Using Cluster ensembles for detection and addressing of Concept drifts in data streams

Project progress report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Computer and Communication Engineering

By

Sriharsha Daparti
Reg. No. 140953194

Visal Kancharla
Reg. No. 140953148

Under the guidance of

Mr. Nirmal Kumar Nigam
Assistant Professor – Senior Scale
Department of I & CT
Manipal Institute of Technology
Manipal, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

May 2018

I dedicate my thesis to my friends and family.

DECLARATION

I hereby declare that this project work entitled Title of your project is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of your guide name, Guide's designation, Department of Information and Communication Technology, M. I. T., Manipal. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Manipal

Date: 04-05-18

Sriharsha Daparti

Visal Kancharla

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that the project titled **Using Cluster Ensembles for detecting and addressing of Concept Drift in data streams** is a record of the bonafide work done by **SriHarsha Daparti** (*Reg. No. 140953194*) at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Computer and Communication Engineering.

Mr. Nirmal Kumar Nigam
Assistant Professor – Senior Scale
Department of I & CT
Manipal Institute of Technology
Manipal, India

Dr. Balachandra
Professor & Head
Department of I & CT
Manipal Institute of Technology
Manipal, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that the project titled **Using Cluster Ensembles for detecting and addressing of Concept Drift in data streams** is a record of the bonafide work done by **Visal Kancharla** (*Reg. No. 140953148*) at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Computer and Communication Engineering.

Mr. Nirmal Kumar Nigam
Assistant Professor – Senior Scale
Department of I & CT
Manipal Institute of Technology
Manipal, India

Dr. Balachandra
Professor & Head
Department of I & CT
Manipal Institute of Technology
Manipal, India

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Mr. Nirmal Kumar Nigam for the continuous support of my project and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and project. I could not have imagined having a better advisor and mentor.

Besides my advisor, I would like to thank the Head of Department, Information & Communication Technology, Prof. Dr. Balachandra for providing with resources without which the project would not have been possible.

ABSTRACT

The scalability of data mining methods is constantly being challenged by real-time production systems that generate tremendous amount of data at unprecedented rates. They are having a huge volume of data. The underlying data generating mechanism, or the information that we try to learn from the data, is constantly evolving. In order to make time-critical predictions, the model learned from the streaming data must be able to capture up-to-date trends and transient patterns in the stream. The challenges presented by concept drifts in streaming data can be handled by adapting cluster ensembles.

[Mathematics of Computing]: Probability and statistics-Statistical paradigms-Exploratory data analysis
[Computing Methodologies]: Machine learning-Learning paradigms-Unsupervised learning-Cluster analysis; Machine learning algorithms –Ensemble methods-Stacking

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1 INTRODUCTION

In the present world, the source and structure of data are constantly evolving. Such change presents a great difficulty during data mining and predictive analysis. To handle this, newer methods and techniques associated with data extraction and analysis are being developed.

1.1 Data & Information

Data is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized.

When data is processed, organized, structured or presented in a given context to make it useful, it is called information. So by this fact we can say that data is used as input for the computer system and information is the output of data. While data is unprocessed facts or figures but information is the processed data, also data does not depend on information but information depends on data, data is not specific whereas information is specific, information must carry a logical meaning

But data doesn't carry a meaning. Data is a single unit but the group of data with a meaning makes information. Data is the raw material whereas the product is the information.

Example: Market data might include the details of 1.2 billion trades that occurred over a ten-year period for a large company's stock. Such data is summarized in an informative graph that shows the price over time. People can instantly understand information in a graph of the historical price of a stock. The raw data used to create such graphs is prohibitively large to be useful to a person.

1.2 Static Data Vs Streaming(Dynamic) Data

Static data is the data that does not change. Or you can say that it is not real-time.

Static Data is self-contained or controlled.

Example: When you fill up a form then you select your city from a defined list of cities. The list does not change (generally). Therefore, we can say that the data collected after filling your form is static data.

You may require some cleansing, preparation, and preprocessing before you can use static data for an analysis.

Streaming data is all about real-time data. The data collected from various sensors, web feeds, etc. constitute this type of data.

Example: Suppose you install a health app in your phone, which collects information on your health and sends it to a server that may be accessed by a hospital. Therefore, you can imagine the amount of varied data flooding into the server every minute. The data is constant and relentlessly changing.

In processing of the stream data the summary of the data is only stored by discarding the processed data which makes the processing better rather than a tedious task that requires more resources.

1.3 Features of Data Streams

The following are the distinct features of the data streams,

- Huge volumes of continuous data, possibly infinite
- Fast changing and requires fast, real-time data
- Data stream captures our day to day data processing needs very well
- Random access is expensive – single scan algorithm (can only have one look)
- Store only the summary of the data seen thus far
- Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing

1.4 The Choice: Data Streams

Rather than storing the whole data as seen in static data examples, streaming data has the concept of storing only the summary required for processing and discarding the rest of it. This reduces the storage space usage by huge limits. Whenever the data stream comes in, the summary is updated according to need. In some cases, a part of the data is stored to observe certain trends that cannot be deducted by the summary[6].
This is known as the Windowing or the Sliding Window[3] concept. A sample of the data is stored in a buffer created to grab the data stream temporarily until the next time of change. This is done using the Forgetting mechanism.

1.5 Concept Drift

Data streams are also characterized by drifting concepts. Concept drift means that the properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes. Handling these data drifts is necessary to keep the data analysis efficient.

1.6 Clustering and Cluster Ensembles

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).
While working on data which requires unsupervised learning, the best way to handle it is to use clustering techniques.

Cluster ensembles[1][8] provide a framework for combining multiple base clustering of a dataset to generate a stable and robust consensus clustering.
Ensemble is used to increase the measure of ability to predict and analyze the data.
When clustering algorithms are used on their own, they may not be able to handle the given data in the best way. Therefore, by analyzing the data, we can compose and design a cluster ensemble which fits better and predicts much more accurately for that data.

2 OBJECTIVE

- *To detect Concept drifts in a streaming environment using Cluster Ensembles*

3 CLUSTERING USING DATA STREAMS

There are two ways to handle data

- Supervised Learning,
- Unsupervised Learning

Here the emphasis is on unsupervised learning methods, clustering in particular.
Data stream clustering is defined as the clustering of data that arrive continuously such as telephone records, multimedia data, financial transactions etc. Data stream clustering is usually studied as a streaming algorithm and the objective is, given a sequence of points, to construct a good clustering of the stream, using a small amount of memory and time**.**

Types of Clustering Methods:
The different types of clustering approaches are discussed below:
Partitioning methods:

- These methods find mutually exclusive clusters of spherical shape.
- They are distance-based.
- These methods may use mean or medoid (etc.) to represent a cluster center.
- These methods are effective for small- to medium-size data sets

Hierarchical methods:

- These methods employ a hierarchical decomposition (i.e., multiple levels)
- These cannot correct erroneous merges or splits
- They may incorporate other techniques like micro clustering or consider object “linkages”

Density-based methods:

- These clusters can find arbitrarily shaped clusters.
- They produce clusters that are dense regions of objects in space that are separated by low-density regions

- Cluster density: Each point must have a minimum number of points within its neighborhood.
- These methods may filter out outliers.

