BTECH 451: MID-YEAR REPORT

INTERNATIONAL TELEMATICS LTD.

Nizamuddin Shaik
5695040
nsh726@aucklanduni.ac.nz

1 TABLE OF CONTENTS

2 ABSTRACT

This report gives a detailed overview of the current progress in my BTECH 451 project. For my project I am working with the International Telematics company and the project involves in an investigation and analysis of location based services. In the first section 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 the second section, you will find an outline of the project goals and technologies that were used for the development in this project. The following section explains my initial research into the different data mining techniques and also explains the choice I made for this project. The next section provides a basic overview of the standard version of the algorithm I have decided to use in my implementation, the *k-means* clustering algorithm. The following section discusses my research into similar areas of research that involves the concepts used in this project. Then the next two sections give a detailed explanation of the techniques I have used in my implementation, and

explained screenshots of my current application in progress as well as showing its limitations. Finally, the last section of this report provides some information on the future work of my project.

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 designing 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 which will allow 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 in developing 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 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 s*upervised* 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 within 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:[3]

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 $\{x_1, x_2, \ldots x_n\}$ 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 in 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:

Cluster 1					
Name	Blood	Skin	Legs	Aquatic	Hibernates
Human	Warm	Hair	Yes	No	No
Whale	Warm	Hair	No	Yes	No
Bat	Warm	Hair	Yes	No	Yes
Cluster 2					
Name	Blood	Skin	Legs	Aquatic	Hibernates
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					
Name	Blood	Skin	Legs	Aquatic	Hibernates
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

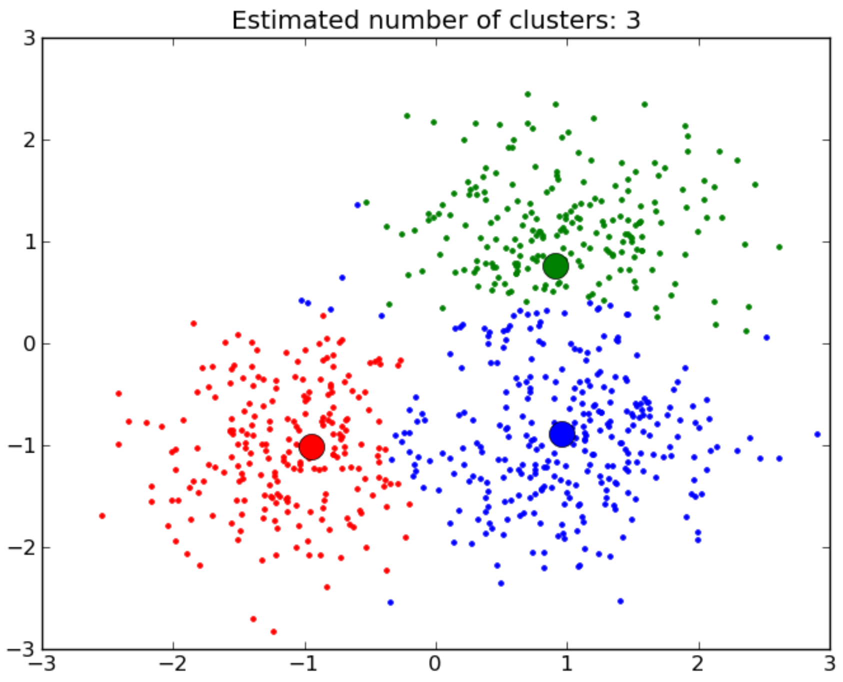
From here it can be seen that the initial input dataset has been grouped into four different clusters. What also can be seen is that within each of these clusters the vertebrate within them seem to have similar attributes with each other, specifically in terms of the "blood" and "skin" categories. Now 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 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. 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.

This is the basic procedure of how the standard *k-means* clustering algorithm performs its clustering task. The next section of the 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*.[1] 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.[1] 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.[1] 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*[7], 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*[3], 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*[6], 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 IMPLEMENTATION

After I had familiarized myself with a sound understanding of the different techniques used in prediction models, I decided to begin my implementation. For my implementation of the *k-means* clustering algorithm, I first made an analysis on the sample data in 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 7:50	1/6/13 8:01	689
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 the method and an explanation of the techniques I have used in my implementation.

