FAST CLUSTERING WITH NOISE REMOVAL FOR LARGE DATASETS

by

Afees Adegoke Odebode

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2017

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Availability of large temporal data enabled by improved collection tools and storage devices has posed a new set of challenges in data mining, especially in the area of clustering data into different groups according to the basic attributes. The existing clustering algorithms, such as K-means, tend to suffer from slow processing speed. In addition, most of them lack the ability to eliminate outliers and anomalies. In this thesis, we present three fast clustering algorithms with noise removal capability: KD, KDS, and KDSD.

Technically, the proposed algorithms make use of the features of three existing data mining methods, K-means, DBSCAN and K-Nearest Neighbor (KNN). K-means has been an effective clustering algorithm. However, the clusters resulting from K-means are likely to include many outliers. In addition, K-means does not scale well with cluster size. In our research, to tackle the outlier problem, we proposed KD, a novel clustering algorithm with noise removal capability that is based on K-means and DBSCAN. Essentially, DBSCAN is employed to remove the outliers in the clusters resulting from K-means. To solve the scaling problem with K-means, we proposed KDS, a fast clustering algorithm that scales well. Finally, KDSD, a fast clustering algorithm with noise removal capability was proposed to achieve both excellent scalability and noise removal ability.

The performance of the proposed algorithms is thoroughly investigated through extensive experiments with a large power consumption data set. Our experimental results indicate that, compared to K-means, KDS runs at a much faster rate. Specifically, it takes K-means 7.56 seconds to cluster the whole data set under investigation. However, it takes KDS 0.363 seconds and 0.513 seconds in the case of 1% and 5% training sample respectively. In addition, although KDSD is not as fast as KDS due to the final anomaly removal operation, it outperforms KD. In our experiments, it takes KD 268.62 seconds to complete the clustering process while it takes KDSD 237.836 seconds in the worst case.

LIST OF ABBREVIATIONS USED

AMI	Advanced Metering Infrastructure
KM	K-Means
DBSCAN	Density-Based Spatial Clustering of Application with Noise
KNN	K Nearest Neighbor
KDS	K-means, DBSCAN with Small Training Set Nearest Neighbor
KDSD	K-means, DBSCAN &Small Training Set Nearest Neighbor and DBSCAN
ODIN	Outlier Detection using Indegree Number
MDMS	Meter Data Management System
HAN	Home Area Network
DNO	Distribution Network Operators
DR	Demand Response
ToU	Time of Use
MODH	Modified Hausdorff
DTW	Dynamic Time Warping
LCSS	Longest Common Sub-Sequence

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Dr. Qiang Ye for his expert guidance, support, valuable feedback and wealth of knowledge which has contributed immensely towards this research work. I am equally thankful to my co-supervisor Dr. Srinivas Sampalli, whose door is always open whenever I needed assistance and for providing the enabling environment to work towards my goals. Their counsel, support, and encouragement as well continuous constructive criticism helped me immeasurably throughout this research. I equally thank all my friends at MyTechLab: Saurabh, Manveer, Shristi, and others for their encouragement and positive feedbacks.

Lastly, my deepest appreciation goes to my wife Dr. Mrs. Adeola Odebode, my children (Adekorede, Ayomide, and Adeshewa) and the rest of my family for their continuous prayer, support and encouragement. This thesis would not have been possible without their unconditional love and affection. I thank you all!

CHAPTER 1 INTRODUCTION

Advancement in data collection tools, storage devices and continuous improvements in new technologies and applications as well as people's capability to generate data have contributed to a phenomenal growth in data available for analysis. Improvement in networking, wireless technologies, data storage, communication and sensor devices etc. have increased traffic on the internet which has resulted in access to increased high-volume, high dimensional data sets [1].

The fast-growing, large volume of data collected and managed in numerous databases, requires powerful analytical tools to properly understand them [2]. Although the process of discovering or organizing data into meaningful patterns arises naturally in many scientific endeavors, new methods and techniques will be needed to assist us to transform today's high-volume, high-velocity data sets into meaningful information. In fact, this has opened a new set of challenges for data mining experts such as extracting interesting patterns and features, establishing relationships between the different features, identifying anomalies and correctly grouping the data into different groups [3-4].

The task of identifying classes or grouping data into clusters is normally accomplished through classification or clustering techniques. Classification is a supervised learning method that uses a sampled data set to represent and correctly describe the features of the data to generate a model for putting the data into different classes. Clustering is an unsupervised learning process that groups data into classes or clusters so that data objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters.

In this thesis, we propose three innovative clustering algorithm for a large data set with noise removal using a test data set from Smart Meters [5-6]. As part of the process of achieving a fast algorithm for clustering power data, we first present the motivation of this study, then we describe the overview of the proposed algorithms. Finally, a short outline of the thesis is included.

1.1 MOTIVATION

The vast amount of data that are being generated every day, render the traditional method of capturing, storing, analyzing, querying and processing inadequate for handling

large data set. The quantity of data increases at a phenomenal rate because devices for gathering information such as – mobile devices, remote sensing, cameras, and wireless sensor networks – keep improving. Opportunities for big data could be found almost in all human endeavors: manufacturing, transportation, automotive, energy as well as cyber security. The task of organizing data into meaningful information through traditional methods such as relational database systems is no longer effective. Not only is the data increasing in volume, the variety, as well as the veracity of the data set, also keeps changing [7]. In essence, improvement of traditional method is highly necessary.

The energy industry has been undergoing changes lately and one major change is the introduction of smart meters. The device can record millions of data per day which can be classified as a big data [2]. Installation of smart meters; under the Advanced Metering Infrastructure (AMI) has opened a new set of opportunities for understanding customers' electricity consumption patterns. This new information can readily be used by Distribution Network Operators (DNO), to identify suitable customers for energy management solutions such as Demand-Side-Response. In addition, this can improve the effectiveness of storage device in electricity distribution. Choosing correct attributes, a DNO can use the pattern to identify suitable customer group for demand reduction solutions. Gaining insight into the consumption patterns of different categories of electricity users, based on their load data classification, will be of huge benefit to electricity producer as well as the consumers. It will assist in achieving proper planning and distribution of electricity. It can also aid the development of a more competitive market policy, and with a good knowledge of their consumption patterns, consumers can adjust their electricity consumption to suit their needs [8].

Several algorithms have been developed to group time series (Smart Meter) data into clusters. However, the two main algorithms are the partition based and the hierarchical clustering algorithms [2]. K-Means is an example of partition or centroid based clustering algorithm. It partitions several observations into *k* clusters, with each observation belonging to a cluster with the nearest centroid. It is a popular clustering technique because it is easy to implement although it may fail to converge to an optimum if the dataset is not well separated. K-means is also not sensitive in detecting outliers [9].

Most algorithms developed for smart meters use centroid-based approach without provision for detecting the outlier and improving the speed of clustering the dataset [8], [10–12]. The increase in deployment of smart meters requires a more efficient, scalable and outlier-aware algorithm to meaningfully cluster the dataset and provide accurate information for customers' needs. In this thesis, we present a series of algorithms that are fast and efficient for the large data set and can correctly detect possible outliers. As part of the process of achieving a fast algorithm for clustering power data, we implemented the following steps:

(1) Import time series data for each meter ID, clean it and calculated aggregates on daily basis.

(2) Define the consumption pattern of customers based on electricity usage for each meter ID on the imported time series data.

(3) Cluster all meter IDs based on weather characteristics such as temperature to reduce the volume of data for analysis. The process can be extended to other behavioral characteristics of the customer obtained from the time series.

(4) Predict unlabeled time series data using already designed predictive model based on a Nearest Neighbor algorithm.

(5) Clean the cluster generated

(6) Evaluate and compare our model to existing model based on cluster overlap, running time and time complexity.

1.2 OVERVIEW OF THE PROPOSED ALGORITHMS

The major objective of this thesis is to design series of algorithms that are capable of detecting outliers and that scale well with large data sets. The proposed algorithms utilize the positive features of three existing algorithms: K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and K- Nearest Neighbor Algorithm.

K-Means Algorithm partitions a set of observations $(x_1, x_2 \dots x_n)$ in d-dimensional real vector into k groups where ($k \leq n$). The aim is to minimize the within-cluster sum of squares and the objective can be stated formally as:

$$\underset{s}{\arg\min} \sum_{i=1}^{k} \sum_{x=s_i}^{k} ||x - \mu_i||^2 = \underset{s}{\arg\min} \sum_{i=1}^{k} |S_i| \; Var \; S_i$$

where μ_i is the means of points in S_i

It has two major steps: assignment and update. In the assignment step, the observations are assigned to the cluster that yields the least within-cluster sum of squares (WCSS) and the update step recalculates the centroids of the observations in the new clusters. The two-step process is repeated until convergence [13]. The proposed algorithms use K-Means to determine the clusters from a selected sample of observations.

In this thesis, DBSCAN was used to prepare the initial training set for the K-Nearest Neighbor Algorithm. It was also used to remove outliers from the final clusters generated from the Nearest Neighbor Algorithm in the KDSD version of the algorithm.

The proposed algorithm explores the nonparametric nature of K-Nearest Neighbor to classify the rest of the dataset using a Small Training Set K-Nearest Neighbor(SKNN). SKNN uses a small training set obtained from the cleaned data set resulting from DBSCAN. The training set is then used to generate labels. The labels are then used to classify the rest of the dataset based on majority vote of neighbors.

1.3 THESIS OUTLINE

The rest of the thesis is organized as follows. Chapter 2 includes a detailed background on clustering types, application, tools and clustering algorithms. It describes some of the key terms and research that has been undertaken especially in clustering of smart meter data. Some existing algorithms are examined in detail from the viewpoint of research that has recently been undertaken in the field. In addition, this chapter presents an overview of several important concepts and technologies that are closely related to our proposal. Chapter 3 discusses the idea behind our algorithm “Fast Clustering with Noise Removal”. It highlights the different stages of the algorithm. We present the proposed fast clustering algorithm for time series energy data and time series dataset in general. Chapter 4 contains a detail discussion of the experimental result obtained from the algorithm’s application on smart meters. Finally, Chapter 5 discusses the result and the conclusion drawn from the result of the experiment.

CHAPTER 2 RELATED WORK

Due to the increase of the data available for data analysts because of improved sensing and data storage capacity, the opportunity of generating information from raw/unlabeled data set through data analysis has been tremendous. Research in Big data and Data mining, as well as Machine learning, have applied existing and new algorithms to resolve issues related to identifying patterns/ characteristics of observations based on behavioral as well as attitudinal characteristics inferred from the data. This chapter describes some of the key terms and research that has been conducted especially in clustering of smart meter data. Some existing algorithms are examined in detail from the viewpoint of research that has recently been undertaken in the field. In addition, this chapter presents an overview of several important concepts and technologies that are closely related to our proposal.

2.1 CLUSTER ANALYSIS

Cluster analysis has been described as the organization of several patterns into groups based on similarity, the patterns are usually points in multidimensional space. Patterns that belong to the same cluster are closely similar to one another compared to patterns that belong to another cluster [14]. Webster, an online dictionary defines Cluster Analysis as "a statistical classification technique for discovering whether the individuals of a population fall into different groups through quantitative comparison of multiple characteristics " [15]. Fraley and Raftery, conceptualized cluster analysis as the process of determining the intrinsic structure of a data set when there is no other information other than the values of the observation[16]. Cluster Analysis is a major task in exploratory data mining, and a useful tool for statistical data analysis used in the various field such as image analysis, bioinformatics, data compression and machine learning among others. Clustering can be defined formally as a set of subsets of the form:

$H = H_1 , \dots , H_k \subseteq S$

such that:

$S = \cup_{i=1}^{k} H_i$ and $H_i \cap H_j \neq \emptyset \quad \forall i \neq j$

There are different tools and approaches to clustering each with its own definition of what represents a cluster. Clusters even from the same data set differ in shape, size and density and the presence of noise may constitute a roadblock in detecting the right cluster in a dataset. Clusters can be differentiated based on how small the distance between the members of a group is to one another or the density of the data space or as mentioned earlier the shape may be the defining characteristic. Flynn [16], correctly captured the task of clustering as a subjective process and that the choice of the clustering algorithm is a function of the individual data set as well as the use of the result obtained from clustering [14]. A cluster can be described as isolated or distinct groups in a data set or the organization of a collection of data points based on similarity. The pattern that describes a cluster is like one another while those that belong to the different cluster are dissimilar to one another [6]. Most data set that is often encountered in practice such as time series data, documents and text data set often do not have distinct characteristics to exhibit compactness in their distribution.

As difficult as the process of clustering may be, it is a useful tool in a wide number of applications too numerous to fully explore. One of its applications is in Document clustering- a field in information retrieval and language processing that arrange documents into clusters with each cluster exhibiting some common characteristics based on the similarity measure. Documents clustering is a technique in information retrieval aimed at the efficient organization, browsing, and summarization of large volume of text documents [17]. Cluster analysis has also been applied in medical image processing; a good segmentation of the medical images produced by the clusters will benefit clinicians and patients especially for visualization, surgical planning and for early disease detection [18]. Mining in time series are often characterized by high dimension and large volume with a propensity for continuous updates; the mining tasks associated with time series data can be roughly classified into four main areas: Pattern discovery and clustering, classification, rule discovery, and summarization. Examples of time series data include counts of sunspots, heights of ocean tides, ocean isotope levels, exchange rates, Smart Meter data etc.

The natural temporal ordering of time series data makes it unique in its requirement for clustering. Time series data during clustering are often represented as either aggregate or as a sample of the original data. The reason is to reduce the dimensionality of the data and the volume. The continuous increase in the volume and variety or type of Time Series data available for analysis requires improvement in processing, summarization, and understanding of information buried in the data. Such data is mostly unstructured and often meaningless in their raw form; making it difficult to analyze them [19]. Clustering can be effectively employed in understanding the underlying structure, detect anomalies, estimate the degree of similarity and organize data into meaningful summary [20]. Zhou, in studying the consumption pattern of electricity users (a Time Series Data), affirms that there is a marked difference in the consumption pattern of all categories of consumers from commercial to residential or industrial users; electricity consumers, even of the same type exhibit different consumption patterns[8]. Knowing the different consumption pattern and being able to adequately cater for different categories of customers need through consumption information is a potent means of improving service delivery to the customers.

2.2 CLUSTERING METHODS

Clustering groups data objects into subsets in a way that objects that are similar are grouped together, while dissimilar objects are grouped into different clusters. Classification is predictive while Clustering is descriptive. Clustering algorithm can be classified into five (5) categories: Partitioning Methods, Density based methods, Grid based methods, Hierarchical method and Model-based Methods. Three of the approaches will be explained briefly below [21].

2.2.1 PARTITIONING METHODS

The idea behind this method of clustering data is to partition a given data set say *n* into a set of *k* clusters. The goal in partitioning methods is to find a specified number of cluster (*k*) say: $C_1, C_2 \ldots . C_k$ of the input dataset *n* that optimizes a certain criterion. The criterion is usually of the form

$$E = \sum_{i=1}^{k} \sum_{x=s_i}^{k} d(x, \mu_i)$$

Where μ_i represents the centroids of the clusters C_i and $d(x, \mu_i)$ represents the Euclidean distance between x and μ_i. It is required that the estimated number of partition be selected at the start of the run. All instances and possible partitions are then searched iteratively and enumerated to achieve a global optimum. Rokach and Maimon, stated that achieving global optimum is not feasible instead greedy heuristics are often employed towards a global optimum[22]. Two types of partitioning methods were identified by [8]:

(i) Error Minimization Algorithms- In error minimization, the underlying idea is to determine cluster pattern that minimizes a chosen error criterion usually Sum of Squared Error (SSE) through exhaustive search or heuristics. An example is K-Means Algorithm.

K-Means Clustering: Is a classic example of partitioning or centroid-based method and it minimizes the SSE. It can be described as follows. Given a set of *n* observations $\{x_1, x_2, \dots, x_n\} \in \mathbb{R}^d$ where each $x_i = \{x_{i1}, x_{i2}, \dots, x_{id} \in \mathbb{R}^d\}$ k-means partition the dataset into k clusters $C_1, C_2 \dots . C_k$. Each cluster C_i has an associated centroid $C_i = (C_{i1}, C_{i2,} \dots, C_{id}) \in \mathbb{R}^d$ which can be computed using:

$$C_{ij} = \frac{\sum_{x_i \in C_i} x_{ij}}{|C_i|},$$

The methods consist of two main steps: The cluster assignment where the data points are assigned to their respective clusters, re-computation of the centroids until there are no more changes. A simplified k-means algorithm is as shown below:

Algorithm 1: A simplified *k*-means Algorithm

Input: A time-series data set

Parameters: Number of cluster *k*

Output: Clusters

```
(1) Arbitrarily choose k as initial cluster centers
(2) Repeat
        a. (re) assign each object to the cluster with the closest centroid
           Or the cluster to which it is most similar using the mean value of the
           objects in the cluster
        b. Recalculate and update the cluster means (centroids)
(3) Until convergence
```

Although it might not converge to the expected optimum if the initial seed(k) is not selected appropriately or the data is not clearly separated, K-means is capable of handling large data set if it can fit into memory and it is efficient.

(ii) Graph-theoretic clustering is an alternative approach that constructs a similarity graph where two elements *i* and *j* are connected by an edge if and only if *i* and *j* are similar enough to belong to a single cluster. If the similarity measure is clear and consistent the graph will consist of disjoint cliques otherwise there could be overlaps in the graph structure. An example is the Minimal Spanning Tree.

2.2.2 HIERARCHICAL METHODS

Hierarchical clustering methods work by grouping data objects into a tree-like structure. It may be either agglomerative or divisive. If the hierarchy is formed in a bottom-up fashion it is agglomerative otherwise it is divisive [23]. The clusters from hierarchical clustering suffer from its inability to adjust once a merge or a split has been concluded. It cannot backtrack in case a poor choice has been made in the hierarchical decomposition and it is not scalable for large dataset. The figure below shows the two main types of Hierarchical clustering.