Grid-based methods:

- These methods use a multiresolution grid data structure
- They have Fast processing time (typically independent of the number of data objects, yet dependent on grid size).

The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters. Partition methods are relatively scalable and simple. Suitable for datasets with compact spherical clusters that are well separated. The most well known and commonly used partitioning method is k-means. K-means[1][2] is a simple and easy to run efficient algorithm, it works well with large datasets, results obtained are easy to interpret, has less computational costs.

There are different variants of K-means, few of them are

- **K-medians** is a variation of k-means where in place of calculating mean to compute centroid of the cluster, median is computed.
- **K-medoids** this method chooses an object per cluster as representative of that cluster and rest of the objects are assigned to cluster on the basis of their similarity with the representative object. Then partitioning is performed on the basis of dissimilarities between rest of the objects and the representative objects. The algorithm iterates until each representative object actually becomes medoid or is the central most object of the cluster.
- **K-modes** is a technique for clustering categorical data. K-modes modifies k-means by replacing Euclidean distance metric with simple matching dissimilarity measure, using modes to represent cluster centers and updating modes with the most frequent categorical values in each iteration of clustering process.
- Variants based on different distance metrics, which use other distance metrics like Manhattan, least Squares, etc., unlike the Euclidean in the standard K-means .

Clara Clustering algorithm is another prominent partition based method of clustering, It is a sampling based method for large datasets which is used instead of K-medoids. Instead of taking the whole data set into consideration, CLARA uses a random sample of the data set. The PAM(partition around medoids) algorithm is then applied to compute the best medoids from the sample.

The following algorithms deal with the real time streaming data:

- STREAM Clustering
- CluStream Clustering
- Massive-Domain Stream Clustering

However, if the data is not analysed in a dynamic fashion, there is no need for the above algorithms. A k-means algorithm or any of its fitting variant will suffice.

Many clustering algorithms are available for data stream, which uses k-means algorithm as a base. Even in the case of data streams the data has to be analysed instantaneously. This makes data streams look like static datasets at that instance. As k-means works with ease of implementation, simplicity, efficiency, and empirical success for static data the same applies to dynamic data.

4 CLUSTERING DATA STREAMS

Along with the objective to cluster the data and to apply ensemble learners on the data, we also wanted to detect concept drifts and address them if possible. As previously stated, most of the clustering on data streams was done using partitioning methods. The reason behind primarily working on partitioning algorithms is that they give better outliers. For example, in case of CLARA, the outliers can be seen out of the clusters' sample space and in case of DBSCAN, noise points are detected.

These outliers are nothing but the data that does not fit into a regular or a pattern specific model. Detection of such an instance means that the pattern of the data has changed. If this change happens we can say that a drift in data (or Concept Drift) has occurred. Therefore, to achieve this, partitioning methods are chosen.

5 CONCEPT DRIFTS IN DATA STREAMS

Different from traditional stationary database, a data stream[3] poses distinct challenges due to its dynamic nature, one pass scan requirement and rigorous memory limitation. An emerging and attractive challenge there into is the drifting concept, or known as concept drift[6], which refers to the time-changing property of the underlying distribution in data stream. For example, a network intrusion, the changing weather conditions, or a user's new interest. Concept drift is a thorny issue, which, unfortunately, occurs commonly in real world. It usually leads to the expiration of patterns learned from the up-to-date examples. It is highly provable that examples of data stream are generated from a sequence of concepts instead of from a stable concept. Therefore, approaches being capable of addressing data stream concept drift are essential and meaningful.

Data streams are not only known for their vastness and continuous flow, but are also characterized by their drifting concepts[5]. Concept drift means that the properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes. Handling these data drifts is necessary to keep the data analysis efficient.

Case: The behavior of the customers in an online shop may change over time. For example, if weekly merchandise sales are to be predicted, and a predictive model has been developed that works satisfactorily. The model may use inputs such as the amount of money spent on advertising, promotions being run, and other metrics that may affect sales. The model is likely to become less and less accurate over time – this is concept drift. Apart from the online sales predictor, concept drift can be observed during weather changes or a bank fraud.

Data streams can be processed in two ways: incrementally one instance at a time or data stream is divided into blocks of equal size and all instances of a block are processed at a time to update classifier[6]. The former technique is called online classifier, while later is a block-based classifier

.

Data drift make the learning algorithms to give the results which are not in agreement with the incoming data. The learning algorithms need to adapt to the changes in the data that is changing. for instance, in the sales prediction application example we discussed above, concept drift might be compensated by adding information about the season to the model. By providing information about the time of the year, the rate of deterioration of your model is likely to decrease, concept drift is unlikely to be eliminated altogether. This is because actual shopping behavior does not follow any static, finite model. New factors may arise at any time that influence shopping behavior, the influence of the known factors or their interactions may change such as socioeconomic processes, and biological processes.

Multiple methods for detecting concept drifts in data streams have been proposed by research over years. These techniques can be categorized as windowing technique, ensemble classifiers, and drift detectors. Windowing techniques provide a simple mechanism that helps to keep most recent data and to update classifier on it to maintain accuracy. But drift detection process is hampered by the size of the window used. Ensemble classifier provides the advantage of modifying classifier structure or aggregation methods when the performance of base classifier degrades due to drifts. Drift detectors keep track of classifier performance and maintain triggering mechanism that is activated to alert about possible drift and rebuilding classifier.

As said earlier in this chapter, concept drifts are posed by data streams. But, due to the lack of a live data stream, the static data had to be manipulated with, and ensemble clustering was applied in order to observe the change in the data and address it accordingly.

6 CLUSTER ENSEMBLES

Clustering ensembles combine multiple partitions of the given data into a single clustering solution of better quality**.** Clustering ensembles can offer better solutions in terms of robustness, novelty and stability. Combination of clustering's is a more challenging task than combination of supervised classifications. In the absence of labeled training data, we face a difficult correspondence problem between cluster labels in different partitions of an ensemble.
Ensemble techniques[6] can be applied for Classification, Clustering and regression.They include:

Bayes optimal classifier:
The Bayes Optimal Classifier is a classification technique. It is an ensemble of all the hypotheses in the hypothesis space. Naive Bayes Optimal Classifier is a version of this that assumes that the data is conditionally independent on the class and makes the computation more feasible. Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true. To facilitate training data of finite size, the vote of each hypothesis is also multiplied by the prior probability of that hypothesis.

Bootstrap aggregating (bagging):
Bootstrap aggregating, often abbreviated as bagging, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set.

Boosting:
Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data.