8.1 Method

8.1.1 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. 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.

8.1.2 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.

8.1.3 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.

8.2 Techniques

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

8.2.1 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,

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

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 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, 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.

8.2.2 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:

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

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.

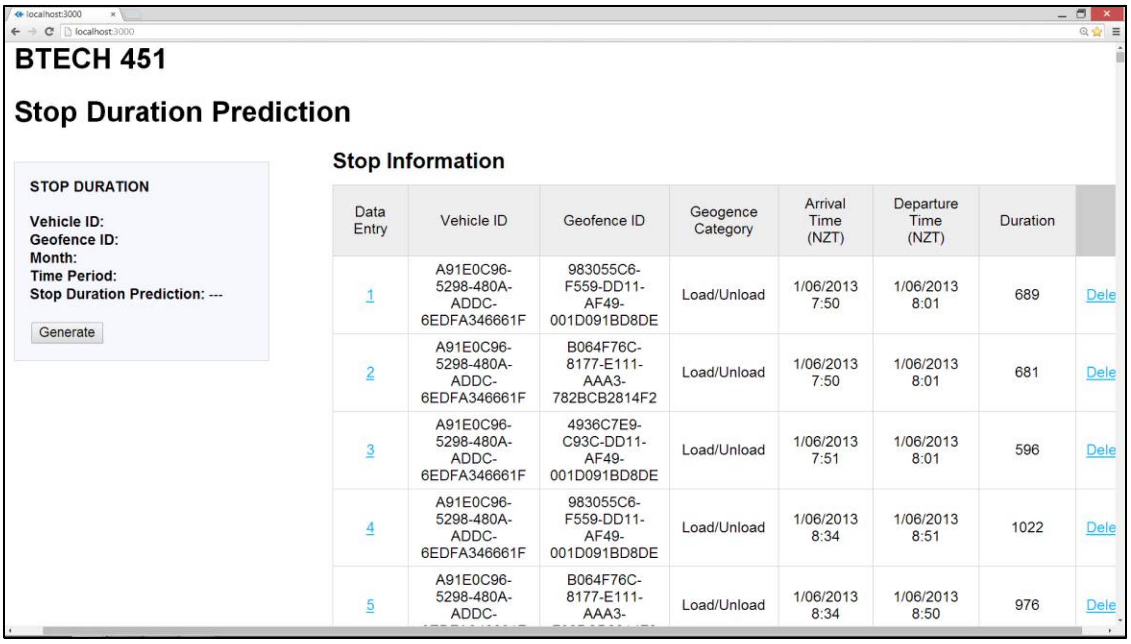
This measure is done for each of the five clusters within their respected geofence partitions. The results of this is seen in the next section.

9 THE APPLICATION

The following section of this report, gives an overview of how I decided to build my application. It provides screenshots of my working application, an explanation on the design decisions I made, and then I describe the current limitations in progress of my application.

9.1 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, 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 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

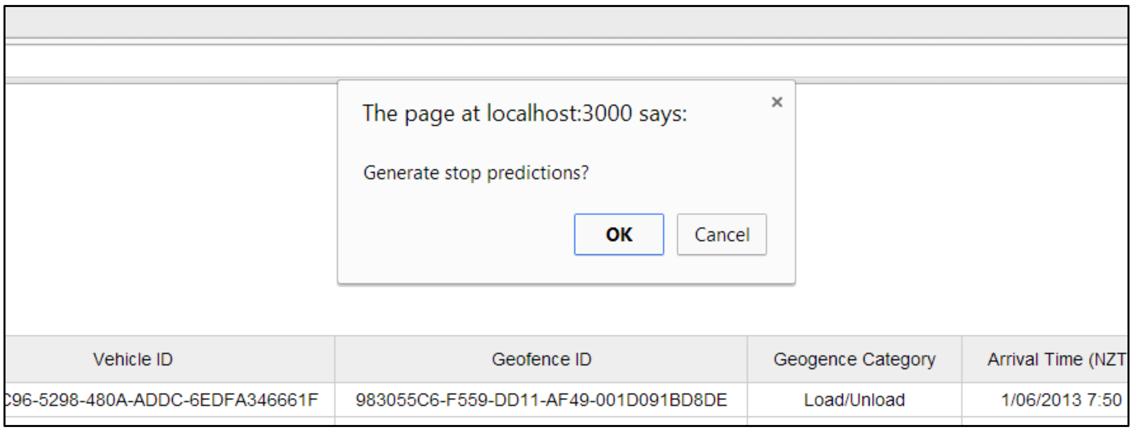
Vehicle ID:
Geofence ID:
Month:
Time Period:
Stop Duration Prediction: ---