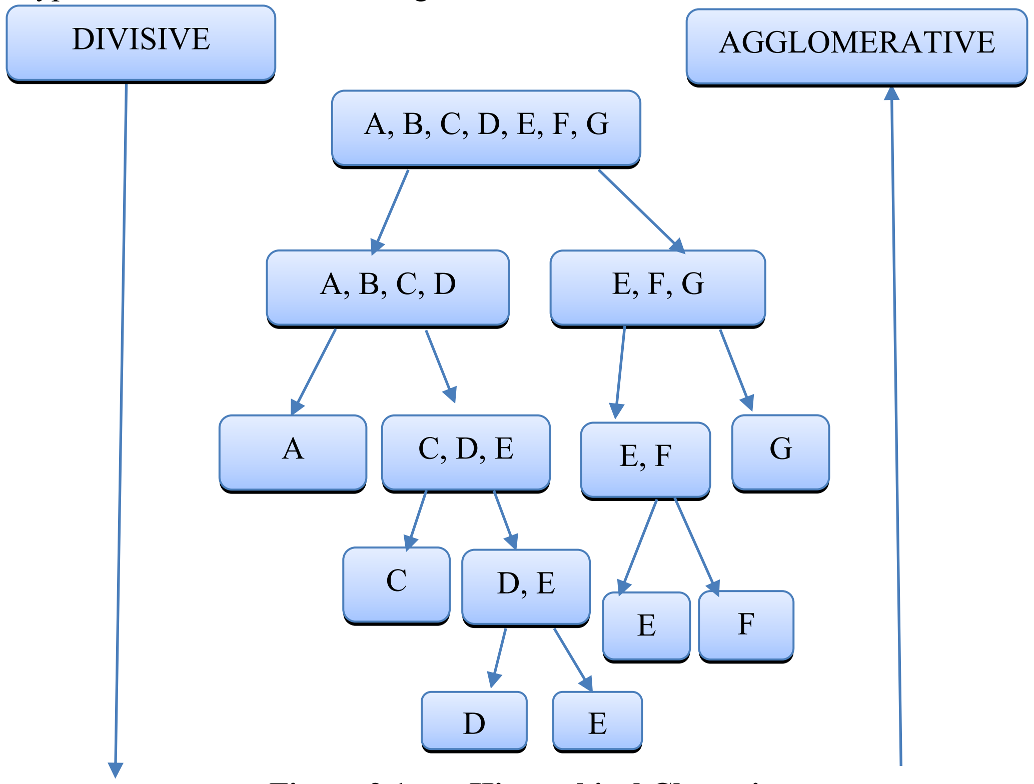


Figure 2.1 Hierarchical Clustering

2.2.3 DENSITY BASED METHODS

Density based clustering is designed around the idea of density. Unlike partition-based methods, the cluster continues to grow if the density in the neighborhood exceeds a certain threshold. Each of the data point in a cluster is considered a member of the neighborhood of a given radius if it contains at least a minimum number of points. Density based methods are used to filter and detect possible outliers in a cluster and they can discover clusters of unusual shape [24]. OPTICS is an example of Density Based Clustering algorithm. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is another example of the density based clustering algorithm. It is efficient for large spatial databases and it can discover arbitrarily shaped clusters. We would explore DBSCAN further because it is one of the main algorithms that will be employed in the design of our new algorithm. DBSCAN has the following parameters and they are as defined:

(i) $\mathcal{E}$-neighborhood of an object is the maximum radius of the neighborhood from a point say *p*

(ii) Given a set of objects, say D, an object *p is* directly density-reachable from another object *q* if p is within $\mathcal{E}$-neighborhood of *q,* and *q* is a core object.

(iii) Core Point: A point *p* is a core point if at least minimum points (minPoints) are within $\mathcal{E}$ distance of it (p included). The core points are said to be directly reachable from *p.* A point cannot be directly reachable from a non-core point.

(iv) Density-Reachable: An object *p* is density-reachable from another object *q* w.r.t to minPoints and $\mathcal{E}$ in a set of objects, D, if there exist objects $p_1, \dots p_n$ where p_1=q and $p_n = p$: p_{i+1} is density reachable from $p_i \; for \; 1 \leq i \leq \; n, p_i \; \in D$

(v) All points or object not reachable from any other points are outliers.

(vi) An object *p* is density-connected to another object *q* w.r.t $\mathcal{E}$ and minPoints in a set of objects D, if there is an object say $o \in D$: both p and q are density-reachable from o.

DBSCAN requires two input parameters: minimum number of points (minPoints) and $\mathcal{E}$; the two requires minimum domain knowledge to be able to correctly determine them, [21]. In this thesis, a further explanation of the two parameters will be expatiated upon during the presentation of the case study in chapter 4.

2.3 ADVANCED METERING INFRASTRUCTURE

Advanced Metering Infrastructure is a combination of home network systems such as thermostats, smart-meters, communication networks from the meters to corporate data centers, the meter data management system and the integration of those systems into the software platforms. The entire data collection system that revolve around the customers' meter's site, the communication networks and the service provider that the customers interact with, (gas, water utility as well as the medium(systems), through which data is been received and managed constitutes an Advanced Meter Infrastructure(AMI) [41]. AMI deployments have improved the collection and acquisition of refined and accurate consumption information associated with each customer from which behavioral and consumption characteristics of individual customers can be inferred. If external stimuli such as weather are accounted for then customer's energy consumption can be attributed to lifestyle [25]. Figure 2.2, shows an overview of the different components that make up the Advanced Metering Infrastructure.

NETL Modern Grid Strategy Powering our 21st-Century Economy, US Department of Energy Office of Electricity and Energy Reliability: page 5 (2008)

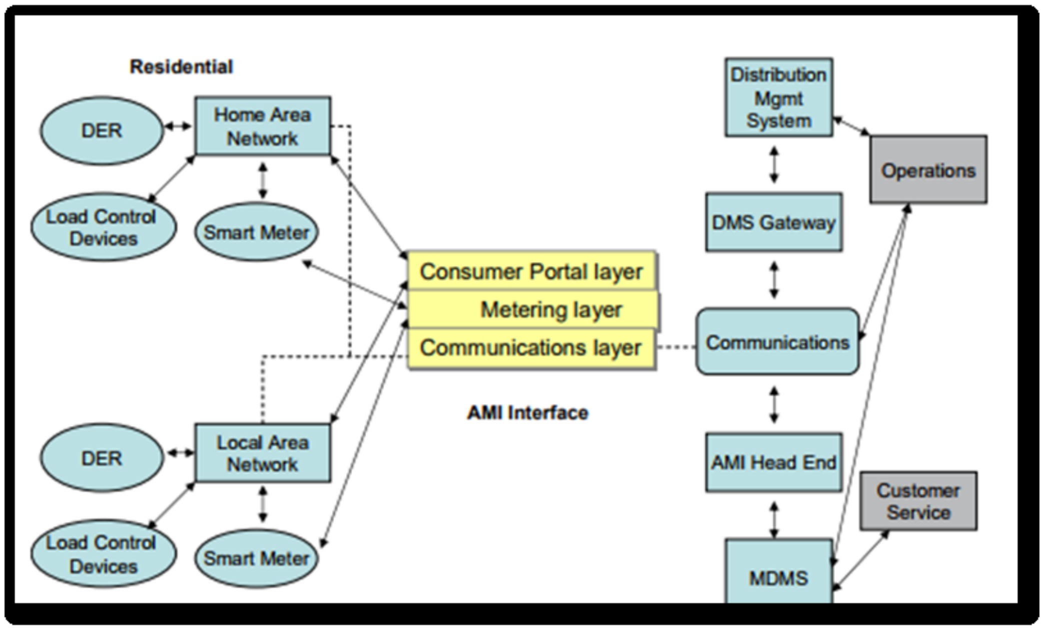


Figure 2.2 Advanced Metering Infrastructure **[26]**

As shown above, AMI is a combination of technologies that have been integrated to work together as one. Some of its components include:

- Smart meters
- Wide-Area Network Infrastructure
- Meter Data Management Systems (MDMS)
- Home Area Network (HANs)
- Operational gateway

AMI improves customers' choice about price and services provided by energy companies, as well as higher reliability and improves power quality. Utility companies also benefit from the installation of AMI through the provision of accurate and timely bills, a more efficient and reliable service delivery as well as improving customer service. It is fast becoming the standard in today's utility industry in that it has contributed to improving the performance of the grid and customer experience have also been positively impacted[26][27]. To achieve maximum performance utility companies will need to scale up research in analyzing the data collected from AMI devices in other to continuously improve customers experience and efficiency in service delivery[28].

2.4 SMART METERS

Smart meters are solid state programmable devices that can perform time-based pricing, record consumption data for use by consumer and utility companies, calculate net metering, notify the customer in case of loss of power, and remotely perform turning on and off operations as well as perform power quality monitoring and energy prepayment. Smart meters record electricity consumption in intervals of hours or minutes. It can communicate both wired and wirelessly (Wi- Fi) on the network to relay information between itself and server[29][30][26][31]. A smart meter can measure electricity consumption at the installed facility and transmits this information to the consumer and the energy supplier or operator. It can measure and monitor electricity consumption in real-time or near real-time as well as record the power quality. It is at different stages of implementation in most developed countries. For example, Australian Essential Service Commission (ESC) adopted a new Electricity Customer Metering Code in 2004. In

Canada, Ontario Energy Board had been working to define a regulatory framework for the effective implementation of Smart Meters since 2007. Likewise, other provinces in Canada are at various stages in their implementation. Oxxio, a Dutch company, is at the forefront of ensuring that smart meter is fully deployed in Netherlands [32]. Vassileva and Campillo, in evaluating consumer interaction with smart meters based on their preferences and feedback for smart meter deployment in Sweden; observed that smart meters provide customers with more detailed information about their energy consumption as well as variable pricing. When such potential (improved information and variable pricing) are fully explored by the network operators, energy saving through increased knowledge in energy consumption will be enhanced. They advocated for improvement in customer experience analysis rather than only dynamic pricing and improved electricity consumption information[28].

Figure 2.3 Smart Meter [42]

2.5 CLUSTERING AND SMART METER DATA

Before the advent of smart meters, various Non-Intrusive Load Monitoring Techniques have been adapted in the past, towards understanding the consumption pattern of consumers in households. Greveler et al constructed a power prediction function, which was based on series of pictures made of elementary shades. Additive RGB color notation was used in constructing each byte of the Red, Green and blue portion. Frames are then extracted; the darker the picture frames, the higher the rate of power consumption. The application of the algorithm on live TV programs shows that it is possible to accurately monitor power consumption profiles of customers, the signatures of the devices associated with them, and even the viewed content [33]. Chicco et al identified the electrical behavior of customers by forming the Representative Load Pattern (RLP) of each customer using unsupervised clustering algorithms (K-means, fuzzy K-means, and modified follow-the-ladder). All the clustering algorithms tested could form the required number of clusters. Modified follow-the-leader algorithm outperformed other algorithms in that it built well-separated classes and created uncommon load patterns. They reiterated the importance of clustering algorithms in the classification of electricity customer using load pattern shape especially for the distribution service providers [34].

Various clustering methods and approaches have been applied to analyze trends and identify groups using time-series smart meter data. To manage customer user specific characteristics, enhance theft detection in electricity use, track behavioral changes and enable development of consumer specific demand response programs; Frameworks have been developed to segment customers based on consumption in relation to other context-specific attributes such as weather and ToU (Time of Use) [30]. Mandel et al[35], while examining the impact of raw data temporal resolution on the residential electricity profiles of customers using clustering concludes that the key to properly represent the consumption pattern of a customer is to be able to select a data resolution that satisfies both the level of detail to be represented as well as the essential behavior associated with a user. Three algorithms were used in the process (K-Means, Hierarchical algorithm and Dirichlet process mixture model algorithm). They concluded that 8 minutes' resolution data would provide a useful basis for establishing customer consumption pattern and that data collected slower than 30 minutes is not sufficiently reliable. The data used for this experiment were collected at 5 minutes' interval.

The process of enlisting customer for Demand Response (DR) program to determine their consumption characteristics based on adjusting the demand for power instead of adjusting supply is another aspect that has been studied in the clustering of smart meter data. Data-driven methodology for predicting customer eligibility to be enlisted in a DR program by using a combination of clustering and classification techniques have been developed with an algorithm such as Random forest [23], neural networks, decision trees, k-nearest neighbor, genetic algorithms and fuzzy clustering etc [36]. Pereira et al identified three major advantages of demand response which is: energy reduction during peak times, facilitating a balance between supply and demand, and reduction in energy bills [36]. Han and Piette were of the view that a successful implementation of demand response, aside from promoting a fair electric market operation is an effective short-term tool for correcting the imbalance between supply and demand [37].

Smart meters are susceptible to several attacks: Man-in-the-middle, Denial of service attack, Authentication, False-Data-Injection, and Disaggregation. Solutions to address privacy issues in smart meters have been developed using applied Fuzzy c-means to infer energy profile characteristics of customers. The result indicated that gaining insight into such knowledge can be a potent platform for curbing abuse by attackers through home invasion and other behavioral profiling of customer even by utility companies against customers wish, [38]. Parvesh et al, similarly observed that due to the wireless nature of communication in AMI there are potential threats to data privacy through injection of false data, unauthorized decoding of energy consumption readings and network jamming. It is equally possible for an attacker to learn customer movements either present or absent from home which may pose an unimaginable level of threat to the customer and the community at large [39]. Understanding customer consumption pattern through data obtained from a smart meter is a potent source of improving overall service delivery in the energy sector.

2.5 OUTLIER DETECTION AND DATA CLUSTERING

An outlier can be described simply as a data point that exhibits significant difference when compared with the rest of the data sets. Hawkins formally defined an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism"[40]. They are often referred to as abnormalities, deviants or simply anomalies. An outlier can sometimes contain useful information about behavior not yet captured by the data analyst. It has been used for fraud detection, intrusion detection systems, credit card fraud identification and medical diagnosis. Several approaches have been used in outlier detection, two common example of such approach will be explored further. The first approach is Distance based methods of outlier detection[41][42]. It assumes that the k-nearest neighbor distances or outlier data points differ appreciably from that of normal data points. Distance based methods exhibit high level of granularity in its analysis and this allows it to handle varied and tricky outliers. The second approach is the density-based method. It based on identifying dense regions and points that fall outside of the dense regions is classified as outliers, a good example is DBSCAN. Density based methods of outlier detection provide high level of interpretability, especially if the original attributes of the data set can be represented as sparse regions in the data[43]. Bae et al proposed two methods for detecting outliers in a data set: centrality and center-proximity. Centrality is a measure of how much, objects in a cluster recognize another object within the same cluster as the center of their cluster and center proximity measures how much close an object is to the center of the cluster. It was implemented using graph-based outlier detection methods such as a k-NN graph. The algorithms effectively solve the problem associated with local density and the micro-cluster problems[44]. The outlier detection algorithm, ODIN (Outlier Detection based on Indegree Number) has been used to improve the cluster generated by K-means. ODIN uses K-nearest neighbor graph, every vertex represents a data vector, and the edges are pointers to neighboring vectors[45]. In this thesis, we identified outliers in our dataset using a Density-based clustering algorithm (DBSCAN) that groups a set of densely packed points into clusters of the same unit while labeling points outside the dense region as outliers [46]. DBSCAN is robust to outliers and is very effective in detecting arbitrary shape. In addition, ODIN has only been applied to small data set to the best of our knowledge.

2.6 RECENT DEVELOPMENT IN FAST DATA CLUSTERING

Several algorithms have recently been developed to overcome the shortcomings of existing algorithms such as K-Means, DBSCAN, PROCLUS, CLIQUE[47] and a host of other algorithms. The main driving reason for the need for such improvement is mostly associated with the speed of clustering, choice of similarity, the number of clusters (parameters), curse of dimensionality, the data set to be clustered (Spherical or non-spherical) and a host of other factors. The speed of clustering has recently been gaining traction in the research community. Lin et al, improves the speed of processing of k-means algorithms for image retrieval by using the discrete function of the levels' histogram value along with a k-means algorithm to train cluster centers of the image database[1]. Rodriguez and Laio developed CFSFDP an algorithm that assumes that centroids of clusters are surrounded by neighbors with lower local density at a distance far from points with higher densities. The algorithm can be easily implemented, it is fast and it can cluster non-spherical data[48]. One major disadvantage of the algorithm is that it cannot be used to cluster data with multi-density peak accurately. To improve the algorithm, a grid-based partition version of the algorithm was developed (GbCFSFDP)[49], it is aimed at resolving some of the shortcomings of CFSFDP. Ghanem et al, introduces Dimension based Partition and Merging(DPM) to cluster large scale data sets with automatic cluster number detection. The algorithm is in three stages: Stage one, partition the data space into small dense partitions using dimension histogram and locating partition dimension value once. The second stage filters the noise based on the density of the dimension from the partitions. Lastly, clusters are constructed based on the identified boundaries from the samples[50]. Our own approach is aimed at clustering large time series data sets using a combination of features from three existing algorithms.

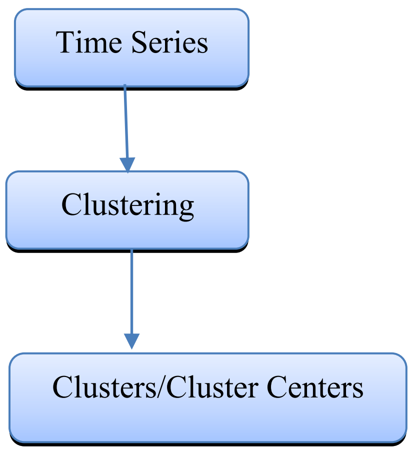
CHAPTER 3 FAST CLUSTERING WITH NOISE REMOVAL

In this chapter, we present the proposed fast clustering algorithms for time series energy data. Two clustering approaches mentioned previously will be combined as they are mostly related to our context: partitioning methods and density-based methods. Partitioning methods identify partitions k from the input data set where each partition represents a cluster. The number of clusters k will need to be manually specified. Density-based algorithms treat as a cluster a collection of data points spread in spatial space of the contiguous region of high density separated by regions of low density. Points outside of these regions are referred to as outliers [46]. The proposed algorithms are designed for speed and anomaly detection.

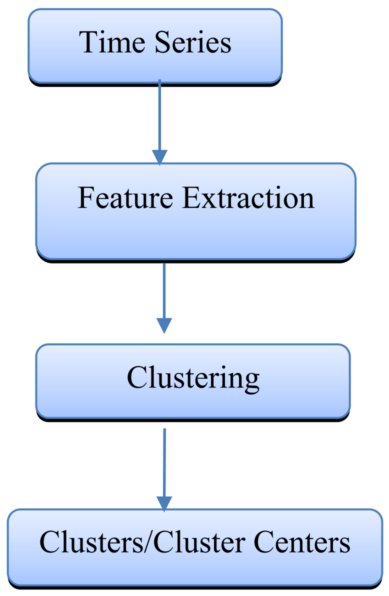
3.1 TIME SERIES DATA CLUSTERING

The data used for the experiment is a time series data set. In this thesis, we attempt to describe briefly time series data and two of the similarity measure that can be used to cluster them. Time series is a common data type widely used in diverse application areas such as engineering, finance, economics, communication, energy sector and online services. Time series data, unlike static data, are temporal and their features change with time. Identifying groups in static data clustering and time series clustering require a clustering algorithm but the approach differs based on the nature of the data available and the purpose of the application. Time series data can be discrete valued, real valued, univariate or multivariate and can be of equal or unequal length. In this section, we will explain time series, the different types of time series data clustering approaches, the proposed method and outline of the algorithm.