Bayesian parameter averaging:
Bayesian parameter averaging (BPA) is an ensemble technique that seeks to approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space, and combining them using Bayes' law. Unlike the Bayes optimal classifier, Bayesian model averaging (BMA) can be practically implemented.

Bayesian model combination:
Bayesian model combination (BMC) is an algorithmic correction to Bayesian model averaging (BMA). Instead of sampling each model in the ensemble individually, it samples from the space of possible ensembles.

Bucket of models:
A "bucket of models" is an ensemble technique in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, a bucket of models can produce no better results than the best model in the set, but when evaluated across many problems, it will typically produce much better results, on average, than any model in the set. The most common approach used for model-selection is cross-validation selection. Cross-Validation Selection can be summarized as: "try them all with the training set, and pick the one that works best".

Stacking:
Stacking involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although, in practice, a logistic regression model is often used as the combiner.

Different clustering algorithms may produce different partitions because they impose different structure on the data. No single clustering algorithm is optimal. The features of ensemble clustering that motivated for using a cluster ensemble more than a classifier ensemble or a regression ensemble are:

Improved quality of solution. Just as ensemble learning has been proved to be more useful compared to single-model solutions for classification and regression problems, one may expect that cluster ensembles will improve the quality of results as compared to a single clustering solution. It has been shown that using cluster ensembles leads to more accurate results on average as the ensemble approach takes into account the biases of individual solutions.

Robust clustering. It is well known that the popular clustering algorithms often fail spectacularly for certain datasets that do not match well with the modeling assumptions. A cluster ensemble approach can provide a ‘meta’ clustering model that is much more robust in the sense of being able to provide good results across a very wide range of datasets.

Model selection. Cluster ensembles provide a novel approach to the model selection problem by considering the match across the base solutions to determine the final number of clusters to be obtained.

Knowledge reuse. In certain applications, domain knowledge in the form of a variety of clusterings of the objects under consideration may already exist due to past projects. A consensus solution can integrate such information to get amore consolidated clustering.

Multiview clustering. Often the objects to be clustered have multiple aspects or ‘views’, and base clusterings may be built on distinct views that involve non identical sets of features or subsets of data points.

This may be a rare instance but Distributed computing. In certain situations, data is inherently distributed and it is not possible to first collect the entire data at a central site due to privacy/ownership issues or computational, bandwidth and storage costs. An ensemble can be used in situations where each clusterer has access only to a subset of the features of each object, as well as where each clusterer has access only to a subset of the objects.

Cluster ensembles, in some cases, also refers to Consensus clustering. It is said that Consensus clustering can provide benefits beyond what a single clustering algorithm can achieve. Consensus clustering algorithms often: generate better clusterings; find a combined clustering unattainable by any single clustering algorithm; are less sensitive to noise, outliers or sample variations; and are able to integrate solutions from multiple distributed sources of data or attributes. As an example, consensus clustering can be employed in "privacy-preserving" scenarios where it is not possible to centrally collect all of the underlying features for all data points, but only how the data points are grouped together.

As efficient and better it may seem, consensus clustering[8] is not the best fit to our scenario. The objective of this experiment and comparison is to formulate an ensemble which will aid in detecting concept drifts in the selected data (static/streaming). We have established earlier that concept drifts can be detected if outliers and noise elements are distinguished. If the consensus clustering algorithm 'has low sensitivity towards outliers, noise and sample variations', the detection of concept drift seems difficult. So we have decided to work with regular clustering algorithms which fit better to the selected dataset and apply ensembling techniques (stacking is predominantly used) to improve them.

The ensemble technique that was deemed better for the project was stacking. The reason to choose stacking was explained better in Chapter 7. Stacking involves feeding the clustered output of one algorithm to another. We have used k-means, CLARA and HDBSCAN for clustering. First, the algorithms were tested independently and then we have formulated the ensembles to know how they work in comparison with the independent computations.

7 UNDERLYNING TECHNOLOGY

This project has a very basic and feasible way of implementation.
The technological resources (both hardware and software) used for this project are:

1. R-Studio with R version 3.4.3 or higher.
2. MySQL database version 5.7
3. RAM : 8GB or higher

8 METHODOLOGY

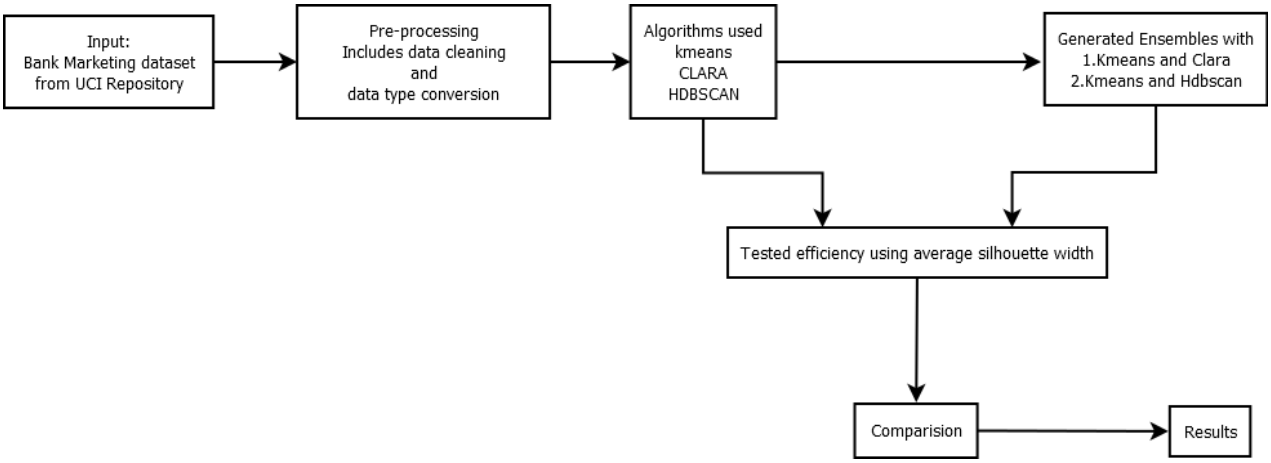

Figure 1.Diagram illustrating the methodology followed

- The dataset is taken from the renowned UCI data repository. The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. This dataset is Multivariate with real attribute characteristics.
- The data, being huge, was incompetent to work on. Therefore, the whole data set had to be stored in a database and fetched from it according to the requirement.
- Once the data is fetched from the database and stored in a data frame, it is evident that the attributes which are to be subjected to clustering are in a mixed fashion i.e., a mixture of categorical and numerical attributes[2]. For most of the clustering algorithms, it is a drawback that they cannot handle this kind of a data. To make this possible the data types of all those categorical data attributes are changed to numerical using data pre-processing tools.
- R-language is the medium adopted for data processing and algorithms implementation.
- After testing several algorithms like k'-means, fuzzy c-means[1] , k-medoids, k-modes, DBSCAN, etc., we finally selected the k-means, CLARA and hierarchical DBSCAN. The reasons to choose these algorithms:
    1. k-means is the most efficient partitioning algorithm and has higher computational ability. For a dataset with higher number of attributes like the above mentioned one, partitioning algorithms are the most preferred to apply.
    2. CLARA is also a partitioning algorithm but with a different approach. This uses the concept of 'Partition around medoids' (PAM) to compute medoids and thus reduce the average dissimilarity of objects to their closest selected objects.
    3. We have also tried the conventional DBSCAN algorithm which is also a partitioning algorithm with a density based clustering approach. But a known distance measure ε (eps) was needed to cluster. So instead of using three partitioning methods we have used the HDBSCAN algorithm which uses a hierarchical clustering approach.
- We have considered and implemented the concept of ensemble learning. We have tried to use boosting techniques on each of the algorithms and give weights to crucial