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

9.2 Displaying Results

The next step in for my application was to decide how I was going to display the many different results that are obtained throughout my implementation. As this is still work in progress and the fact that these results are 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:

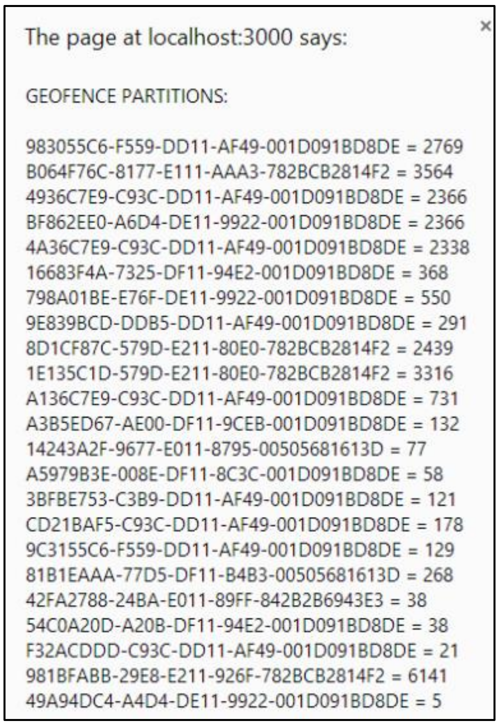


The page at localhost:3000 says:

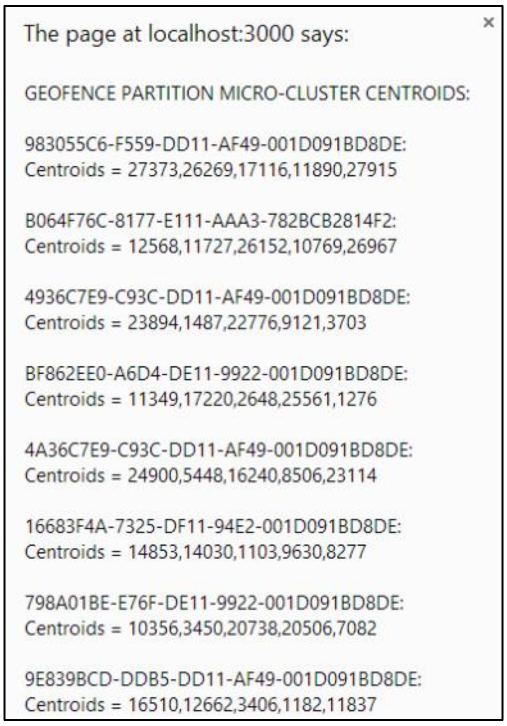
Generate stop predictions?

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.