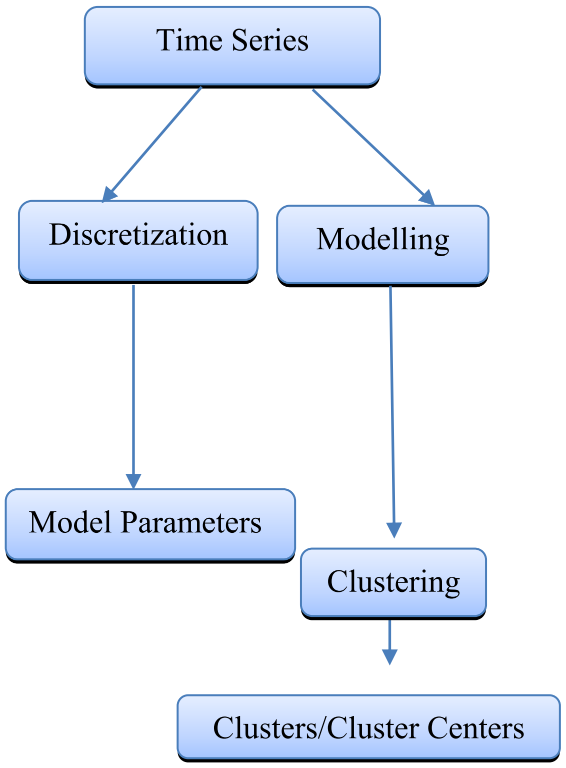
We will now describe briefly Time Series data set and two of the similarity measures that can be used to cluster them. Fu defines a times series data as a collection of observations made chronologically, often they are characterized by high volume, high dimensionality with a propensity for continuous updates[19]. The mining task associated with time series can be classified roughly into four main areas: Pattern discovery and clustering, classification, rule discovery and summarization. Zhang et al, identify three main objectives associated with clustering time series data: similarity in time, the similarity in shape and similarity in change[10] [51]. The large volume associated with time series data often requires that it be represented as either aggregate or with a sample from the original data set to reduce the dimensionality of the data. The simplest means of reducing dimensionality is through sampling but it may distort the original shape of the data especially for low sampling rate. An enhanced method is to use the average value of the features under consideration [19]. Keogh et al in reducing the dimensionality of the data set segments the sequences into equal-length sections and then record the mean value of each section. Each mean value is then indexed efficiently into a lower dimension. Other dimensionality reduction techniques include Spectral decomposition, Wavelet Decomposition and Singular Value Decomposition [52]. Liao identifies three types of approaches to time series data clustering as shown in the figures 3.1



(a) Raw-Data Based



(b) Feature Based



(c) Model-Based

Figure 3.1(a-c) Approaches to Time Series Data Clustering

The raw-data-based approach uses raw time series data. The similarity measure may need to be replaced with an appropriate one for time series data. The other two approaches either converts the time series into feature vectors or model parameters before clusters are generated [53]. Although other features such as the temperature will be used in data selection and processing, only one feature of the customers (consumption) will be used in determining the cluster to which a customer belongs and the overall representation of the data will still be maintained during the sampling process. An overview of the process involved in time series clustering is as shown in the Figures 3.2.

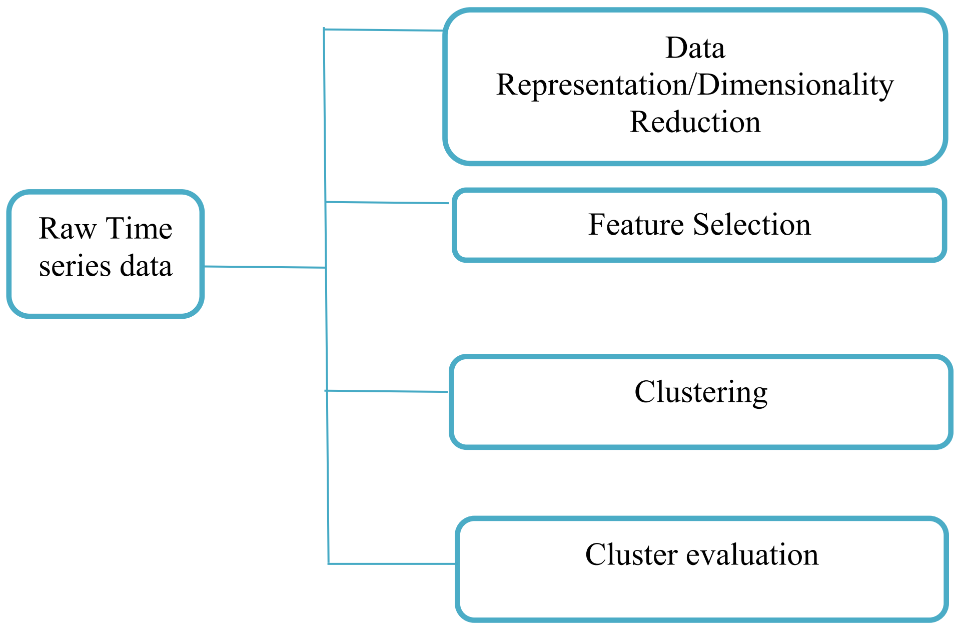


Figure 3.2 An overview of the clustering process.

3.2 SIMILARITY MEASURE

To measure how related two objects are in a cluster, a similarity measure is needed. The similarity measure is a set of rules that define the criteria for establishing a group as a cluster. Clustering algorithms are built on dissimilarity or similarity measure. They are usually non-negative real numbers. If the objects are close to one another the dissimilarity will be smaller and the reverse is the case if the objects are far from one another. The common similarity measure used in time series includes Hausdorff distance, modified Hausdorff (MODH), HMM-based distance, Dynamic Time Warping (DTW), Euclidean distance, Manhattan distance, Minkowski Distance, Euclidean distance in a PCA subspace, and Longest Common Sub-Sequence (LCSS). Each of this similarity measure has proven to be useful for different categories of data set but there is no one-size fit all in any of the methods. The one that has stood the test of time among the similarity measure is the Euclidean Distance, hence our preference for it in our design.

3.2.1 MINKOWSKI DISTANCE

This is a generalization of the Euclidean and the Manhattan distance. This Minkowski distance between two points

$X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$

can be defined as:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

where $p \geq 1$

When p = 1 it is referred to as the Manhattan distance and when it is equal to 2 it is referred to as the Euclidean distance.

3.2.2 EUCLIDEAN FRAMEWORK

This can be defined as the geometric distance between two points in space. Given two points x and y, it is computed using the following formula:

$$distance\ (x, y) = \sqrt{\sum_{i} (x_i - y_i)^2}$$

The Root Mean Square is

$$\sqrt{\Sigma_i (x_i - y_i)^2} \Big/ n$$

It has many advantages, one of which is that addition of a new object to the dataset does not affect the existing distance between objects. However, the difference in scale can have a strong effect on the dimension from which the distances are computed. We have taken care of this effect through normalization of the data set.

3.3 THREE NOVEL CLUSTERING METHODS: KD, KDS AND KDSD

In this thesis, we proposed a layered data-driven approach that is based on the household consumption characteristics and features such as temperature, humidity, and pressure etc. The features were selected based on the average consumption characteristics as well as the increase in consumption over a period e.g. winter period or the Holiday and weekends as well as weekly and daily consumption characteristics. The method is based on the consumption increment instead of the power consumption or the average consumption. This is to be able to identify a house that uses electricity more in winter and which does not and by how much. By knowing this parameter, we could categorize customers on the consumption increment which gives a clearer view of the consumption characteristics of customers rather than using averages over a period. Non-holiday weekdays were separated from holiday weekdays based on the assumption that holiday weekdays and weekends may have the same pattern of power consumption. The days were sorted from lowest temperature to the highest temperature based on consumption increment over the period under observation.

We have divided our algorithm into three different categories for ease of assessment and comparison. The three algorithms are KD- which combines K-means and DBSCAN algorithm on the whole data set, KDS – This algorithm uses DBSCAN to remove outliers from cluster initially generated by K-means from sampled data, this is then used to create clean clusters used as a training set for classification with SKNN. The KDSD is a complete form of KDS in that after the classification of the clusters from unlabeled data set, DBSCAN further removed any cluster, that was not initially removed through sampling.

In other to have a quick view of the consumption pattern of customers at various times of the year and different time frames: weekly, monthly etc. we have presented below the consumption pattern based on the average total consumption at different time/period of the year, as shown in figures (3.3 – 3.5) below. The choice of the period under consideration was chosen randomly to give different views of the data set at the different time of the period under observation.

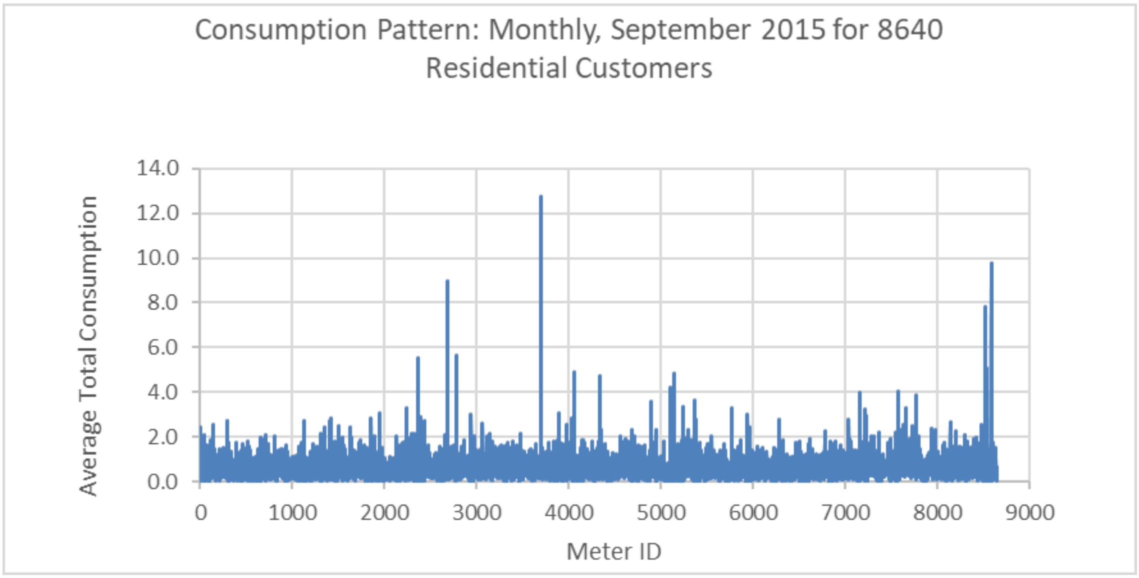


Figure 3.3 Residential Monthly Consumption Pattern

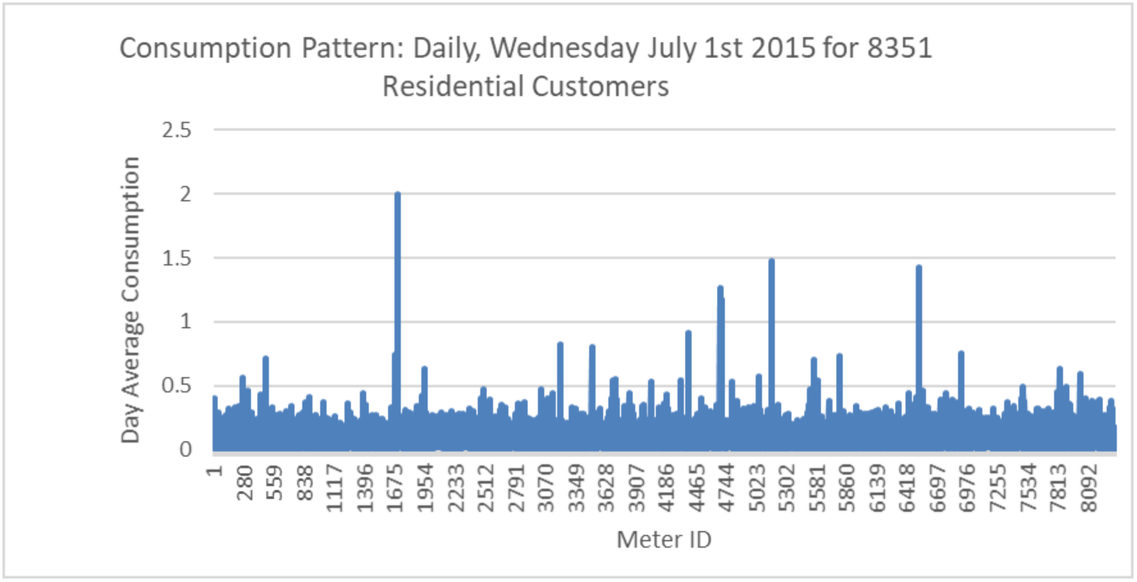


Figure 3.4 Residential Daily Consumption Pattern

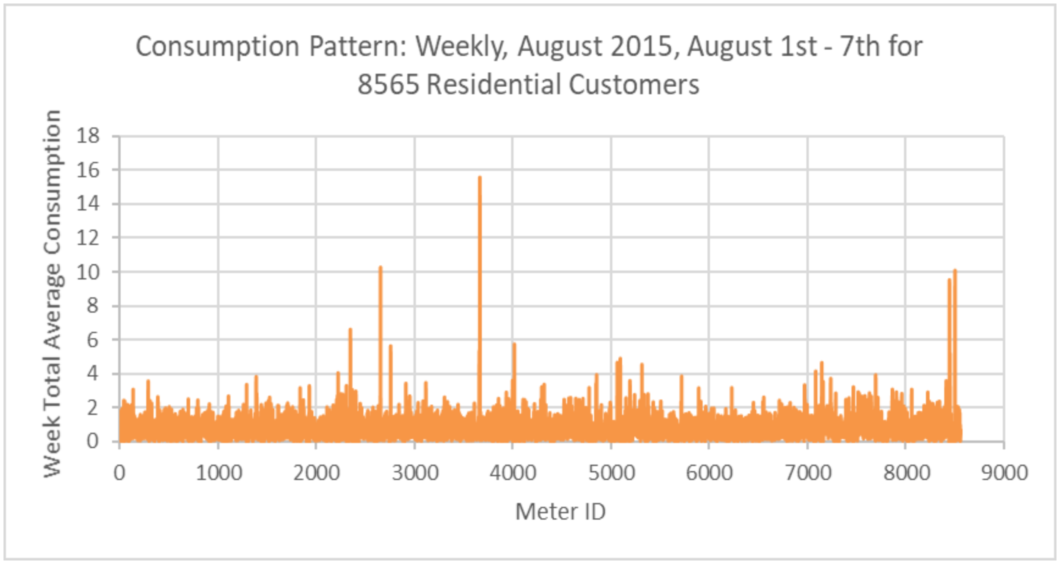


Figure 3.5 Residential Weekly Consumption Pattern

3.3.1 DATA PREPARATION

Smart Meter reads of 18000 electrical meters was provided through the NSERC project in affiliation with SpryPoint Inc. The data was collected at intervals of five minutes'. The daily data collection is around 1.5 GB. Data generated from these meters were initially stored on Amazon S3. The platform for data analysis is as shown in figure 3.5 below. The process of data cleaning and feature selection involves the creation of a local Cassandra database and the transfer of the CSV files stored on Amazon S3 into Cassandra tables. Using a combination of Scala programming interface and the Spark/Cassandra Connection, aggregates were generated from the meter reads and the residential information provided. The aggregates generated were exported into CSV files on the local file system for further analysis.

Using R studio, a data cleaning and pre-processing procedure are applied before the clustering algorithms are used to segment the raw data. These are the steps we followed to prepare data for further analysis:

(i) Filter out weekends and Holidays: We concentrated our analysis on weekdays, which allows us to see stable electricity usage patterns from customers.

(ii) Removed irrelevant variables: There were some zero recordings and inconsistencies, such as reading errors. We removed those customers whose data were not constantly recorded.

(iii) Removed repeated IDs across multiple files: We ensured that there are no repeated IDs (representing each customer) across files.

(iv) Calculated aggregate (mean consumption and total consumption).

(v) For uniform measurement, we normalized the data set.

Fig 3.6 shows the setup of the interface. Cassandra an Apache database system was used for the data processing and feature selection because of the following unique features:

(i) Cassandra is fault tolerant and data can be replicated across multiple clusters.

(ii) It consistently outperforms other popular NoSQL alternatives

(iii)There are no single points of failure

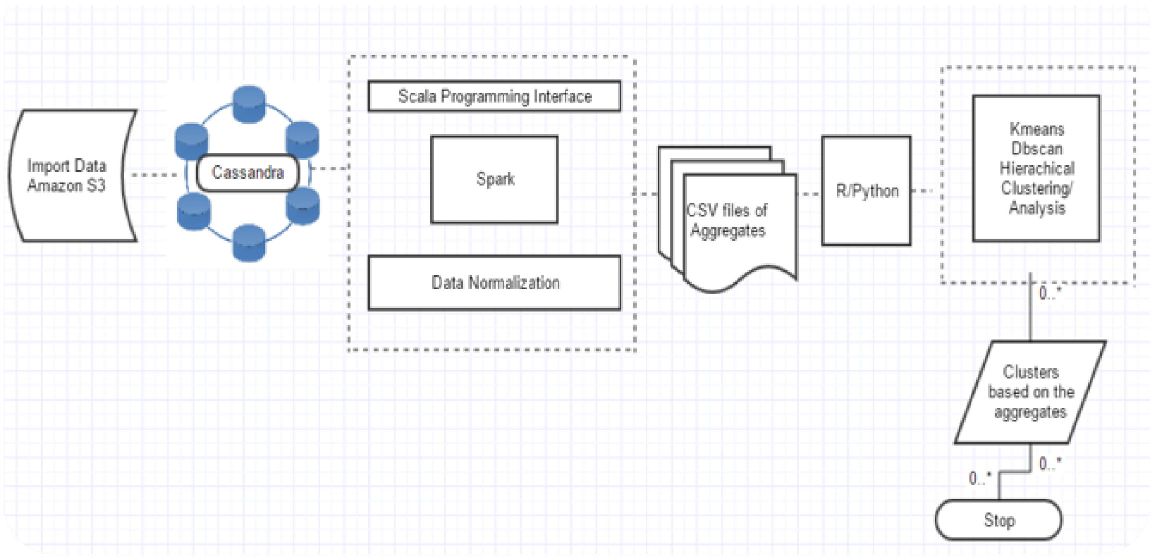

Import Data Amazon S3
Cassandra
Scala Programming Interface
Spark
Data Normalization
CSV files of Aggregates
R/Python
Kmeans Dbscan Hierachical Clustering/ Analysis
0..*
Clusters based on the aggregates
0..*
0..*
Stop

Figure 3.6 Software Architecture of the Proposed System

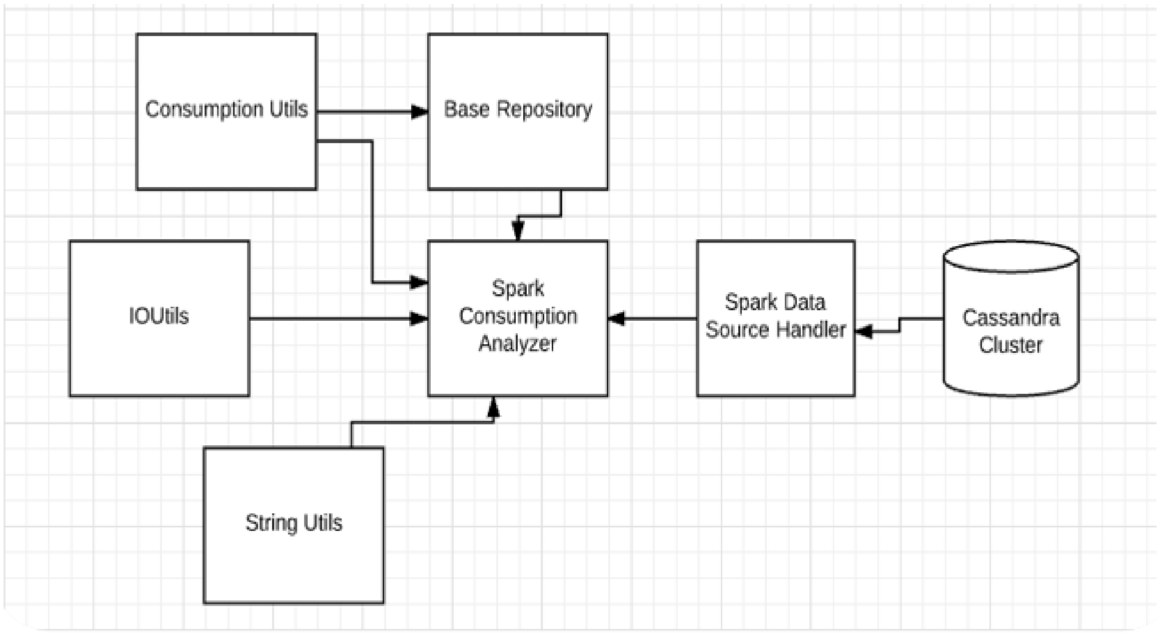


Figure 3.7 System Architecture of the Proposed System

Fig 3.6 shows the components parts of the system architecture. The **Spark data source handler** load data from Cassandra context into a data frame for analyzing consumption data. The **Base Repository** serves as an abstract object that helps store consumption average for each day into a data structure (in this case a List). **Consumption Utils:** Helps Calculate consumption averages or total consumption over a period. The **IOUtils/StringUtils:** provides basic utilities for input/output and string operations respectively. Lastly, the **Spark Consumption Analyzer** serves as the central point for processing data loaded from a Spark Data source.