attributes. But that was very unpredictable due to the large number of attributes. So we have tried a stacking approach where-in we combine outputs from different learning models.

Here we have clustered the data using k-means first by stacking Euclidean k-means on Manhattan k-means and vice-versa. Then, in the same manner, a combination of k-means and CLARA was computed as we have used a large dataset. Also to show some difference from partitioning methods to hierarchical methods, we have brought together k-means and HDBSCAN.

- Towards the validation[4] phase of the project and ensembles computed above, we have used the Silhouette co-efficient. This is explained in detail in the nest chapter.

9 EXPERIMENTS AND RESULTS

The earlier phases of this project did not have a specific face or a body to it i.e., neither a specific algorithm nor a dataset was fixed to work on. The experiments were carried out on generic datasets like 'iris'. The initial experiments were done using algorithms like k-means, k'-means – which is a modification to k-means where the optimal number of clusters is calculated initially using he 'elbow method', fuzzy c-means – where it was assumed that every data instance belongs to every cluster and then further clustered using a nearness factor.

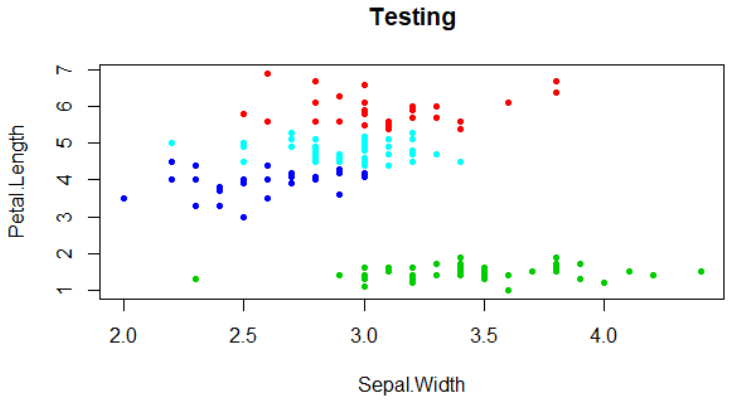

Figure 2. kmeans on Iris dataset - plot between Petal Length and Sepal Width

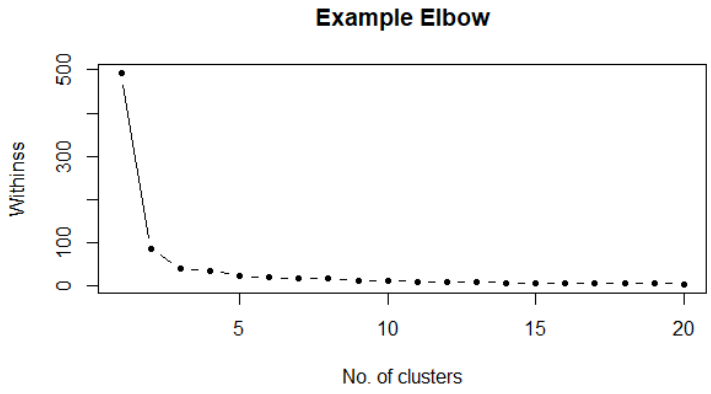

Figure 3. Elbow Graph

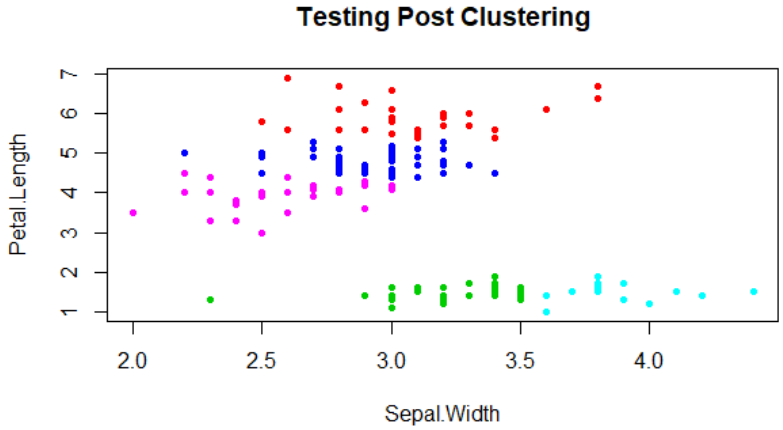

Figure 4. k'-means on iris dataset

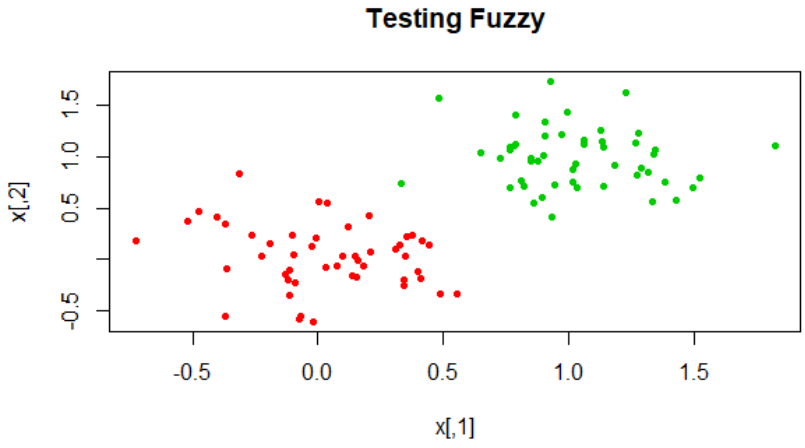

Figure 5. Fuzzy c-means on a randomly generated dataset

Finally, after acquisition of the dataset, suitable algorithms were selected to work on that dataset. As the selected dataset had about 17 attributes, which were both numerical and categorical, the data had to go through pre-processing. Then the processed data was used

to perform clustering using individual algorithms. The clustering algorithms used are mentioned above in the methodology.