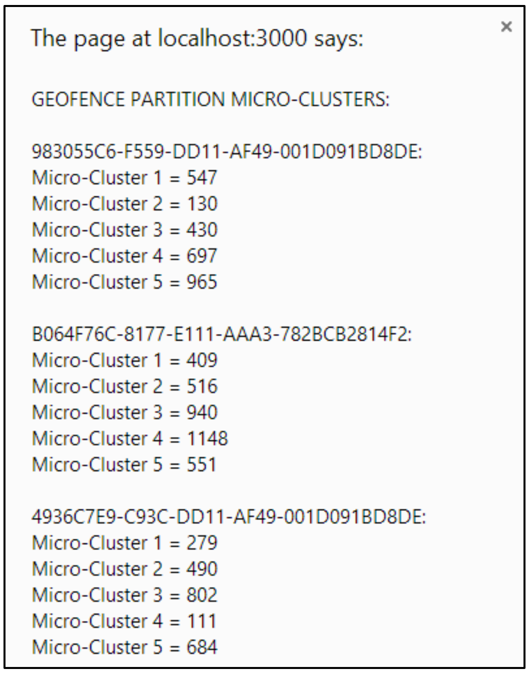


```
The page at localhost:3000 says:

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
```

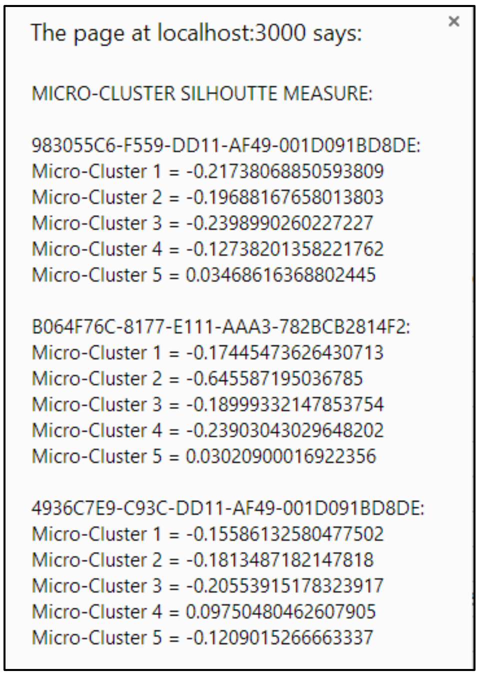


```
The page at localhost:3000 says:

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. As mentioned previously in this report, a negative silhouette value indicates inappropriate clustering and a positive silhouette indicated appropriate clustering. As it can be seen by the screenshot most of my obtained clusters are producing negative values, with the exception of a few clusters. This is the current limitation of my application and implementation and will be discussed in the following sub-section.

9.3 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. 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 has made me believe that there are more factors that are influencing my clustering algorithm, that I will need to consider in the future. These factors are explained in the next section of this report.

10 FUTURE WORK

The current stage of progress I am up to in the development of my application is the stage of where I am assigning each data entry within each geofence partition to its nearest centroid to form the clusters. The next steps of my application development are outlined in this section.

10.1 Increasing Cluster Accuracy

As explained in the previous section, the accuracy of my final clusters is very limited. Before I move on to the next steps of my implementation, I will be needing to make sure my clusters are 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 will need to look deeper into my implementation to find the factors that are influencing poor clusters. One possible factor can include the involvement of a number of outliers within the dataset. Even thou 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 outlier, so my next action could be to try my implementation after removing this outliers that outside the bounds of a set threshold. Another method of increasing the accuracy of my clusters could be to increase the number of runs my algorithm takes with also tinkering around with the number of clusters per partition. The advantages of this have been explained previously in this report.

10.2 Final Prediction and Evaluation

After I have obtained accurate enough clusters, I will move on to the next stage of my implementation, where the extraction of the final prediction result happens. To do this I will first look into the different ways in which this can be done and from there I will choose the best method that is suitable for my application. Then after that, I will need to evaluate once again, the accuracy of this final prediction result. A final prediction evaluation method such as the *10-fold cross validation* technique can be used to do this. This method basically works by dividing the dataset into 10 divisions and then taking one division into the prediction model, and then comparing the outputted prediction result with the other 9 divisions. More on this method will be explained in the future.

11 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; 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; an outline of the chosen clustering algorithm I have used and also how I have implemented the algorithm into my application. Also shown in this report, is the current progress of my application and the limitations it now possesses. It shows that there is still a lot of work to be done to, not only overcome these limitations but, also to further my application to its final stage.

12 REFERENCES

[1] 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.

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

[3] 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.

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

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

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

[7] 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.