The data used for this experiment is from the city of Massachusetts. All analysis will be related to the city's calendar. There are two holidays in Massachusetts between November 15th, 2015 and December 15th, 2015. November 11th is the Veterans Day and November 26th, is the Thanksgiving Day. To understand the consumption pattern of customers, weekdays were used and all weekends were excluded from the calculation because we viewed weekends as holidays too. The weekends between those dates are as listed below:

- Sundays: 15th,22nd,29th November
- Saturday: 21st, 28th November
- Sunday, 6th, 13th December

- Saturday 5th, 12th December.

From the data set a total of 10 days are either weekend or holiday until November 15th, and December 15th, 2015. These days were removed from the experiment to allow for uniformity in consumption pattern.

The figures below show the variation in temperature between 15th, November to 15th, December.

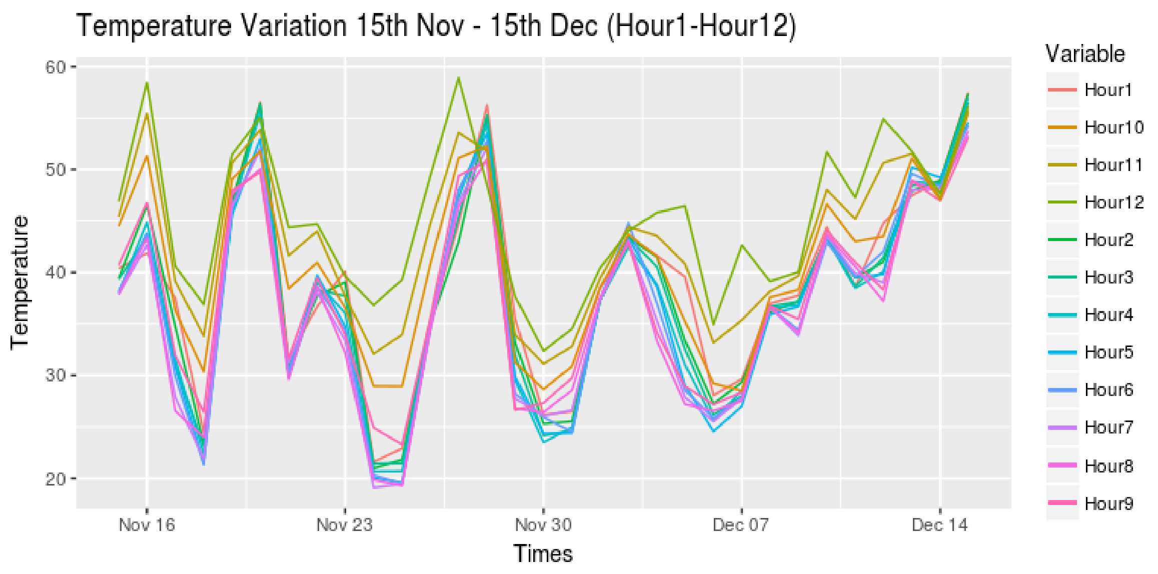


Figure 3.8 Temperature Variation (Hour 1 – Hour 12)

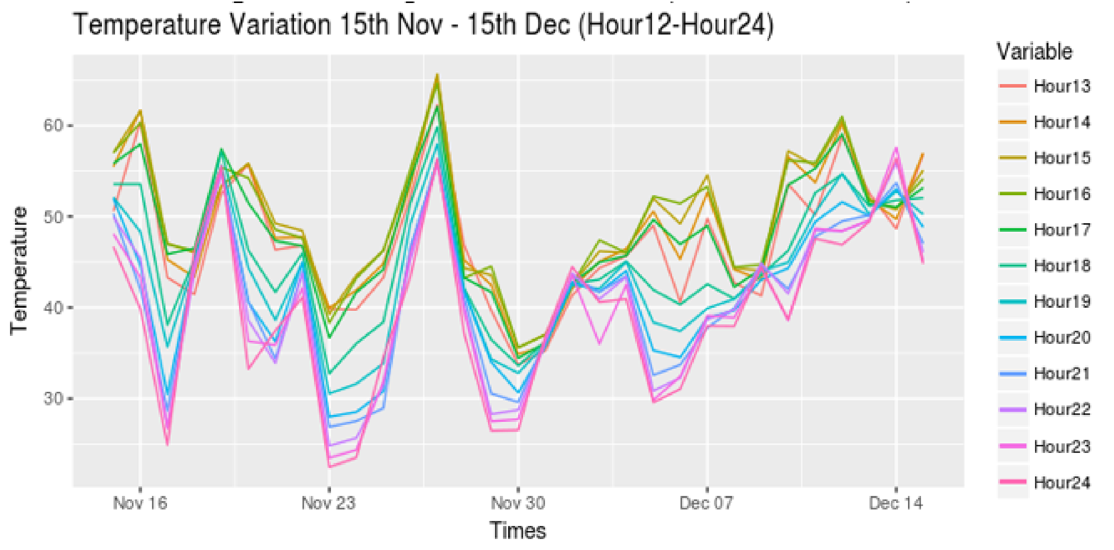


Figure 3. 9 Temperature Variation from (Hour 13 – Hour 24)

The average temperature of each day is as shown in table 3.1

Table 3.1 Average Temperature between Nov 15th – Dec. 15th 2015

S/N	Date	Average Temperature
1	2015-11-15	46.4763
3	2015-11-16	49.2054
4	2015-11-17	34.9187
5	2015-11-18	35.3025
6	2015-11-19	51.2933
7	2015-11-20	49.7988
8	2015-11-21	37.375
9	2015-11-22	42.5063
10	2015-11-23	34.1071
11	2015-11-24	28.9187
12	2015-11-25	31.0229
13	2015-11-26	43.5696
14	2015-11-27	54.1908
15	2015-11-28	47.4942
16	2015-11-29	33.4138
17	2015-11-30	29.385
18	2015-12-01	32.025
19	2015-12-02	40.4058
20	2015-12-03	43.1687
21	2015-12-04	41.9154
22	2015-12-05	37.1812
23	2015-12-06	33.7188
24	2015-12-07	37.2842
25	2015-12-08	39.1575
26	2015-12-09	40.3237
27	2015-12-10	46.3504
28	2015-12-11	46.1221
29	2015-12-12	48.6054
30	2015-12-13	50.1017
31	2015-12-14	50.4154
32	2015-12-15	53.1283

After removing weekends and holidays we obtained the list shown in Table 3.2. Weekends and holiday are adjudged similar in our design. To avoid unnecessary variations in our calculation, we have decided to consider only regular weekdays.

Table 3.2 Average Temperature (Weekdays Only)

S/N	Date	Average Temperature
1	2015-11-16	49.2054
2	2015-11-17	34.9187
3	2015-11-18	35.3025
4	2015-11-19	51.2933
5	2015-11-20	49.7988
6	2015-11-23	34.1071
7	2015-11-24	28.9187
8	2015-11-25	31.0229
9	2015-11-27	54.1908
10	2015-11-30	29.385
11	2015-12-01	32.025
12	2015-12-02	40.4058
13	2015-12-03	43.1687
14	2015-12-04	41.9154
15	2015-12-07	37.2842
16	2015-12-08	39.1575
17	2015-12-09	40.3237
18	2015-12-10	46.3504
19	2015-12-11	46.1221
20	2015-12-14	50.4154
21	2015-12-15	53.1283

In order that we may have a clearer view of the temperature variation from lowest to the highest for the period under consideration, we sorted the days based on non-holiday weekday and their corresponding consumption is as shown in table 3.3. The Temperature is represented in abbreviation as T1…T21 which corresponds to the corresponding values 28.9187…54.1908 of temperature in increasing order

Table 3.3 Temperature sorted in order(ascending)

S/N	Date	Temperature
1	2015-11-24	28.9187 (T1)
2	2015-11-30	29.385 (T2)
3	2015-11-25	31.0229 (T3)
4	2015-12-01	32.025 (T4)
5	2015-11-23	34.1071 (T5)
6	2015-11-17	34.9187 (T6)
7	2015-11-18	35.3025 (T7)
8	2015-12-07	37.2842 (T8)
9	2015-12-08	39.1575 (T9)
10	2015-12-09	40.3237 (T10)
11	2015-12-02	40.4058 (T11)
12	2015-12-04	41.9154 (T12)
13	2015-12-03	43.1687 (T13)
14	2015-12-11	46.1221 (T14)
15	2015-12-10	46.3504 (T15)
16	2015-11-16	49.2054 (T16)
17	2015-11-20	49.7988 (T17)
18	2015-12-14	50.4154 (T18)
19	2015-11-19	51.2933 (T19)
20	2015-12-15	53.1283 (T20)
21	2015-11-27	54.1908 (T21)

The difference in temperature from the coldest day in the period under consideration to the warmest day under the same period i.e. the temperature variation is as shown in table 3.4. below. The temperature increase in table 3.4 is calculated using T_I = T_high – T_lowest, the lowest temperature T1 is removed from each of the corresponding higher temperature for each day as shown in table 3.4

Table 3. 4 Temperature Variations

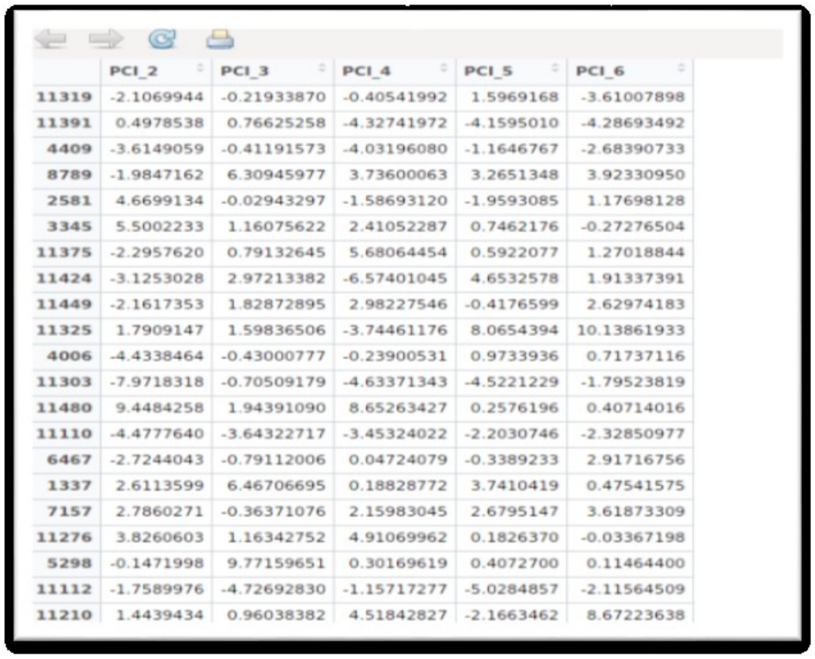
Serial Number	Temperature increase (T_I)
1	0.4663
2	2.1042
3	3.1063
4	5.1884
5	6
6	6.3838
7	8.3655
8	10.2388
9	11.405
10	11.4871
11	12.9967
12	14.25
13	17.2034
14	17.4317

Serial Number	Temperature increase (T_I)
15	20.2867
16	20.8801
17	21.4967
18	22.3746
19	24.2096
20	25.2721

We extracted the power consumption(PC) at times T1…T21 and the corresponding PCI values is also calculated using the following equations:

PC (T2) - PC (T1) = PCI_1
PC(T3) - PC (T1) = PCI_2
.
.
.
PC (T21) - PC (T1) = PCI_20

The data generated for the corresponding consumption retrieved based on the above days and temperature values from the data in R studio is as shown in figure 3.10



	PCI_2	PCI_3	PCI_4	PCI_5	PCI_6
11319	-2.1069944	-0.21933870	-0.40541992	1.5969168	-3.61007898
11391	0.4978538	0.76625258	-4.32741972	-4.1595010	-4.28693492
4409	-3.6149059	-0.41191573	-4.03196080	-1.1646767	-2.68390733
8789	-1.9847162	6.30945977	3.73600063	3.2651348	3.92330950
2581	4.6699134	-0.02943297	-1.58693120	-1.9593085	1.17698128
3345	5.5002233	1.16075622	2.41052287	0.7462176	-0.27276504
11375	-2.2957620	0.79132645	5.68064454	0.5922077	1.27018844
11424	-3.1253028	2.97213382	-6.57401045	4.6532578	1.91337391
11449	-2.1617353	1.82872895	2.98227546	-0.4176599	2.62974183
11325	1.7909147	1.59836506	-3.74461176	8.0654394	10.13861933
4006	-4.4338464	-0.43000777	-0.23900531	0.9733936	0.71737116
11303	-7.9718318	-0.70509179	-4.63371343	-4.5221229	-1.79523819
11480	9.4484258	1.94391090	8.65263427	0.2576196	0.40714016
11110	-4.4777640	-3.64322717	-3.45324022	-2.2030746	-2.32850977
6467	-2.7244043	-0.79112006	0.04724079	-0.3389233	2.91716756
1337	2.6113599	6.46706695	0.18828772	3.7410419	0.47541575
7157	2.7860271	-0.36371076	2.15983045	2.6795147	3.61873309
11276	3.8260603	1.16342752	4.91069962	0.1826370	-0.03367198
5298	-0.1471998	9.77159651	0.30169619	0.4072700	0.11464400
11112	-1.7589976	-4.72692830	-1.15717277	-5.0284857	-2.11564509
11210	1.4439434	0.96038382	4.51842827	-2.1663462	8.67223638

Figure 3.10 Sample Power Consumption Data for 04th Dec and 15th Dec. 2015

PCI_11 and **PCI_20** are the power consumption values for 4th December and 15th December 2015. It is the data used to cluster and test the proposed algorithms.

3.3.2 DETAILS OF THE PROPOSED ALGORITHMS

The algorithms under investigation have been divided into four major methods for ease of comparison. The first approach, KM runs K-means on the entire data set. The second approach (KD) runs k-means on the data set and uses DBSCAN to identify the anomalies of the data set. Our major contribution starts from the third algorithm that is called KDS, which is based on K-means, DBSCAN with SMALL Training set KNN. KDS is presented in Algorithm 3.

Algorithm 2: KD

Input: A time-series data set

Parameters: Number of clusters k, minPoints, epsilon

Output: Clusters

(i) Identify number of clusters k

(ii) Clean each cluster identified (Use DBSCAN)

We observed that because K-Means is not sensitive to outliers the clusters generated may contain anomalies. The main motivation for this algorithm (KD) is to obtain clean clusters from the clusters generated from K-Means.

Algorithm 3: KDS

Input: A time-series data set/ Sample data set (1%,2%,5% etc)

Parameters: Number of cluster *k*

Output: Clusters

(i) Identify number of clusters k

(ii) Partition the sample data set into k clusters

(iii) Remove outliers from each cluster k (Clean Cluster)

(iv) Obtain the centroid of each cleaned cluster k

- o *Note that the centroids are the centroids of the clusters with no outliers. For each cleaned cluster, the average of the x coordinates in the cluster will be the x coordinate of the cluster centroid; the average of the y coordinates will be the y coordinate of the cluster centroid.*

- *Calculate the mean values of the x and y coordinates*
- $Mean_x = \frac{1}{n}\sum_{i=1}^{n} x_i$ *and* $Mean_y = \frac{1}{n}\sum_{i=1}^{n} y_i$

(v) Select (1%, 2%, 5%) sample from each of the clean cluster obtained in (iv) above

(vi) Combine (iv and v) the centroid and the selected sample to create the training set.

(vii) Generate a KNN Model using (vi) as the training set and Algorithm 3.

(viii) Use the Model in (vii) to classify the rest of the dataset.

(ix) Stop.

In this thesis, we were motivated by the fact that although K-Means works well with great speed on large data set, the cluster re-assignment step of the algorithm can be quite demanding in terms of running time, especially for the very large data set. It may thus slow down the convergence of the algorithm. Instead of having K-Means go through the whole data set at once, the proposed algorithm (KDS and KDSD) first select a few sample points from the data set and divide them into clusters. Thereafter, the proposed algorithms allocate each of the remaining data points to one of the previously generated clusters according to the distance to these clusters. This can not only correctly classify the rest of the data set but also be explored by a density based outlier-aware approach to remove possible anomalies. The idea is to efficiently identify points that belongs to the different clusters (cluster pre-assignment) in the data set from initial sub-sampled data through a combination of centroid and small sample of clean cluster member (Cluster pre-processing) and then use the pre-processed cluster as training set to re-assign other members in the unlabelled data set using Nearest Neighbour Algorithm. The clusters generated from the unlabelled data set are not clean because they were only allocated based on the data from the training example, hence the need to finally run DBSCAN for cleaning the final dataset in the case of KDSD algorithm. The details of KDSD are summarised in Algorithm 4.

```
Algorithm 4: KDSD
Input: A time-series data set/ Sample data set (1%,2%,5% etc)
Parameters: Number of cluster k
Output: Clusters
    (i)  Run KDS
    (ii) Remove outlier from the classified data set (Generated Clusters)
    (iii)Stop
```

The details of SKNN are summarised in Algorithm 5.

```
Algorithm 5: SKNN (Small Training Set K Nearest Neighbor)
Input: x: unknown/unlabeled data set
Output: Yi: clusters with labels
Classify (X, Y, x)//
X: randomly select {1%, 2% or 5% plus average point in each cluster i.e centroid}
Y: Class labels for the selected sample and the centroids
x: unlabeled data set
for i=1 to m do
compute
    $\text{ditance } (X,x) = \sqrt{\sum_i (X_i - x_i)^2}$ between the points
end for
Compute set I containing indices for the k smallest distance d
Return majority labels for {Yi where i belongs to I}
```

The diagram below shows the various stages in the algorithmic process

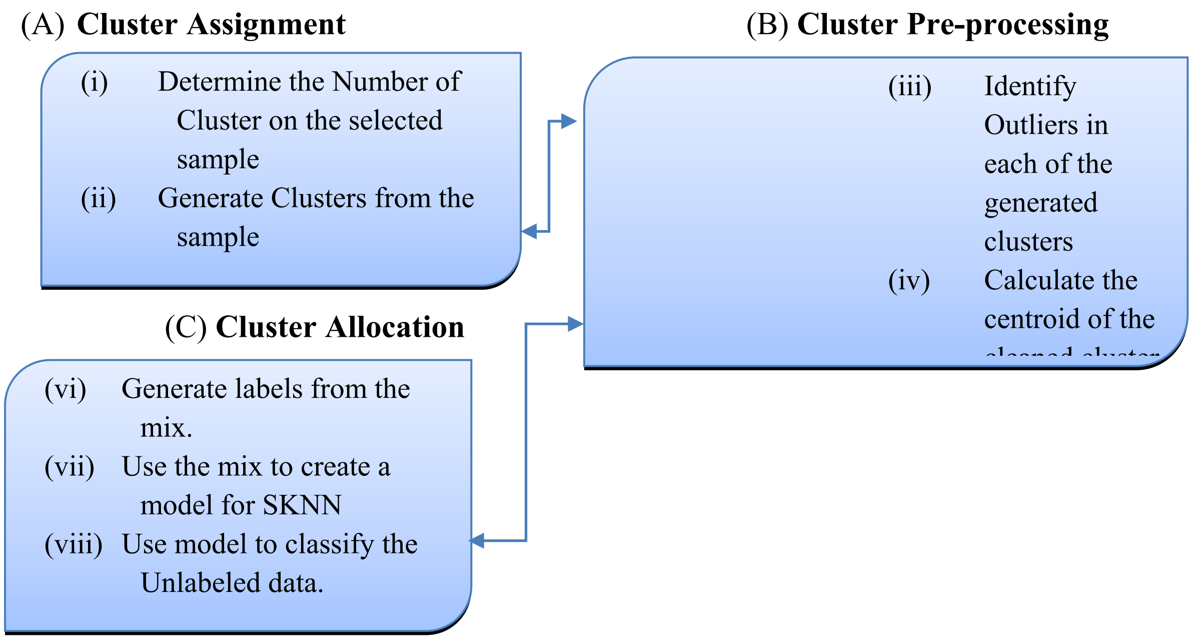

(A) Cluster Assignment
(i) Determine the Number of Cluster on the selected sample
(ii) Generate Clusters from the sample
(B) Cluster Pre-processing
(iii) Identify Outliers in each of the generated clusters
(iv) Calculate the centroid of the
(C) Cluster Allocation
(vi) Generate labels from the mix.
(vii) Use the mix to create a model for SKNN
(viii) Use model to classify the Unlabeled data.