Given below are the samples of individual cluster plots for k-Means (*Figure 7.*), CLARA (*Figure 6.*) and HDBSCAN (*Figure 8.*):

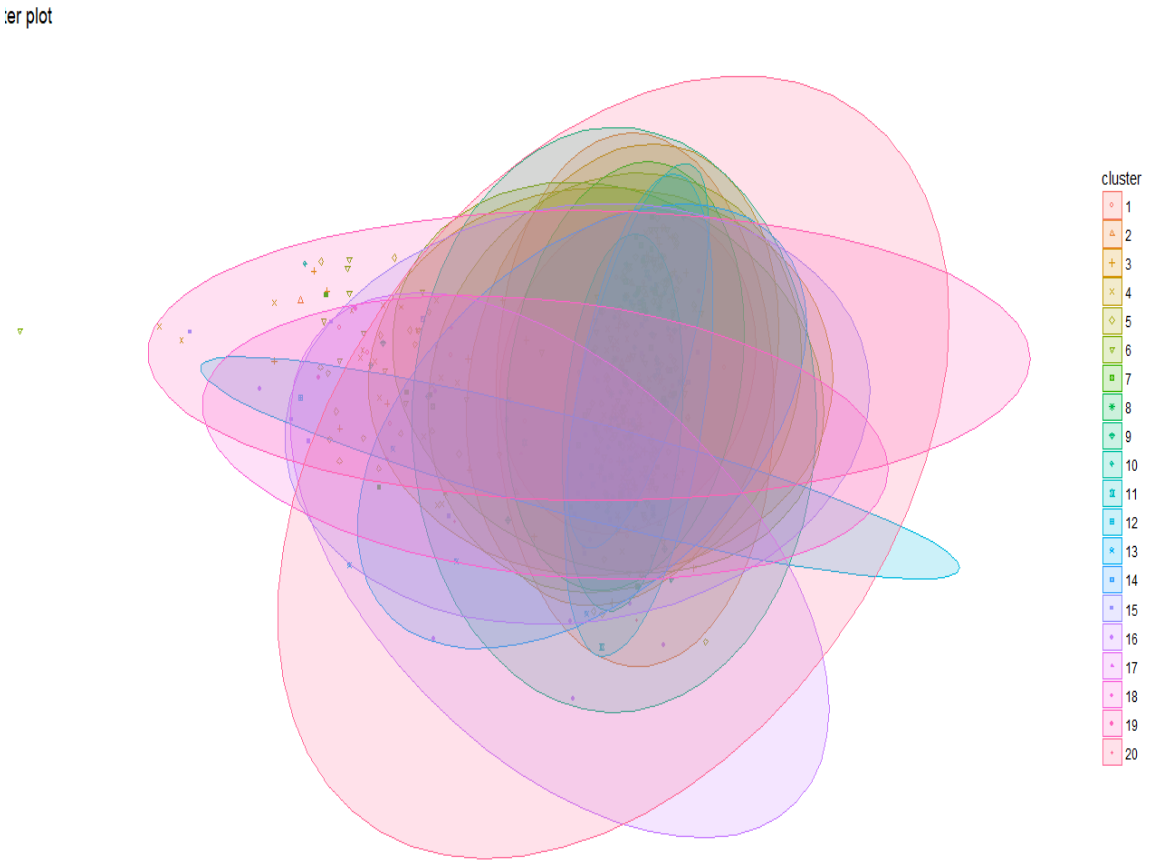

Figure 6. CLARA Clustering based on Manhattan metric

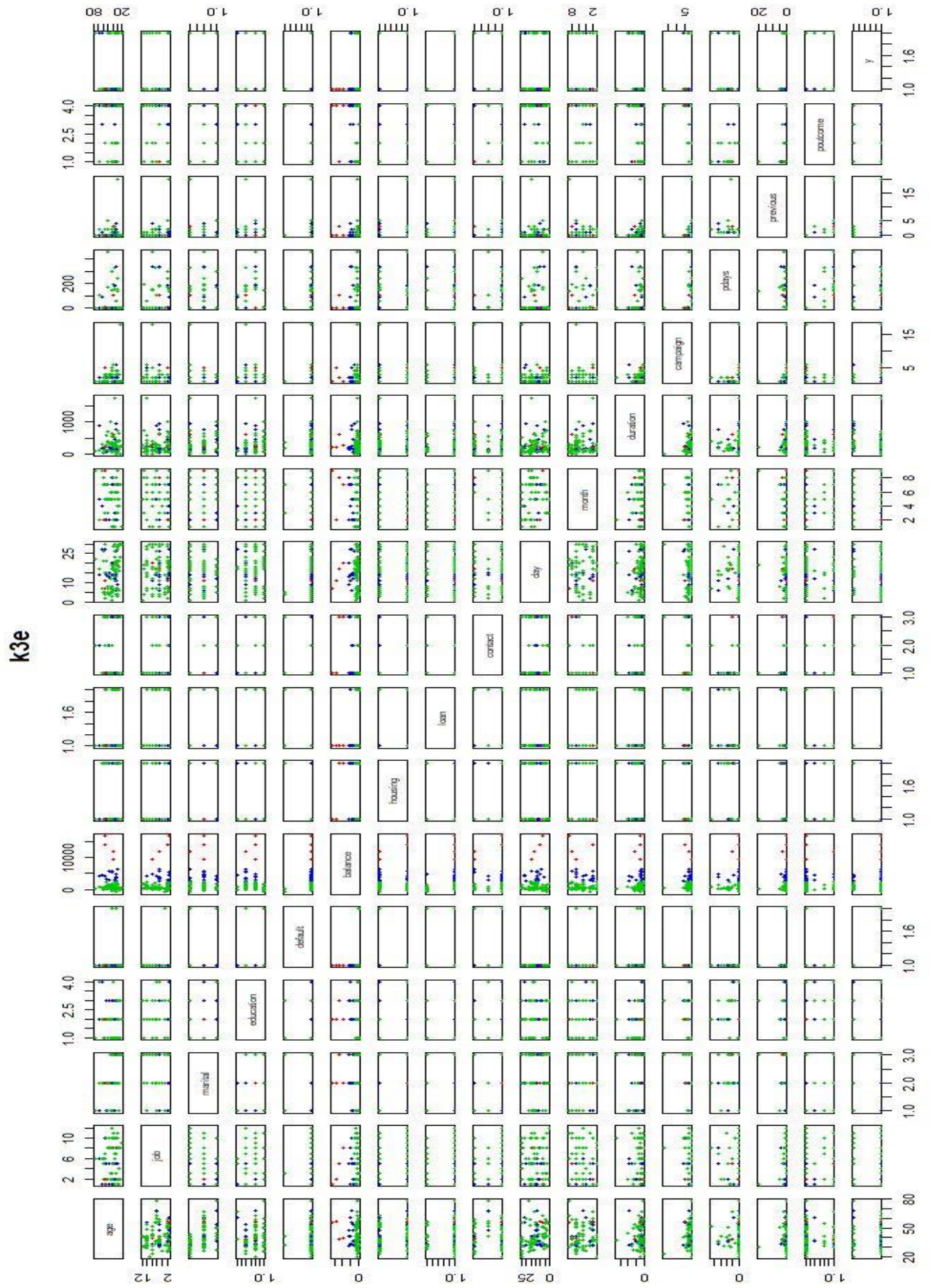

Figure 7. Euclidean k-Means with 3 clusters

Figure 8. HDBSCAN Clustering

Once the individual clusterings were done, their results were stacked onto other algorithms and ensembles were computed. The metric for comparison of how well the clusters have formed was Silhouette co-efficient[9].

