example applications illustrated from my related work research. This report also gives an outline of the chosen clustering algorithm I have used and how I have implemented the algorithm into my application. After building the initial application and performing various evaluation techniques, it was fairly evident that this initial application was not implemented to its most effective manner. This report provides an overview of this evaluation and describes the improvements I have made to deal with the limitations of the initial application. After implementing the various improvements into the application, performing the evaluation techniques once again proved to show that the improvements were in fact successful. A description of why these improvements were beneficial is explained in depth in this report. Finally, the last section of this report outlines a few issues with the current implementation of my application and provides some knowledge on the future work which can be done on my application to overcome these issues to better my final implementation.

16 REFERENCES

[1] Son, Bongsoo, et al. (2004). Bus arrival time prediction method for its application. *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin Heidelberg.

[2] Chaovalit, P., & Zhou, L. (2005, January). Movie review mining: A comparison between supervised and unsupervised classification approaches. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS'05.* (pp. 112c-112c). IEEE.

[3] Horvitz, Eric. Predictive Analytics for Traffic. Microsoft Research.

[4] Kennedy, Ruby L., et al. (1997). Solving Data Mining Problems Using Pattern Recognition Software with Cdrom. Unica Technologies Inc.

[5] Kim, T. H., & Yang, S. B. (2005). An effective recommendation algorithm for clustering-based recommender systems. In AI 2005: Advances in Artificial Intelligence (pp. 1150-1153). Springer Berlin Heidelberg.

[6] Mirkin, Boris. (2005). Clustering For Data Mining: A Data Recovery Approach. Chapman & Hall/Crc Computer Science.

[7] Pang-Ning, T., Steinbach, M., & Kumar, V. (2006). Introduction to data mining. In *Library of Congress*.

[8] Steyerberg, E. W. (2009). Clinical prediction models. Springer.

[9] Zhang, F., Liu, H., & Chao, J. (2010). A Two-stage Recommendation Algorithm Based on K-means Clustering. *In Mobile E-commerce. Journal of Computational Information Systems*, 6(10), 3327-3334.