Figure 3.11 ***Overview of the Algorithmic process***

3.3.3 COMPLEXITY AND RUN-TIME ANALYSIS OF THE ALGORITHMS

To estimate the performance of the algorithms based on storage, amount of time taken to run the algorithm coupled with other resources the algorithm will need for its execution; we considered the big O notation of each of the corresponding algorithms used above on an initial 1% sample (2000) and 1% clean cluster plus the centroid for the SKNN. K means algorithm was run on over 30 iterations, in two dimensions (PCI_11 and PCI_20) for two clusters i.e k =2. The run time for the K-means algorithm is O(*iknd*) where the *i*= number of iterations, *k*=number of clusters, *n*=number of points and d=dimensions. DBSCAN, on the other hand, has a running time complexity of O(nlogn). To reduce the runtime, a KD-tree implementation of the algorithm in C++ in the package DBSCAN (CRAN) was chosen as our preferred alternative. KD-tree is a space-partitioning data structure for organizing points in k-dimensional space. It has two nodes and it splits the k-dimensional points into a binary tree where every leaf node is associated with one of the k-dimensions. K-distance graph was used to enhance the search for the appropriate value of epsilon. The time complexity of this improved DBSCAN is O(logn). The KD-Tree implementation has been demonstrated to increase both the clustering efficiency and speed. It has also been adjudged as the best implementation for DBSCAN[54]. KNN, unlike the two other algorithms, is a supervised learning method where the result of new unlabeled data is classified based on majority vote of the nearest neighbor category. The algorithm consists of two phases the training and the test phase. The algorithm is said to be a lazy learning algorithm because the 1% training phase runs linearly, only the implementation of the actual allocation of the unlabeled data takes O(logn). We can, therefore, conclude that KDS will run at most at O(logn), similarly, KDSD will run O(logn).

3.3.4 SAMPLING TECHNIQUE

The sampling process is multi-staged. Initially, the simple random technique was used to have a good representation of the population. One major advantage that we explored at this stage is that we want every member of the population to be equally-likely represented in the clustering process. After determining the possible cluster in the population, we selected sample from each cluster to serve as representative of each cluster initially identified, to prevent the population from been under-represented or over-represented. A clean copy of the targeted cluster characteristics was the main population from where the selection was made. This approach increases the chance of arriving at a perfectly classified data set using a modified Nearest Neighbor Algorithm called SKNN.

CHAPTER 4 EXPERIMENTAL RESULTS

This chapter presents the details of the algorithm implementation and the experimental results. The primary goal of the study is to design an algorithm that is fast and able to effectively detect the anomalies in large data sets. Our experimental results show that the goal has been achieved.

4.1 EXPERIMENTAL DESIGN

This section describes the experiment performed on the four algorithms: KM, KD, KDS, and KDSD. The data was prepared from CSV files of raw consumption data using Scala programming language, R and a few Python scripts. The data used for this experiment consists of 200,000 households simulated from initial 12000 households provided using the following steps:

We generated from each data point (x, y) in the 12000-element data set, a new data point (x1, y1). Specifically, $x1 = x + \delta x$ and $y1 = y + \delta y$. δx is a random number in the range of $(-x + 2\%, x + 2\%)$ and δy is a random number in the range of $(-y + 2\%, y + 2\%)$. The choice of δx and δy is based on the data set under consideration. The values may differ for a different data set. We observed that there was too much overlap in the data set, we decided to separate the data set into odd-numbered and even-numbered data, for the odd-numbered meters we added 200 points to have two separate clusters. The resulting data set consist of nearly equal number of odd-numbered meters and even-numbered meters. The rest of the section describes the result obtained from the algorithm. Two days in winter were selected (the warmest day and the coldest day) as the feature. Consumption on the warmest temperature is plotted against coldest temperature in a two-dimensional graph. The result of the experiment is as shown below.

4.2 EXPERIMENTAL RESULTS OF KM

The entire data set (200,000 households) was generated through data pre- processing. Specifically, the data was separated into two clusters to include those users whose consumption increases with temperature and the other includes users whose consumption does not increase much with temperature. For odd-numbered meters, we added a constant value (delta= 200) to x and y axis, so that we could clearly generate two separate clusters. Initially, we identified the number of clusters using elbow method [55]. Elbow method help to determine an appropriate number of clusters in an unlabeled data set. It uses the percentage of variance to estimate the number of clusters in a dataset. It is the point at which any addition of extra cluster to the once already identified will lead to convergence in the number of clusters identified. This point often represented by an angle in the plot of variance as a function of the number of clusters lead to an elbow-like point in the curve, hence the name "elbow curve". The massive data set generated from the meters coupled with little knowledge we had about the data makes an appropriate value of *k* not apparent from mere studying the features of the data set. To give us a clearer picture of the appropriate number of clusters to be generated. We used the Elbow method: The figure indicates that adding another cluster after 2 (two) does not generate any significant change in the data.

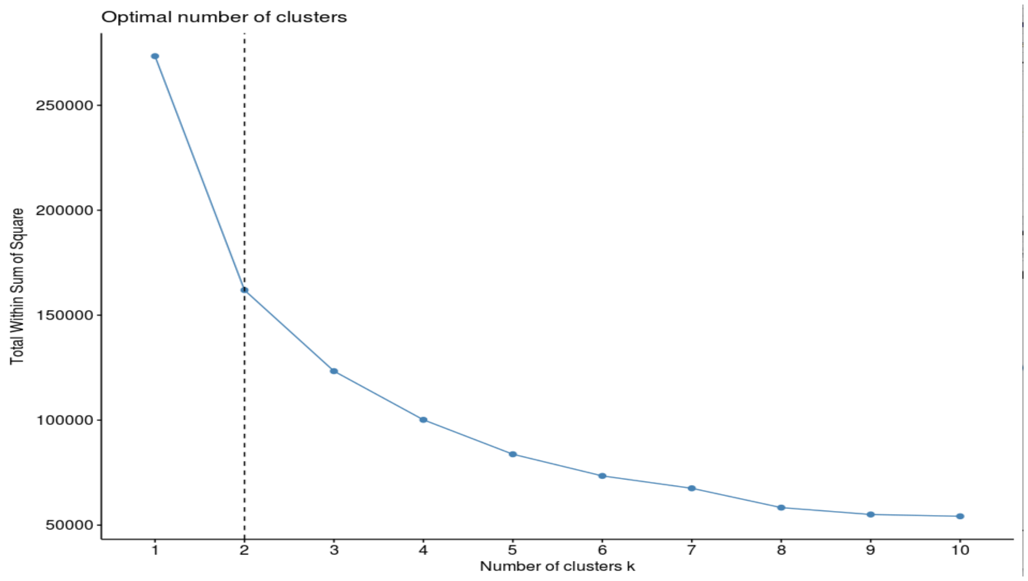


Figure 4.1 Elbow curve to determine the number of clusters

The time taken to run KMEANS on the whole data set = **7.56 seconds**

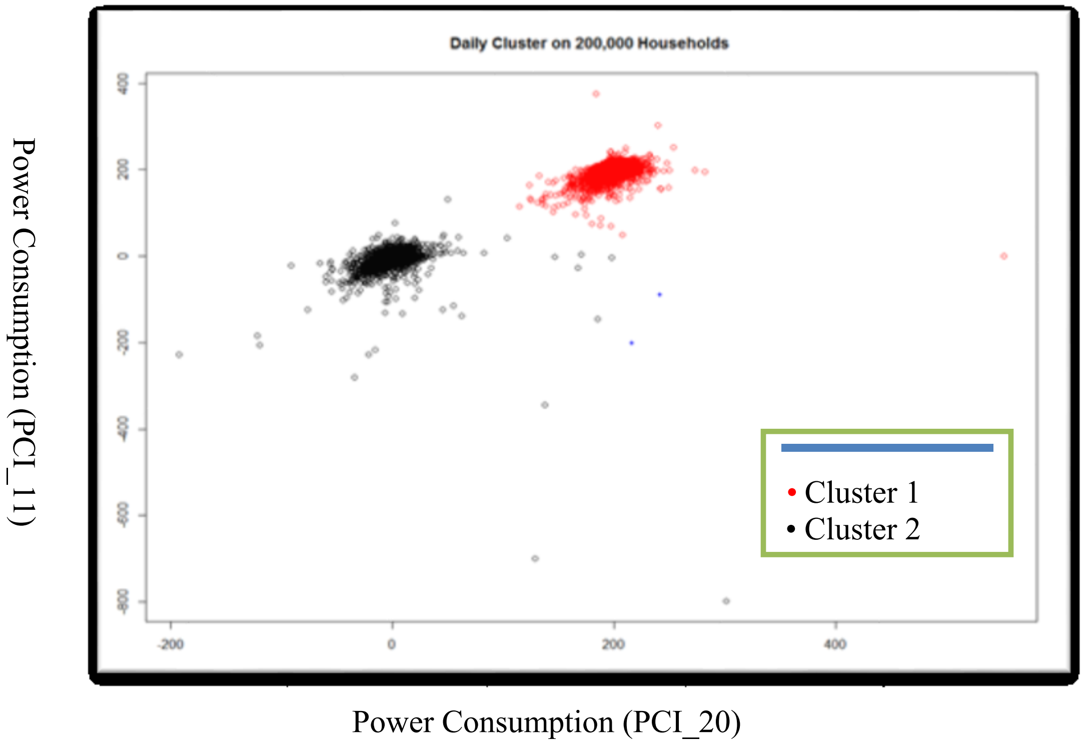


Figure 4.2 Result of K-Means Algorithm on the whole dataset

4.3 EXPERIMENTAL RESULTS OF KD

DBSCAN was used to identify anomalies in the cluster generated above. The result from the two cluster shows that Cluster I have 91744 households and cluster II has 108256 Households. The rest of the result and the corresponding time taken to run DBSCAN on each of the clusters are shown below:

Table 4. 1 DBSCAN Anomaly Detection on Cluster

Cluster Name	Cluster size	Clean Cluster	Noise points	Time Taken
Cluster 1	91744	91439	305	2.039877 minutes
Cluster 2	108256	107999	257	2.310681 minutes

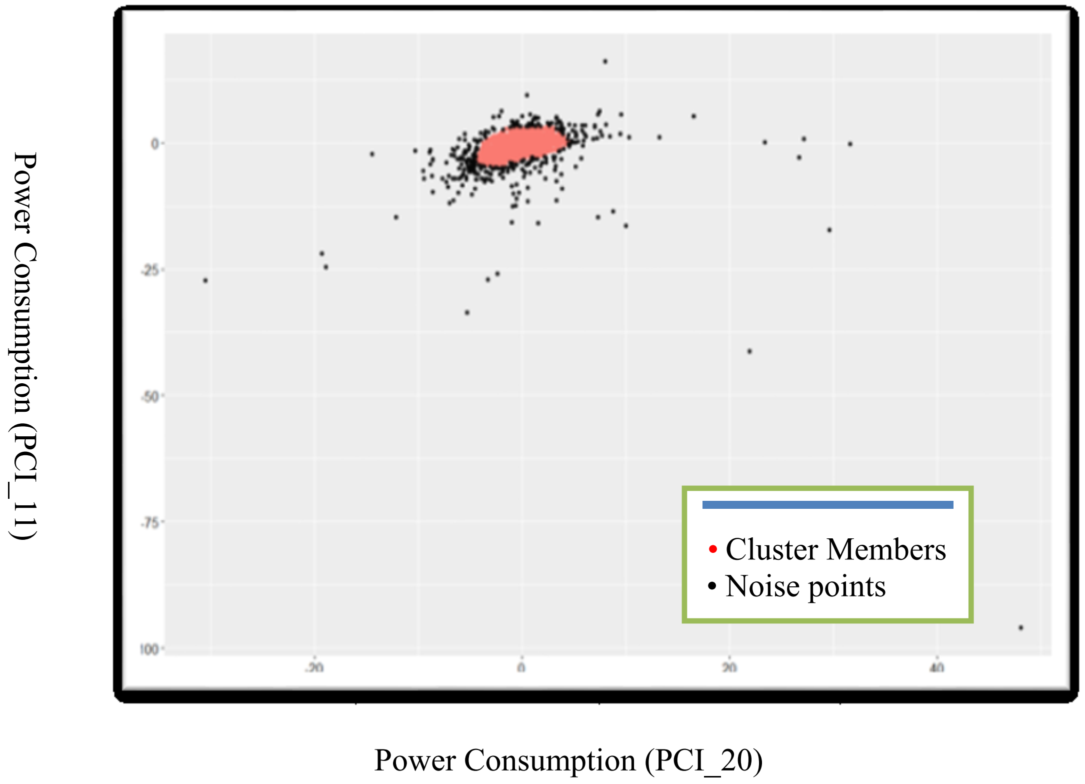


Figure 4. 3 Outlier Detection on Cluster 1: 91744

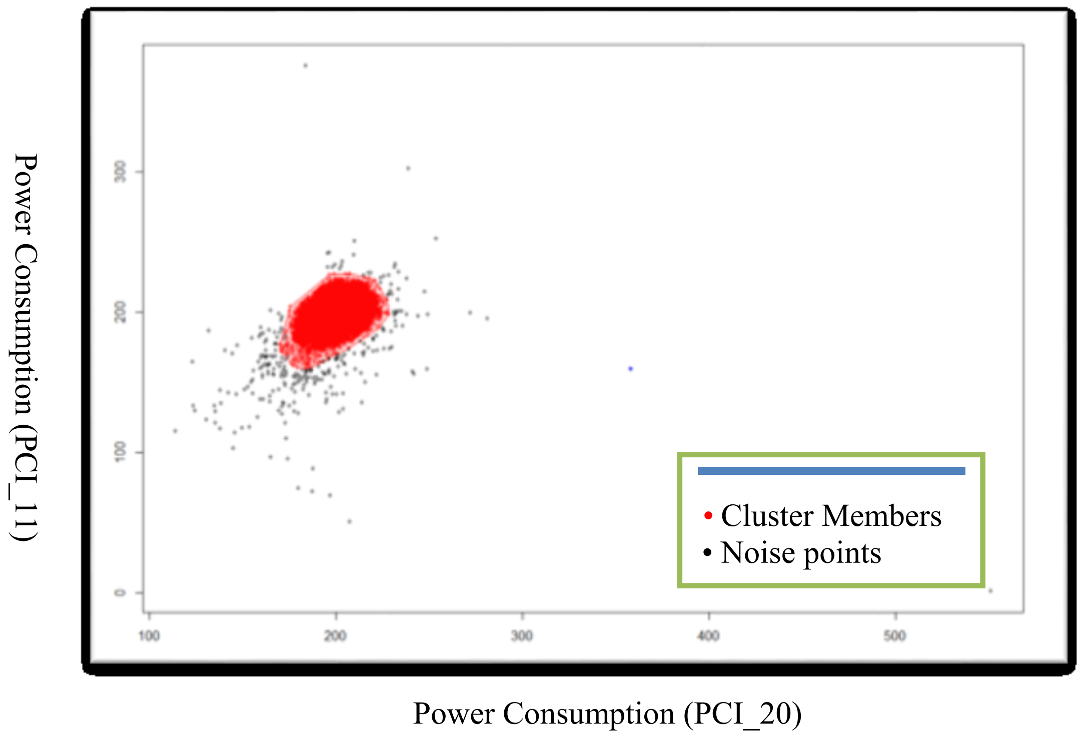


Figure 4.4 Outlier Detection on Cluster II: 108256

4.4 EXPERIMENTAL RESULTS OF KDS AND KDSD

As stated earlier in the algorithm in chapter 3, a small training sample of 1%, 2% or 3% sample will be used to train and classify the data set. We randomly select the sample from the 200,000 households.

4.4.1 KDS AND KDSD WITH 1% SAMPLE

Two thousand samples which represent 1% of the household was initially selected for the experiment, the result is as shown in table 4.2. The time taken by K-means is
0.089 seconds.

Table 4.2 Outlier Detection 1% Sample

Cluster Name	Cluster Size	Cleans Cluster	Noise Points	Time Taken
Cluster 1	882	853	29	0.01 seconds
Cluster 11	1118	1067	51	0.015 seconds

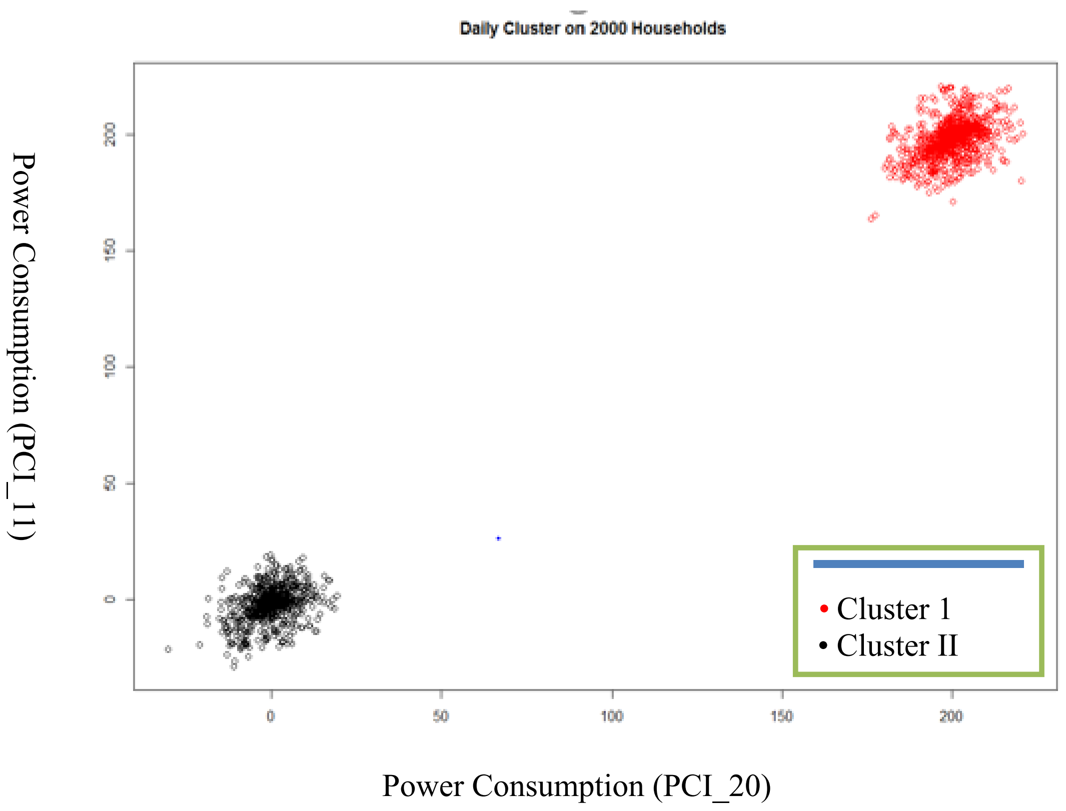


Figure 4.5 Result of K-means on 1% sample

From table 4.2 the total time taken to clean cluster I and II are respectively, 0.01sec and 0.015 seconds. The noise points are 29 and 51 for cluster I and cluster II respectively.