In the case of cluster analysis, there are two internal measures: Cluster cohesion and Cluster separation. Cohesion measures the closeness of the object in its own cluster while Separation shows how distinct a cluster is from the others. Also known as Average Silhouette Width, Silhouette coefficient combines the idea of both cohesion and separation, but for individual points, as well as clusters and clusterings.

For a better understanding, a generic example is given below:
For an individual point, *i*
Let 'a' be the average distance of *i* to the points in its cluster
Let 'b' be the min (average distance of *i* to points in another cluster)
The silhouette coefficient is given by

$$S = \begin{cases} 1 - \frac{a}{b}, & if\ a < b \\ 1 - \frac{b}{a}, & if\ a \geq b (unusual) \end{cases}$$

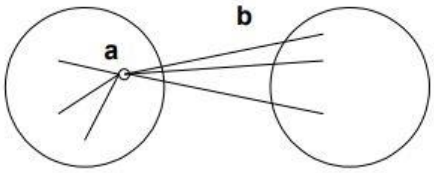

Figure 9. Computing Silhouette

The silhouette value generally fluctuates between 0 and 1, being better when closer to 1. Then the average silhouette width of all the cluster points was calculated to tell us how well a point was clustered. For example, The silhouette plot of k-means (independently) and the silhouette plot of an ensemble – CLARA on k-means are illustrated below:

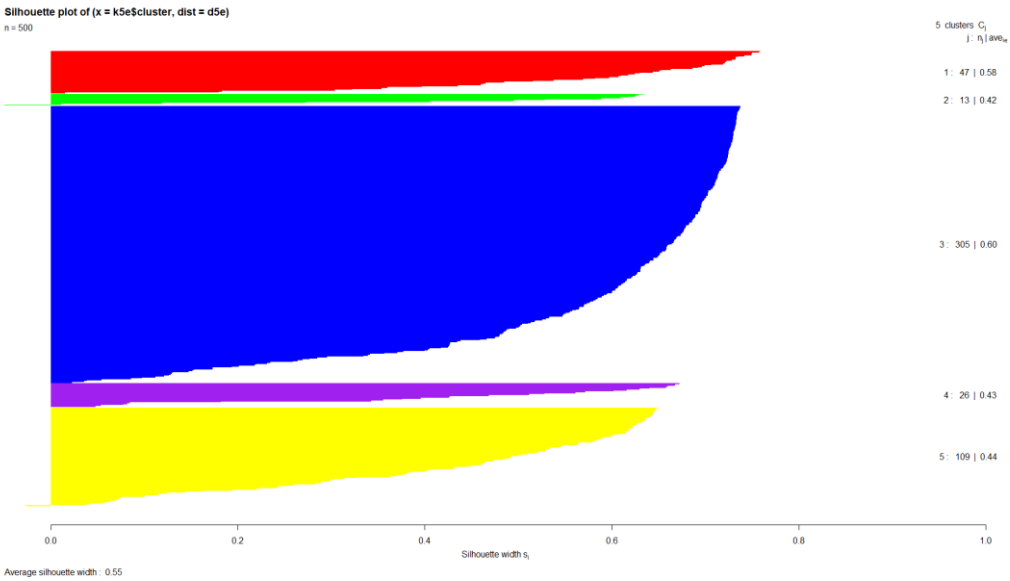

Figure 10. Silhouette plot of k-Means on the bank dataset with 500 instances

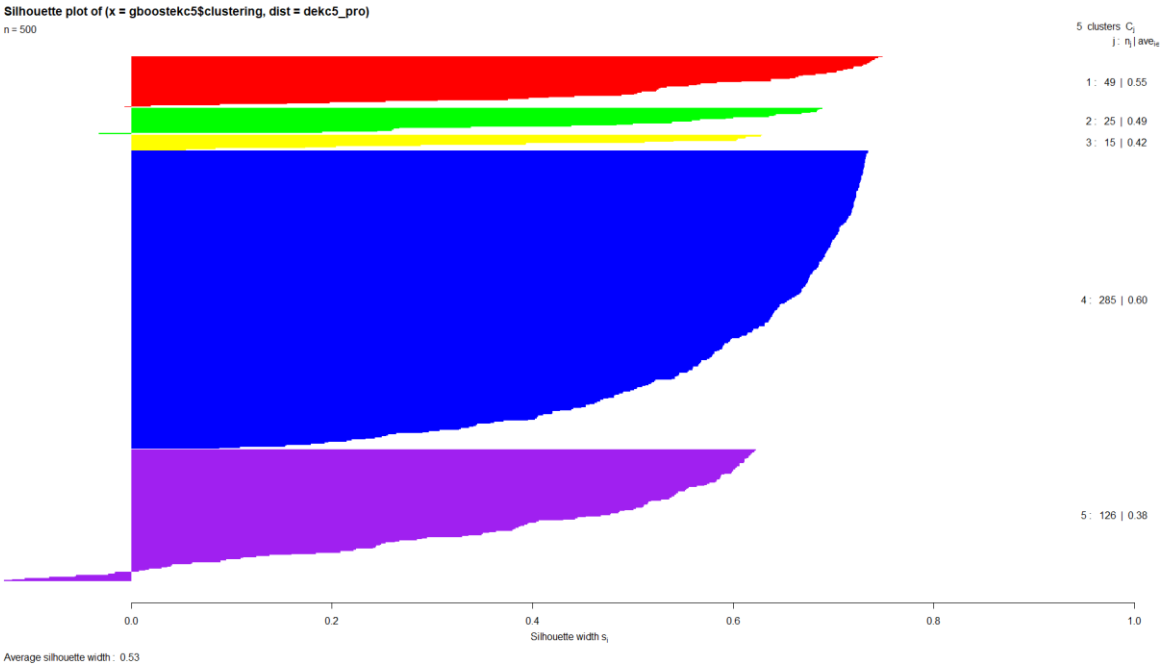

Figure 11. Silhouette plot of the Ensemble of CLARA on k-Means applied on bank dataset with 500 instances

In the same manner, we have computed the graphs and formulated results for several combinations. Here are the results:

Algorithm: k-Means
Dataset: Bank Telemarketing data
Number of attributes: 17 Number of instances: 100

Table 1. Table showing Clusters with various k value and distance metrics

No of Clusters(k)	Distance Measure	Cluster Sizes(no of data points /cluster)
2	Euclidean	96,4
2	Manhattan	4,96
2	Least Squares	8,92
3	Euclidean	4,82,14
3	Manhattan	4,82,14
3	Least Squares	4,82,14
4	Euclidean	66,10,3,21
4	Manhattan	20,3,70,7
4	Least Squares	4,64,22,10
5	Euclidean	12,58,7,4,19
5	Manhattan	9,18,59,4,10
5	Least Squares	10,2,22,2,64

Validation results: Phase I – 100 instances

Table 2. Comparison using Silhoutte - 1

K clusters with specified metrics*	Average Silhouette Width		
	Independent	**Self-Ensemble#**	**Cross Ensemble∞**
2 - Euclidean	0.84	0.83	0.72
2 - Manhattan	0.83	0.84	0.83
3 – Euclidean	0.72	0.69	-
3 – Manhattan	0.69	0.72	-
4 – Euclidean	0.61	0.58	-
4 – Manhattan	0.58	0.62	-
5 – Euclidean	0.50	0.37	-
5 – Manhattan	0.47	0.59	-

2 - CLARA – Euclidean	0.72	0.71	0.84
2 - CLARA - Manhattan	0.68	0.72	0.83

Validation results: Phase II – 500 instances

Table 3. Comparison using silhouette - 2

K clusters with specified metrics*	Average Silhouette Width		
	Independent	**Self-Ensemble#**	**Cross Ensemble∞**
5 – Euclidean	0.55	0.51	0.53
5 – Manhattan	0.51	0.55	0.36
10 – Euclidean	0.40	0.35	0.38
10 – Manhattan	0.37	0.38	0.34
20 – Euclidean	0.32	0.30	0.37
20 – Manhattan	0.34	0.35	0.29

5 - CLARA – Euclidean	0.53	0.38	0.55
5 - CLARA - Manhattan	0.37	0.47	0.51
10 - CLARA – Euclidean	0.37	0.34	0.40
10 - CLARA - Manhattan	0.33	0.37	0.37
20 - CLARA – Euclidean	0.33	0.30	0.36
20 - CLARA - Manhattan	0.29	0.36	0.31

K (Used only with k-means)	K-means on HDBSCAN	HDBSCAN on K-means	HDBSCAN (Independent)
5	0.44	0.37	0.37
10	0.41	0.38	0.37
20	0.35	0.37	0.37

* Default clustering algorithm is k-means

If the initial clustering is done with Euclidean, ensembling is done with Manhattan (using the same algorithm) and vice-versa

∞ If the initial clustering is done with Euclidean, ensembling is also done with the same measure but using a different clustering. Here the cross was done between CLARA and k-means

10 CONCLUSION

The aim of the project was to construct cluster ensembles and help formulate a comparison between regular clustering techniques and cluster ensembles. It was also intended to detect concept drifts of any kind in the data streams or large datasets (which can practically be assumed as data streams).
It constitutes a successful implementation of data type conversion i.e., from categorical to numerical, for working with the partitioning algorithms and hence be able to process and cluster any data with mixed data types.
The project demonstrates a way to design an ensemble using clustering algorithms with the help of ensemble techniques.

11 FUTURE SCOPE

In this work, partition based clustering algorithms were used in majority to solve the cluster ensemble problem. A combination of k-Means with the DBSCAN algorithm was also tried. But it was found that the results were not as expected.

So we hope that a combination of partition based algorithms with hierarchical and grid-based algorithms may be used in conjunction with each other to improve the clustering ensemble accuracy/usage.

APPENDICES

CODE

Self-Ensemble – Manhattan k-Means on Euclidean k-Means

```
g$cluster = k5e$cluster
g5e <-g[order(g$cluster), ]
g5e_pro = g5e[, 1: 17]
g = g[, 1: 17]
gboost5e = Kmeans(g5e_pro, 5, iter.max = 100, nstart = 100, method = "manhattan")
d5e = daisy(g, metric = "euclidean")
# s5e = silhouette(k5e$cluster, d5e)
plot(silhouette(k5e$cluster, d5e), col = c("red", "green", "blue", "purple", "yello
w"), border = NA)
d5e_pro = daisy(g5e_pro, metric = "manhattan")
# s5e_pro = silhouette(gboost5e$cluster, d5e_pro)
plot(silhouette(gboost5e$cluster, d5e_pro), col = c("red", "green", "blue", "purple
", "yellow"), border = NA)
```

Self-Ensemble – Euclidean k-Means on Manhattan k-Means

```
g$cluster = k5m$cluster
g5m < -g[order(g$cluster), ]
g5m_pro = g5m[, 1: 17]
g = g[, 1: 17]
gboost5m = Kmeans(g5m_pro, 5, iter.max = 100, nstart = 100, method = "euclidean")
d5m = daisy(g, metric = "manhattan")
# s5m = silhouette(k5m$cluster, d5m)
plot(silhouette(k5m$cluster, d5m), col = c("red", "green", "blue", "purple", "yello
w"), border = NA)
d5m_pro = daisy(g5m_pro, metric = "euclidean")
# s5m_pro = silhouette(gboost5m$cluster, d5m_pro)
plot(silhouette(gboost5m$cluster, d5m_pro), col = c("red", "green", "blue", "purple
", "yellow"), border = NA)
```

Self-Ensemble – Manhattan based CLARA on Euclidean based CLARA

```
g$cluster = c5e$clustering
gce5 < -g[order(g$cluster), ]
gce5_pro = gce5[, 1: 17]
g = g[, 1: 17]
gboostce5 = clara(gce5_pro, 5, metric = "manhattan", stand = FALSE, samples = 100,
sampsize = 50, trace = 0, rngR = FALSE, pamLike = TRUE, correct.d = TRUE)
dce5 = daisy(g, metric = "euclidean")
sce5 = silhouette(c5e$clustering, dce5)
plot(sce5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
dce5_pro = daisy(gce5_pro, metric = "manhattan")
sce5_pro = silhouette(gboostce5$clustering, dce5_pro)
plot(sce5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Self-Ensemble – Euclidean based CLARA on Manhattan based CLARA

```
g$cluster = c5m$clustering
gcm5 < -g[order(g$cluster), ]
gcm5_pro = gcm5[, 1: 17]
```