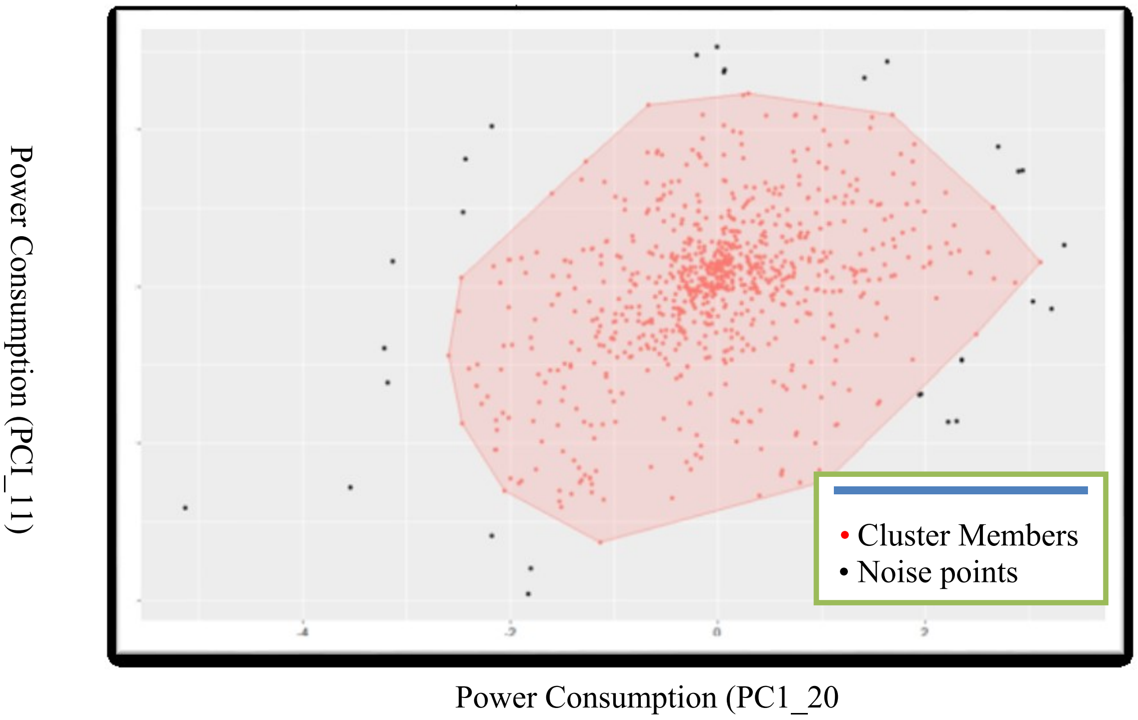


Figure 4.6 Outliers on cluster I using DBSCAN

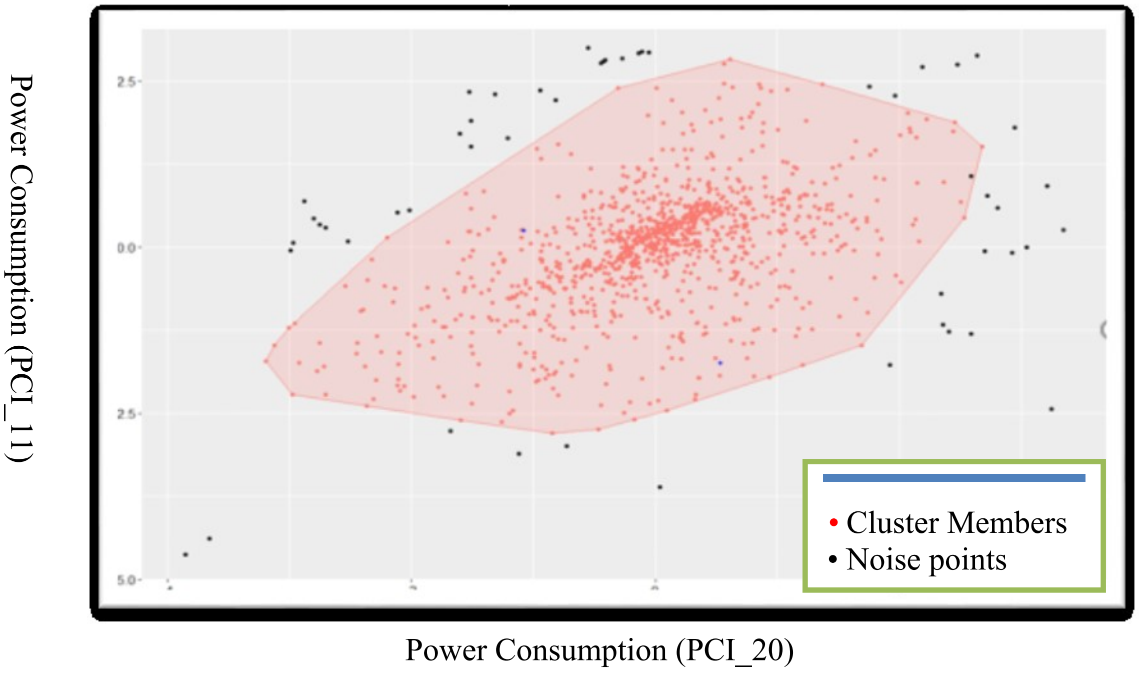


Figure 4.7 Outlier on Cluster II using DBSCAN

KNN on the cleaned Clusters

The next step in the algorithm is to select clean samples from the two clean clusters: 853 and 1067. From each of this clean cluster 1%, 2% and 5% each will be selected and added to the centroids obtained from the axis. The 1% sample will be 0.01* 853 = 9 and 0.01* 1067= 11 approximately. A total of 20 data-points plus the two data-points each from the centroids; gives 22 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.112 seconds. Similar result will be calculated for 2% sample and 5% sample of the data set

Remaining sample = 198000

Table 4.3 Predicted Cluster from remaining Sample (198000)

Cluster name	Predicted Cluster	Mean	Median	1st Quartile	3rd Quartile
Cluster I	90783	0.4042	0.5158	0.2584	0.5288
Cluster II	107217	0.7706	0.8397	0.6791	0.8500

Table 4.4 Time taken to Clean the remaining Sample

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	90783	90491	292	1.929 mins
Cluster 11	107217	106869	348	2.015 mins

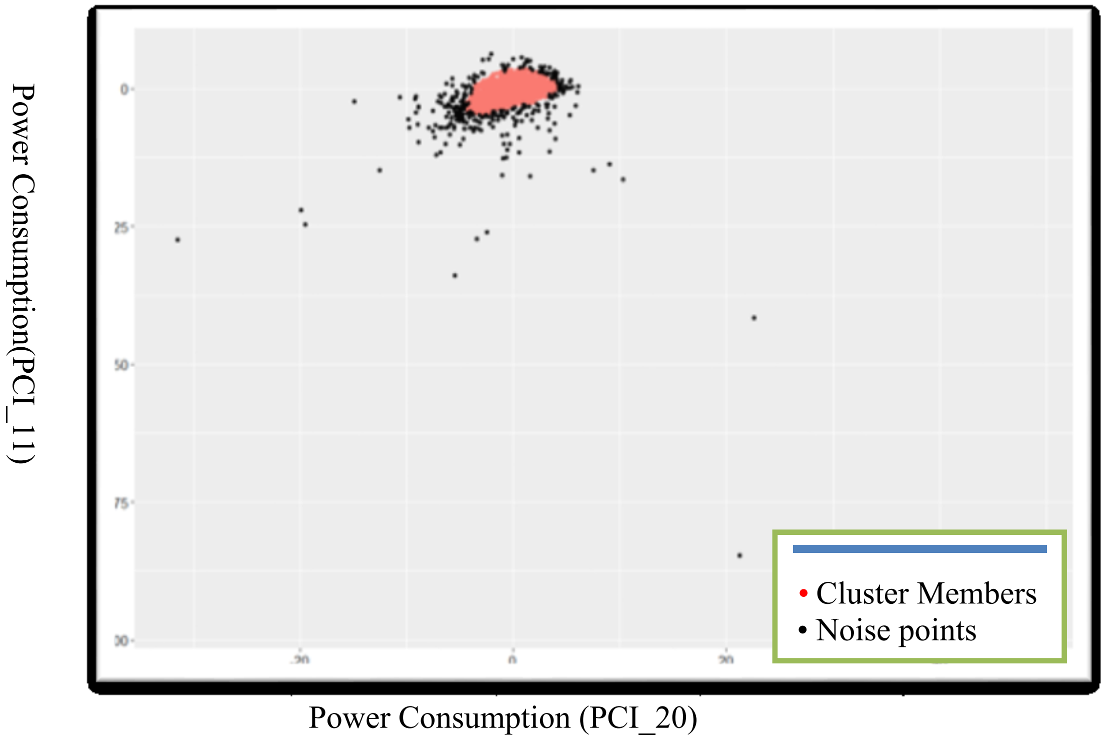


Figure 4.8 Outlier on Cluster I using DBSCAN: 90491

SKNN WITH 2% Sample

The 2% sample will be 0.02* 853 = 18 and 0.02* 1067= 22 approximately, which gives 40 data-points plus the two data-points each from the centroids; in total, we have 42 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.134 secs. A similar result will be calculated for 5% sample.

Table 4.5 Predicted Clusters Using SKNN

Cluster name	Predicted Cluster	Mean	Median	1st Quartile	3rd Quartile
Cluster I	90617	0.4042	0.5158	0.2584	0.5288
Cluster II	107383	0.7706	0.8397	0.6791	0.8500

Table 4.6 Time Taken to Clean the remaining Clusters

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	90617	90377	240	1.905 mins
Cluster 11	107383	106821	562	2.098 mins

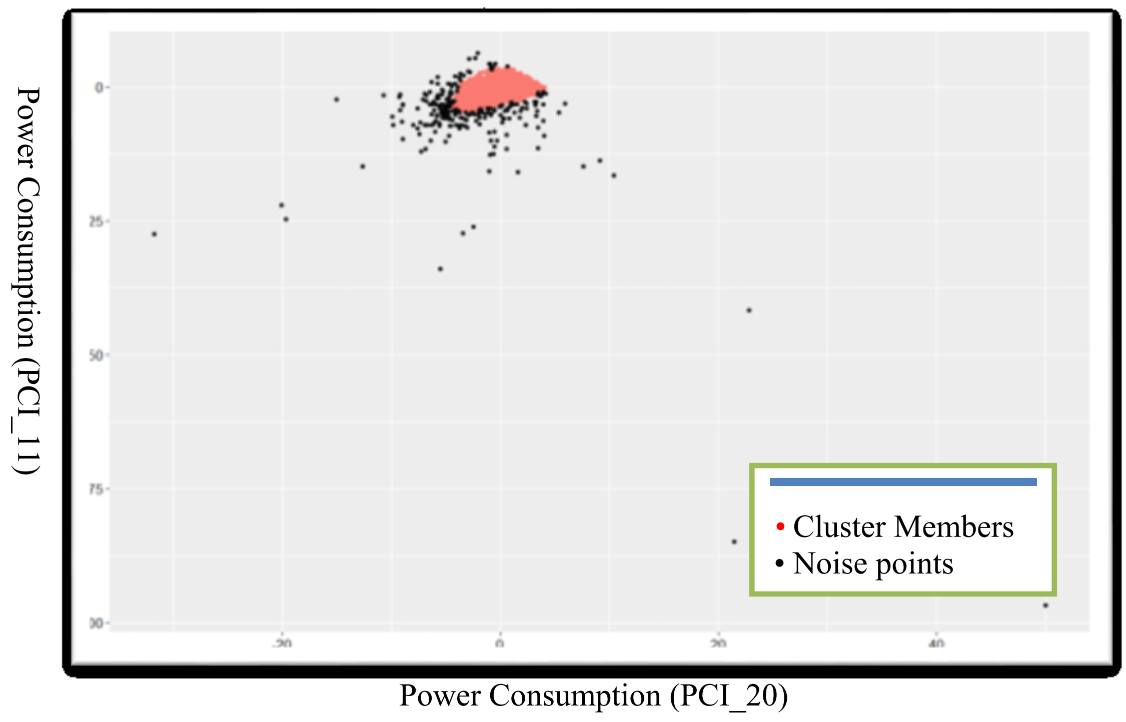
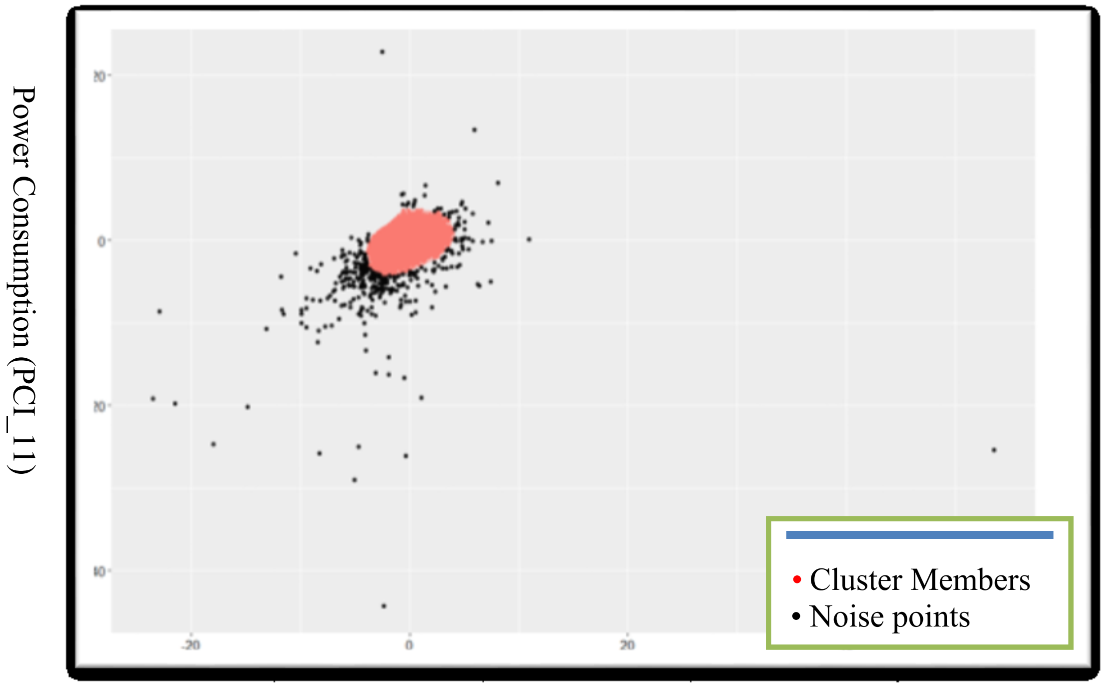


Figure 4.9 Outlier on Cluster I using DBSCAN:90617



Power Consumption (PCI_20)

Figure 4. 10 Outlier on Cluster II using DBSCAN:107383

SKNN WITH 5% Sample

The 5% sample will be 0.05* 853 = 43 and 0.05* 1067= 54 approximately. We have 97 data-points plus the two data-points each from the centroids; in total, we have 99 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.184 secs.