```
g = g[, 1: 17]
gboostcm5 = clara(gcm5_pro, 5, metric = "euclidean", stand = FALSE, samples = 100,
sampsize = 50, trace = 0, rngR = FALSE, pamLike = TRUE, correct.d = TRUE)
dcm5 = daisy(g, metric = "manhattan")
scm5 = silhouette(c5m$clustering, dcm5)
plot(scm5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
dcm5_pro = daisy(gcm5_pro, metric = "euclidean")
scm5_pro = silhouette(gboostcm5$clustering, dcm5_pro)
plot(scm5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Cross-Ensemble – k-Means on CLARA (Euclidean)

```
g$cluster = c5e$clustering
gce5 < -g[order(g$cluster), ]
gce5_pro = gce5[, 1: 17]
g = g[, 1: 17]
gboosteck5 = Kmeans(gce5_pro, 5, iter.max = 100, nstart = 100, method = "euclidean"
)
deck5 = daisy(g, metric = "euclidean")
seck5 = silhouette(c5e$clustering, deck5)
plot(seck5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
deck5_pro = daisy(gce5_pro, metric = "euclidean")
seck5_pro = silhouette(gboosteck5$cluster, deck5_pro)
plot(seck5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Cross-Ensemble – CLARA on k-Means (Euclidean)

```
g$cluster = k5e$cluster
gke5 < -g[order(g$cluster), ]
gke5_pro = gke5[, 1: 17]
g = g[, 1: 17]
gboostekc5 = clara(gke5_pro, 5, metric = "euclidean", stand = FALSE, samples = 100,
 sampsize = 50, trace = 0, rngR = FALSE, pamLike = TRUE, correct.d = TRUE)
dekc5 = daisy(g, metric = "euclidean")
sekc5 = silhouette(k5e$cluster, dekc5)
plot(sekc5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
dekc5_pro = daisy(gke5_pro, metric = "euclidean")
sekc5_pro = silhouette(gboostekc5$clustering, dekc5_pro)
plot(sekc5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Cross-Ensemble – k-Means on CLARA (Manhattan)

```
g$cluster = c5m$clustering
gcm5 < -g[order(g$cluster), ]
gcm5_pro = gcm5[, 1: 17]
g = g[, 1: 17]
gboostmck5 = Kmeans(gcm5_pro, 5, iter.max = 100, nstart = 100, method = "manhattan"
)
dmck5 = daisy(g, metric = "manhattan")
smck5 = silhouette(c5m$clustering, dmck5)
plot(smck5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
dmck5_pro = daisy(gcm5_pro, metric = "manhattan")
smck5_pro = silhouette(gboostmck5$cluster, dmck5_pro)
plot(smck5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Cross-ensemble – CLARA on k-Means (Manhattan)

```
g$cluster = k5m$cluster
gkm5 < -g[order(g$cluster), ]
gkm5_pro = gkm5[, 1: 17]
g = g[, 1: 17]
gboostmkc5 = clara(gkm5_pro, 5, metric = "manhattan", stand = FALSE, samples = 100,
 sampsize = 50, trace = 0, rngR = FALSE, pamLike = TRUE, correct.d = TRUE)
dmkc5 = daisy(g, metric = "manhattan")
smkc5 = silhouette(k5m$cluster, dmkc5)
plot(smkc5, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
dmkc5_pro = daisy(gkm5_pro, metric = "manhattan")
smkc5_pro = silhouette(gboostmkc5$clustering, dmkc5_pro)
plot(smkc5_pro, col = c("red", "green", "yellow", "blue", "purple"), border = NA)
```

Cross-Ensemble – k-Means on HDBSCAN

```
g$cluster = hdb1$cluster
gdb < -g[order(g$cluster), ]
gdb_pro = gdb[, 1: 17]
g = g[, 1: 17]
ddbk5 = daisy(g, metric = "euclidean")
plot(silhouette(hdb1$cluster, ddbk5), col = p1, border = NA)
gboostdbk5 = Kmeans(gdb_pro, 5, iter.max = 100, nstart = 100, method = "euclidean"
ddbk5_pro = daisy(gdb_pro, metric = "euclidean")
plot(silhouette(gboostdbk5$cluster, ddbk5_pro), col = distinctColorPalette(5), bord
er = NA)
```

Cross-Ensemble – HDBSCAN on k-Means

```
g$cluster = k5e$cluster
g5e < -g[order(g$cluster), ]
g5e_pro = g5e[, 1: 17]
g = g[, 1: 17]
dkdb5 = daisy(g, metric = "euclidean")
plot(silhouette(k5e$cluster, dkdb5), col = c("red", "green", "blue", "purple", "yel
low"), border = NA)
gboostkdb5 = hdbscan(g5e_pro, minPts = 4)
dkdb5_pro = daisy(g5e_pro, metric = "euclidean")
plot(silhouette(gboostkdb5$cluster, dkdb5_pro), col = p1, border = NA)
```

REFERENCES

[1]. Anil K. Jain, “Data Clustering: 50 Years Beyond K-Means” International Conference on Pattern Recognition, Michigan, 2010

[2]. H. Ralambondrainy, "A conceptual version of the K-means algorithm", Pattern Recognition Letters 16(1995) 1147-1157

[3]. Shankar B. Naik, Jyoti D. Pawar, “Clustering Attribute Values in Transitional Data Streams”, International Conference on Computing, Communication and Automation (ICCCA2017)

[4]. Junjie Wu, Jian Chen, Hui Xiong, Ming Xie, "External validation measures for K-means clustering: A data distribution perspective", Expert Systems with Applications 36 (2009) 6050–6061

[5]. M.S.B. PhridviRaj, C.V. GuruRao, “Data mining – past, present and future – a typical survey on data streams” The 7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013)

[6]. Aditee Jadhav, Leena Deshpande, “An Efficient Approach to Detect Concept Drifts in Data Streams”, 2017 IEEE 7th International Advance Computing Conference

[7]. Lei Du, Qinbao Song and Xiaolin Jia, “Detecting concept drift: An information entropy based method using an adaptive sliding window”, Intelligent DataAnalysis 18 (2014) 337–364 -

[8]. Reza Ghaemi, Md. Nasir Sulaiman, Hamidah Ibrahim, Norwati Mustapha, "A Survey: Clustering Ensembles Techniques", World Academy of Science, Engineering and Technology 26 2009 – clustering combination approach,

[9]. Saitta, S., Raphael, B. and Smith, I.F.C. "A comprehensive validity index for clustering", Intelligent Data Analysis, vol. 12, no 6, 2008

Table 4. Project Detail

Student Name	Sriharsha Daparti		
Registration Number	140953194	Section/Roll No.	B/24
Email Address	sriharsha.daparti@gmail.com	Phone No.(M)	9550854856
Student Name	Visal Kancharla		
Registration Number	140953148	Section/Roll No.	A/22
Email Address	vishalkancharla725@gmail.com	Phone No.(M)	9916416111

Project Details

Project Title	Detection of concept drifts in data streams using cluster ensembles		
Project Duration	4 Months	Date of Reporting	10-01-2018

Internal Guide Details

Faculty Name	Mr. Nimal Kumar Nigam
Full Contact Address with PIN Code	Assistant Professor – Senior Scale Department of Information and Communication Technology, Manipal Institute of Technology Manipal-576104
Email Address	nirmal.nigam@manipal.edu