Table 4.7 Predicted Cluster with 5% Sample SKNN

Cluster name	Predicted Cluster	Mean	Median	1st Quartile	3rd Quartile
Cluster I	90783	0.4042	0.5158	0.2584	0.5288
Cluster II	107217	0.7706	0.8397	0.6791	0.8500

Table 4.8 Time taken to clean the remaining Cluster

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	90783	90491	292	1.959 mins
Cluster 11	107217	106869	348	2 mins

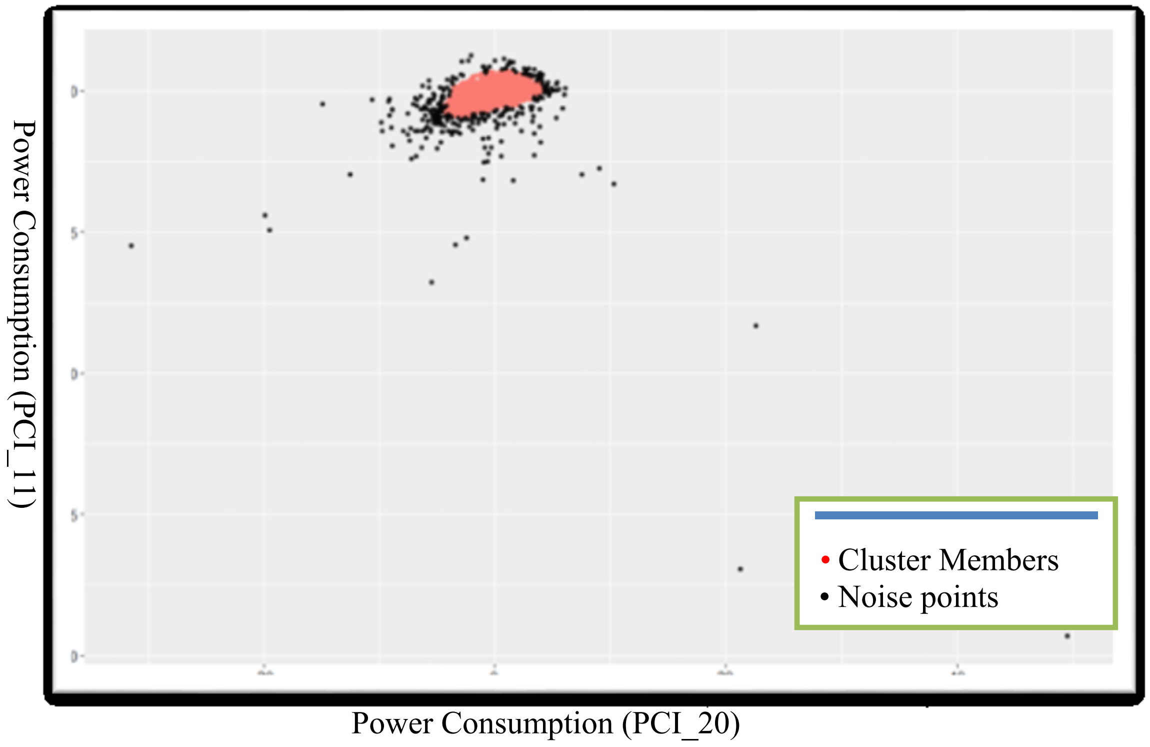


Figure 4. 11 Outlier on Cluster I using DBSCAN:90783

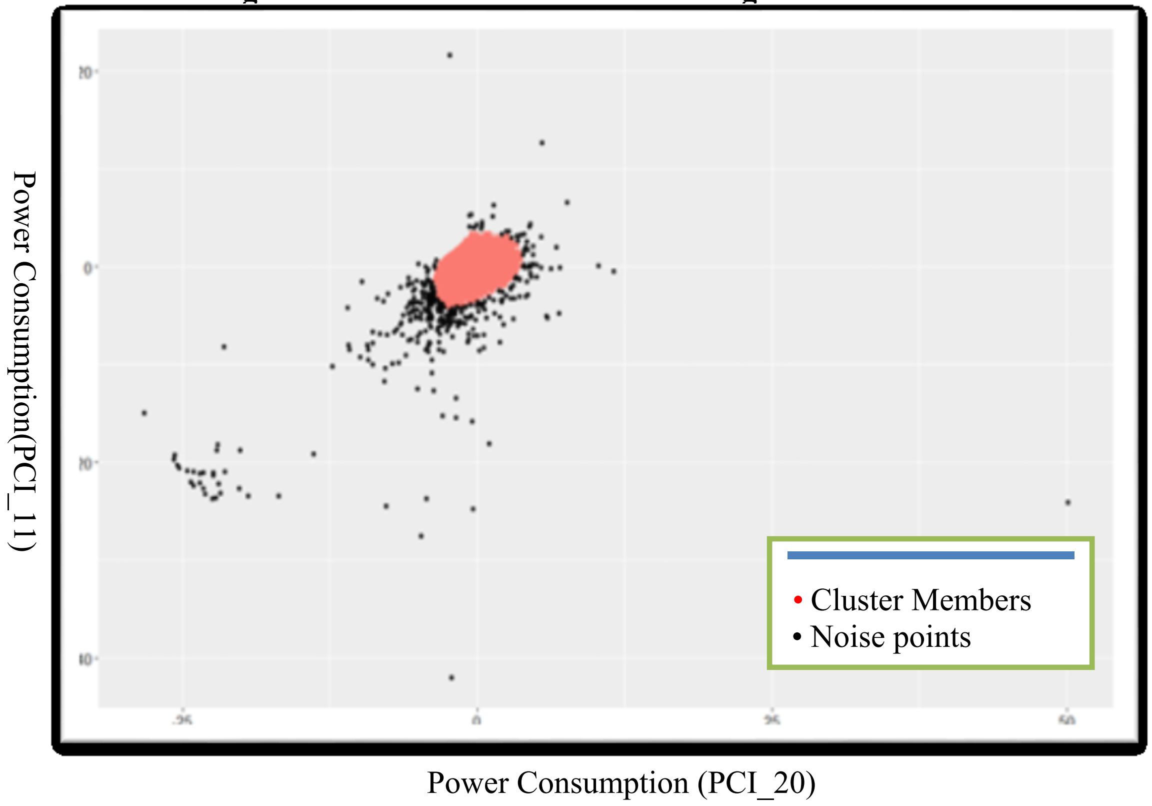


Figure 4.12 Outlier on Cluster II using DBSCAN:107217

4.4.2 KDS AND KDSD WITH 2% SAMPLE

We now repeat the experiment with 2% (4000 households) of the total sample. Time taken to run K-means on the selected sample is 0.159 secs

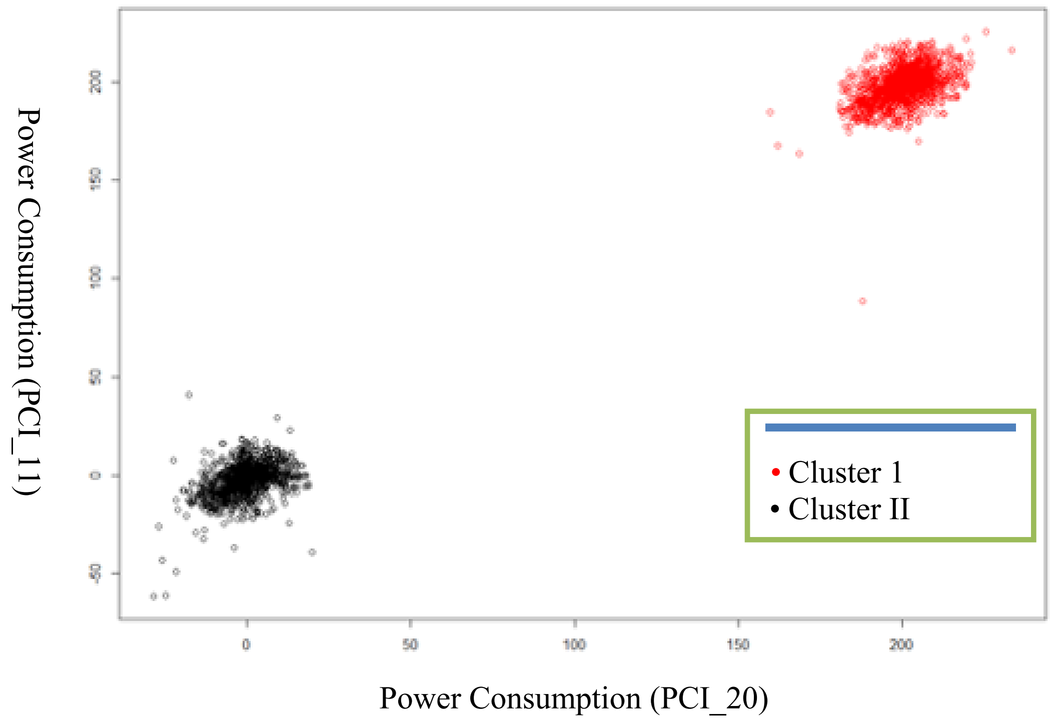


Figure 4.13 Result of K-means on 2% sample

Table 4.9 Initial Cluster from 2% Sample

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	1822	1795	27	0.028 secs
Cluster 11	2178	2157	21	0.037 secs

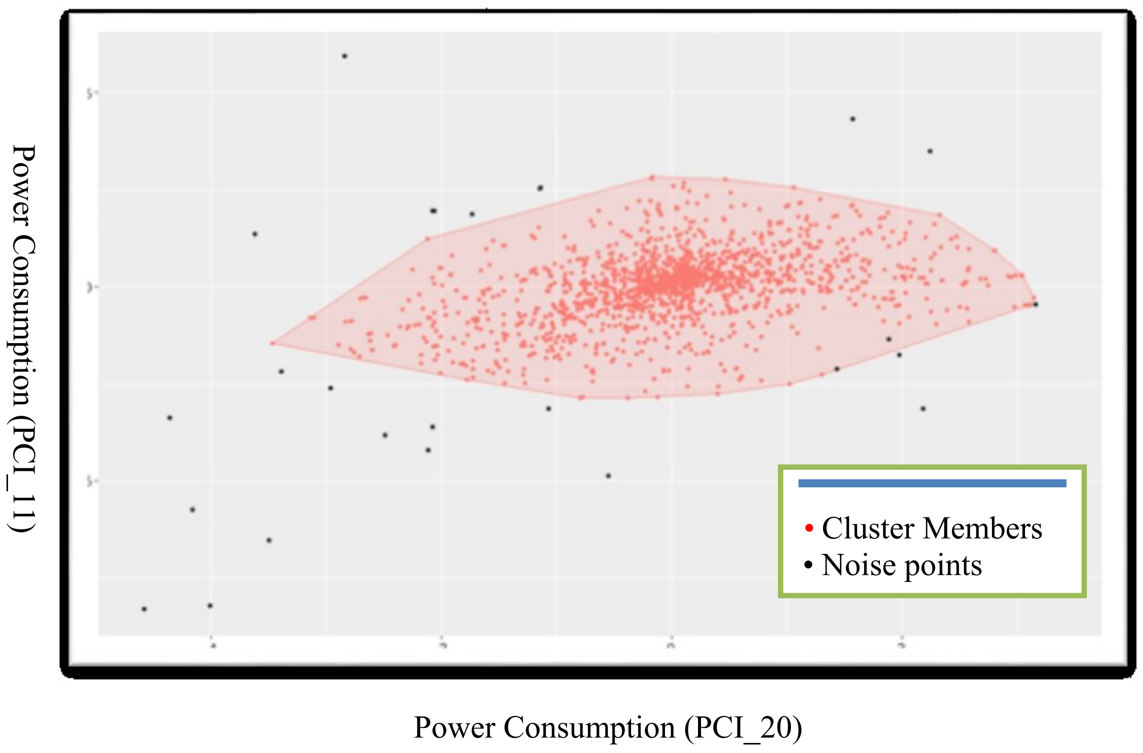


Figure 4.14 Outlier on Cluster I using DBSCAN:1822

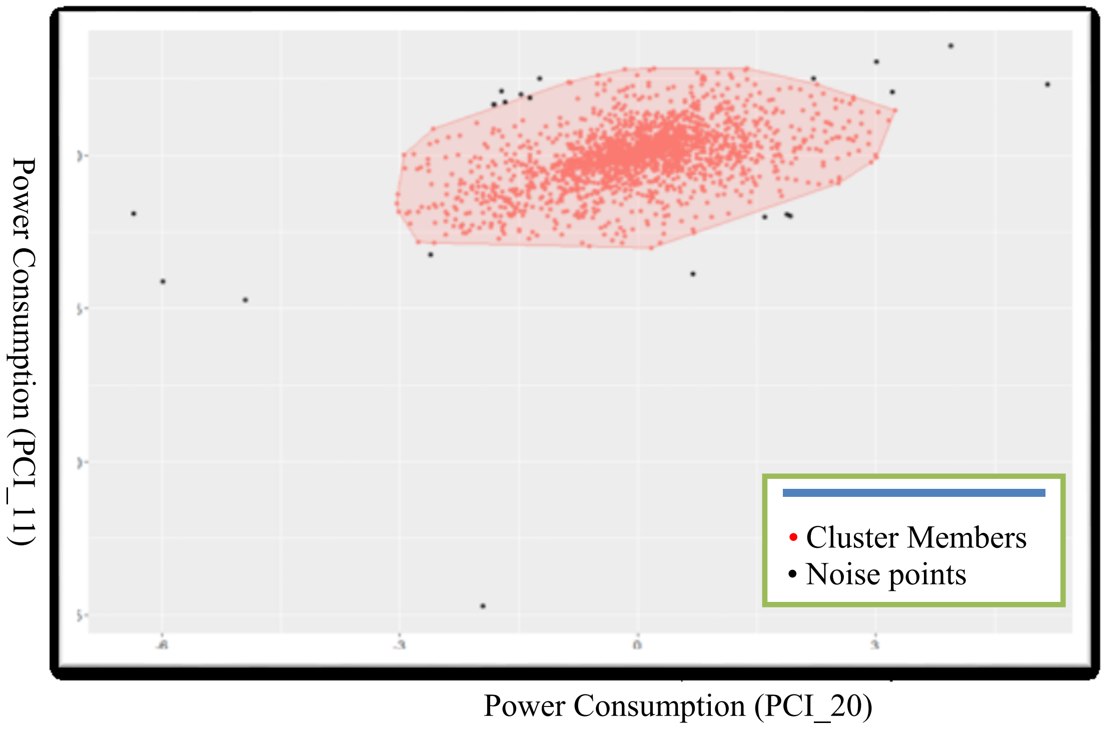


Figure 4.15 Outlier on Cluster II using DBSCAN:2178

SKNN WITH 1% clean Sample

The next step in the algorithm is to select clean samples from the two clean clusters: 1795 and 2157. From each of this clean cluster 1%, 2% and 5% each will be selected and added to the centroids obtained from the axis. The 1% sample will be 0.01* 1795 = 18 and 0.01* 2157= 22 approximately. We have 40 data-points plus the two data-points each from the centroids; in total, we have 42 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.139 secs seconds. A similar result will be calculated for 2% sample and 5% sample of the data set.

Table 4.10 Predicted Cluster with 1% SKNN

Cluster name	Predicted Cluster	Mean	Median	1st Quartile	3rd Quartile
Cluster I	89934	0.4041	0.5158	0.2584	0.5288
Cluster II	106066	0.7705	0.8397	0.6791	0.8500

Table 4.11 Time taken to clean the remaining Sample

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	89934	89621	313	1.873 mins
Cluster 11	106066	105737	329	2.00 mins

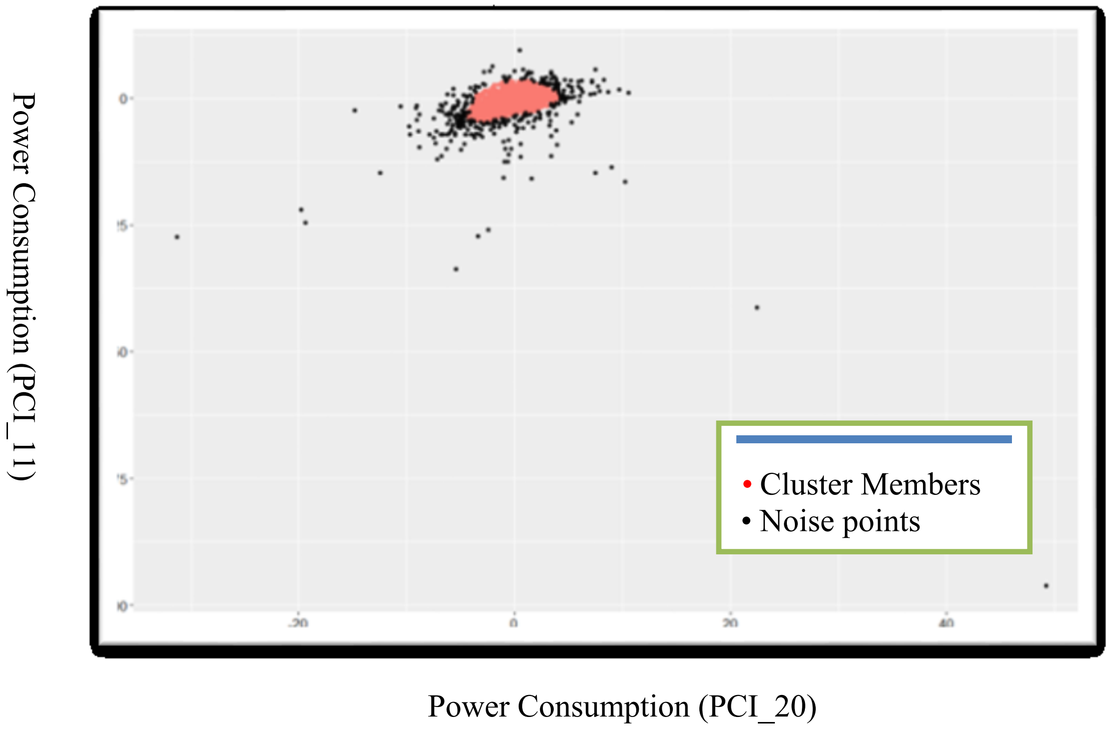


Figure 4.16 Outlier on Cluster I using DBSCAN:89934

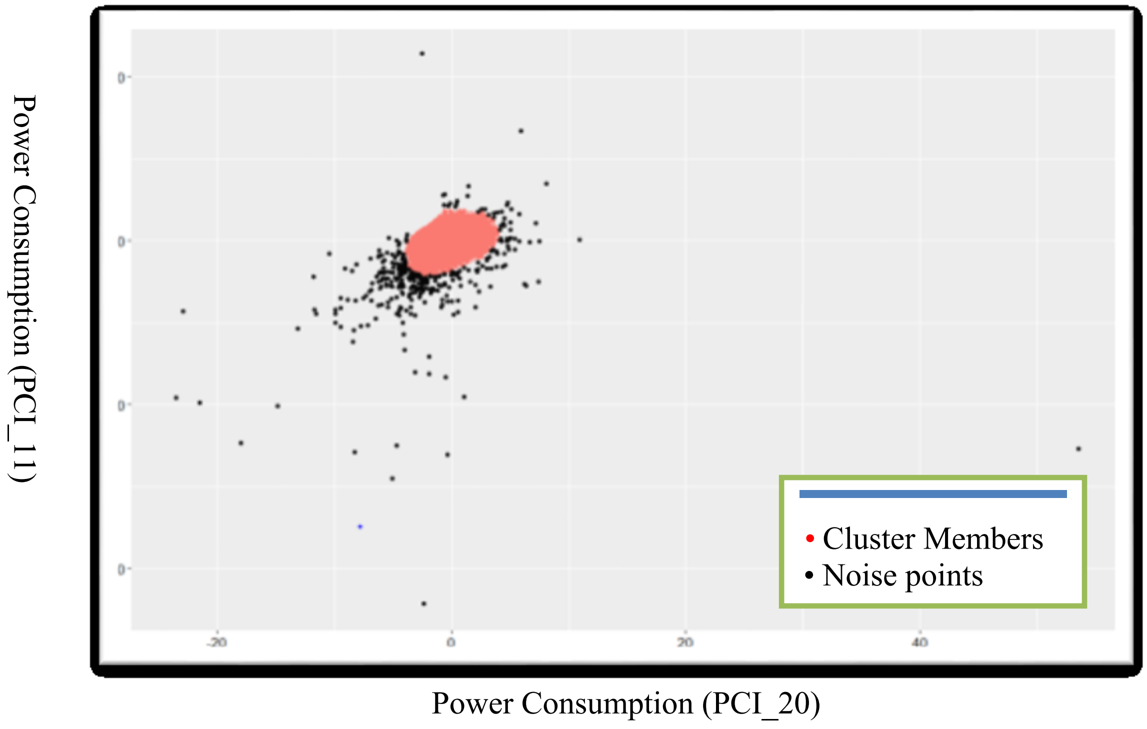


Figure 4.17 Outlier on Cluster II using DBSCAN:106066

SKNN WITH 2% Sample

The 2% sample will be 0.02* 1795 = 36 and 0.02* 2178= 44 approximately, which gives 80 data-points plus the two data-points each from the centroids; in total, we have 82 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.189 secs. Similar results will be calculated for 5% sample.

Table 4. 12 Predicted Clusters on 2% Sample

Cluster name	Predicted Cluster	Mean	Median	1st Quartile	3rd Quartile
Cluster I	89921	0.4041	0.5158	0.2584	0.5288
Cluster II	106079	0.7705	0.8397	0.6791	0.8500

Table 4.13 Time Taken to Clean each Cluster (SKNN 2%)

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	89921	89621	300	1.934 mins
Cluster 11	106079	105737	342	2.00 mins

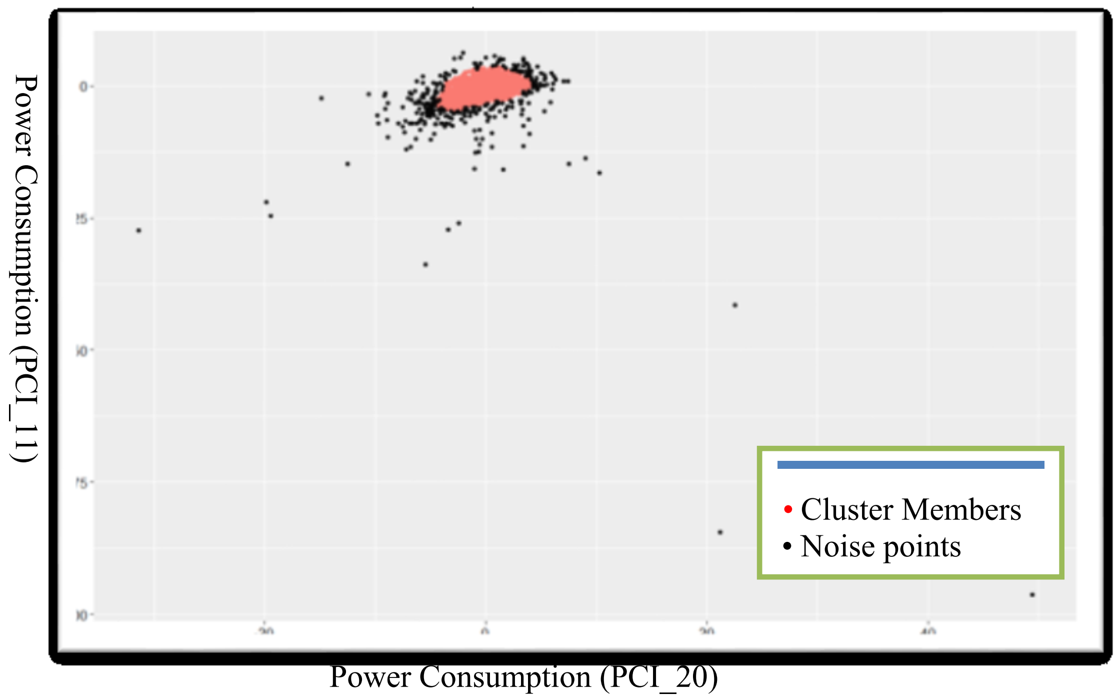


Figure 4. 18 Outlier on Cluster I using DBSCAN:89921

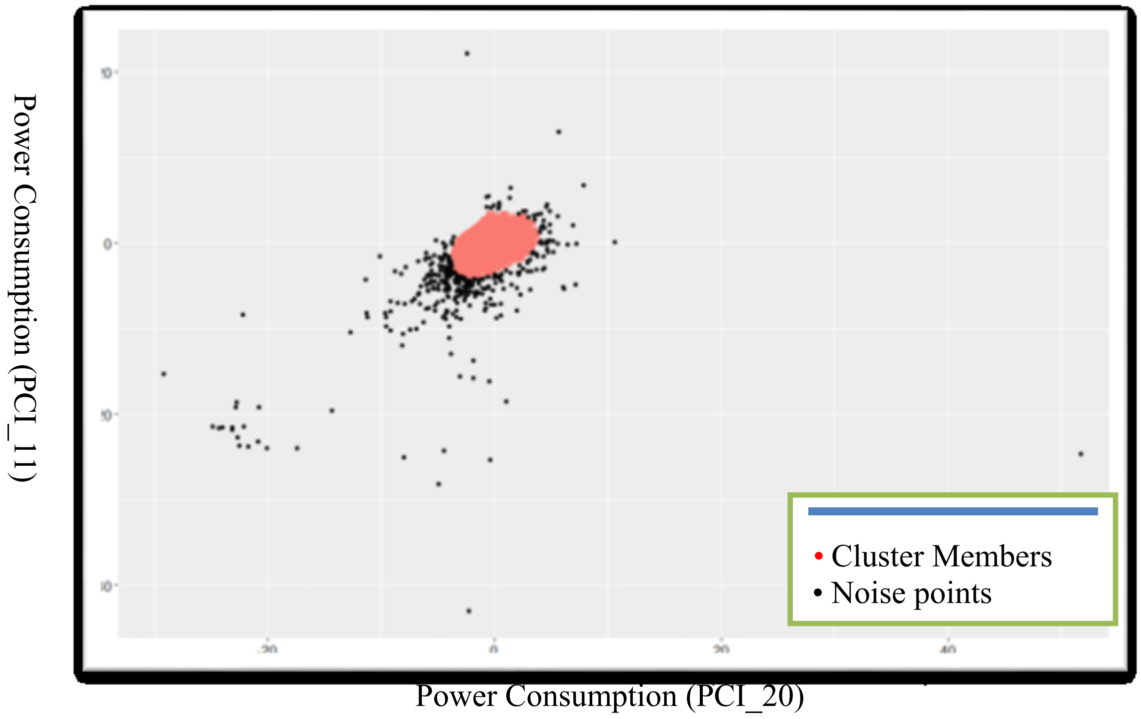


Figure 4.19 Outlier on Cluster II using DBSCAN:106079

SKNN WITH 5% Sample

The 5% sample will be 0.05* **1795** = 90 and 0.05* **2178**= 109 approximately, which is equivalent to 199 data-points plus the two data-points each from the centroids; in total, we have 201 data points as the seed for our training set for K Nearest Neighbor Algorithm. The total time taken for KNN to classify the sample is 0.184 secs.

Table 4.14 Predicted Clusters (SKNN 5%)

Cluster name	**Predicted Cluster**	**Mean**	**Median**	**1st Quartile**	**3rd Quartile**
Cluster I	89888	0.4041	0.5158	0.2584	0.5288
Cluster II	106112	0.7705	0.8396	0.6791	0.8500

Table 4.15 Time taken to Clean the Clusters (SKNN 5%)

Cluster Name	Cluster Size	Clean Cluster	Noise Points	Time Taken(DBSCAN)
Cluster 1	89888	89621	267	1.905 mins
Cluster 11	106112	105737	375	2 mins

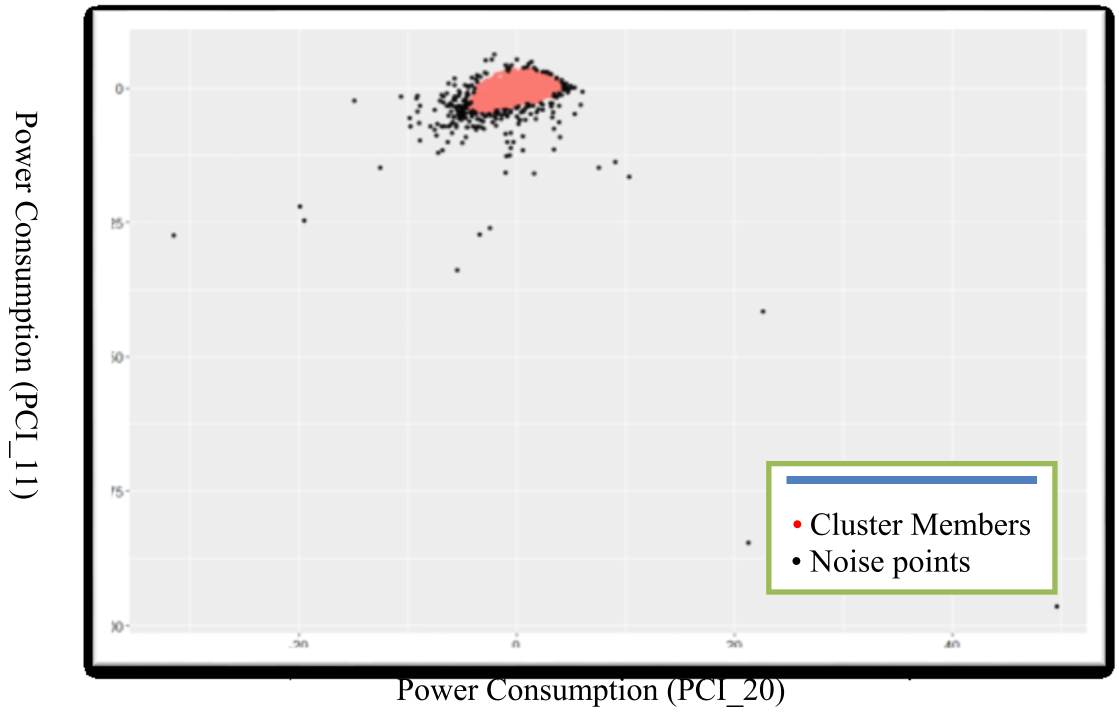


Figure 4. 20 Outlier on Cluster I using DBSCAN:89888

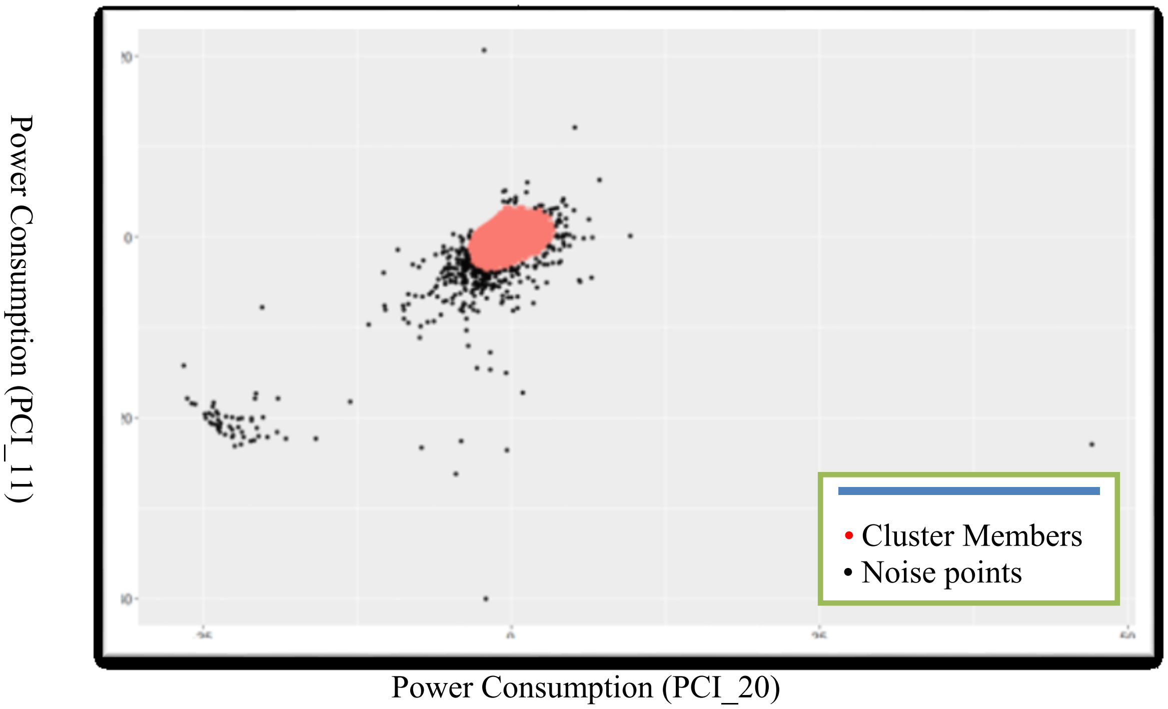


Figure 4. 21 Outlier on Cluster II using DBSCAN:106112

Analysis of the data based on the cluster overlap and the run time obtained for each of the algorithms shows that the time taken by K means to partition the data set into the two clusters identified by Elbow method is 7.56 seconds, while K-means with DBSCAN (KD) took 268.62 seconds. K-means, as expected, is quite faster than a combination of K-means and DBSCAN because both methods were used directly on the entire sample only that KD identified the anomalies. The noise points identified by KD on cluster1 is 305 while the noise point identified for cluster II is 257 with a total of 91439 clean clusters in cluster I and 107999 clean clusters in cluster II. The rest of the result for KD algorithm is as shown in Figure. 4.16

Table 4. 16 Predicted Cluster From KD

S/N	Cluster Size			
		Clean	Noise	Total
1	Cluster I	91439	305	91744
2	Cluster II	107999	257	108256

4.4.3 SUMMARY OF KDS RESULTS

Table 4.17 and 4.18 shows the result of running time and cluster overlap for KDS. This approach uses Small Training Set obtained from clean samples of both clusters, from 1% sample and 2% sample as a training set for K Nearest Neighbor Algorithms. The result shows that the highest time taking is 0.226 seconds and 0.298 seconds for 1% and 5% sample as the training set. While it was 0.363 seconds and 0.513 for 1% and 5% training sample for 2% sample, the algorithm outperforms K-means which took 7.56 seconds. As earlier reported the cluster do not defer much when compared with one another in both methods, there was above 95% similarity in the cluster sample for both cluster I and Cluster II when the cluster was compared with initial clusters from K-means.

Table 4.17 Running Times for KDS Algorithm

S/N	1 Percent Sample/sec (2000)		2 Percent Sample/sec (4000)	
1	K Means	0.089s	K Means	0.159
2	DBSCAN for Cluster I	0.010s	Clean Cluster I	0.028
3	DBSCAN for Cluster II	0.015s	Clean Cluster II	0.037
4	SKNN (1% + Centroids)	0.112s	SKNN (1% + Centroids)	0.139
5	SKNN (2% + Centroids)	0.134s	SKNN (2% + Centroids)	0.189
6	SKNN (5% + Centroids)	0.184s	SKNN (5% + Centroids)	0.286
7	Subtotal (1% + Centroids)	**0.226s**	Subtotal (1% + Centroids)	**0.363**
8	Subtotal (2% + Centroids)	**0.248s**	Subtotal (2% + Centroids)	**0.413**
9	Subtotal (5% + Centroids)	**0.298s**	Subtotal (5% + Centroids)	**0.513**

Table 4. 18 Cluster and noise points for KDS

S/N	I Percent Sample/Cluster Size			2 Percent Sample/ Cluster Size	
		Cluster I	**Cluster 2**	**Cluster 1**	**Cluster 2**
1	SKNN (1% + Centroids)	90783	107217	89934	106066
2	SKNN (2% + Centroids)	90617	**107383**	**89921**	**106079**
3	SKNN (5% + Centroids)	90783	**107217**	**89888**	**106112**

4.4.4 SUMMARY OF KDSD RESULTS

Tables 4.19 and 4.20 shows the outcomes of running time and anomalies analysis for KDSD algorithm. The results are similar to the one obtained by the earlier two methods but due to the cleaning of the generated cluster, there was an increase in the time taken. The result shows for 1% and 2% training sample for initial 2000 observations is 236.526 seconds and 234.548 seconds for 1% and 2% sample as training set and 232.363 seconds and 236.453 seconds for 1% and 2% sample with 4000 observations as initial sample. Although this was not able to perform as fast as K-means algorithm, it outperforms KD which took 268.62 seconds for all sample considered.

Table 4.19 Running Times for KDSD Algorithm

S/N	1 Percent Sample/sec (2000)		2 Percent Sample/sec (4000)	
1	K-means	0.089	K-means	0.159
2	DBSCAN for Cluster I	0.010	Clean Cluster I	0.028
3	DBSCAN for Cluster II	0.015	Clean Cluster II	0.037
4	SKNN (1% + Centroids)	0.112	SKNN (1% + Centroids)	0.139
	Clean Cluster I	115.400	Clean Cluster I	112.38
	Clean Cluster II	120.900	Clean Cluster II	120.000
5	SKNN (2% + Centroids)	0.134	SKNN (2% + Centroids)	0.189
	Clean Cluster I	114.300	Clean Cluster I	116.04
	Clean Cluster II	120.000	Clean Cluster II	120.000
6	SKNN (5% + Centroids)	0.184	SKNN (5% + Centroids)	0.286
	Clean Cluster 1	117.540	Clean Cluster I	114.540
	Clean Cluster II	120.000	Clean Cluster II	120.000
7	Subtotal (1% + Centroids)	**236.526**	Subtotal (1% + Centroids)	**232.363**
8	Subtotal (2% + Centroids)	**234.548**	Subtotal (2% + Centroids)	**236.453**
9	Subtotal (5% + Centroids)	**237.836**	Subtotal (5% + Centroids)	**235.05**

Table 4.20 Cluster and noise points for KDSD

S/N	I Percent Sample/Cluster Size				2 Percent Sample/ Cluster Size		
1	SKNN (1% + Centroids)	Clean	Noise	Total	Clean	Noise	Total
	Cluster I	90491	292	90783	89621	313	89934
	Cluster 1I	106869	348	107217	105737	329	106066
2	SKNN (2% + Centroids)						
	Cluster I	90377	240	90617	89621	300	89921
	Cluster II	106821	562	107383	105737	342	106079
3	SKNN (5% + Centroids)						
	Cluster I	90491	292	90783	89621	267	89888
	Cluster II	106869	348	107217	105737	375	106112

Using the two clusters generated from KM as a baseline, we compared the result with clusters generated from KD, KDS and KDSD element-wise, the table below is a summary of the overlaps from the different clusters. The table shows the overlapping percentage of clusters when KD, KDS, and KDSD were compared with cluster generated from K-Means. The result indicates that clusters generated from K-Means contain anomalies and that each of the other methods generated cleaner cluster as can be inferred from the table.

Table 4.21 Cluster Overlapping Percentage KM, KD, KDS and KDSD

S/N	Cluster 1 Percentage Overlap	Cluster 2 Percentage Overlap
KD	99.20	99.31
KDS	98.25	99.71
KDSD	97.93	99.70

Similarly, the two clusters resulting from KD were used as the baseline to check how the corresponding cluster from KDS and KDSD overlaps i.e the percentage of the cluster members of KD clusters that appear in the corresponding cluster from KDS and KDSD. The result is as shown in the table below 4.22.

Table 4.22 Cluster Overlapping Percentage KD, KDS, and KDSD

S/N	Cluster 1 Percentage Overlap	Cluster 2 Percentage Overlap
KDS	98.95	99.04
KDSD	98.63	98.71

We conclude that although KDSD performs a bit higher in terms of running time, the resulting clusters generated from it are much cleaner than what obtains from either KDS or KD.

CHAPTER 5 CONCLUSION AND FUTURE WORK

This chapter concludes the thesis by summarizing the experimental results and comparing the proposed algorithms to k-means clustering. The improvements that are planned to be completed in the future are also presented.

5.1 CONCLUSION

In this thesis, we propose a series of fast clustering algorithms for large data set with special consideration on time series data set. The proposed methods explore a combination of excellent features of existing clustering and classification algorithms. The proposed algorithms detect anomalies and correctly classify large data set with improved speed.

In order to evaluate the effectiveness of the algorithms, we carried out an extensive analysis of smart meter data, which is a good example of time series data. Extensive data analysis was carried out using Scala, R Studio, and a few Python scripts. The analysis incorporates a careful analysis of data obtained over a period with special emphasis on the weekdays to be able to correctly understand the customers' consumption trend over the periods considered.

In order to achieve the goal, we designed three innovative algorithms, KD, KDS, and KDSD. Our experimental results show that they can quickly cluster large data sets and effectively remove possible anomalies from the data set. KD, KDS, and KDSD use the excellent features of K-means algorithm in handling large data set coupled with DBSCAN ability to cluster data set of arbitrary shape and identify possible outliers. Our implementation incorporates an improved multi-stage sampling method to arrive at a training set that uniquely mirrors the data set and generates a model that gives near accurate classification of the unlabeled data set using the proposed SKNN approach.

In detail, compared to K-means, KDS runs at a much faster rate. Specifically, our experimental results show that it takes K-means 7.56 seconds to cluster the whole data set under investigation. However, it takes KDS 0.363 seconds and 0.513 seconds in the case of 1% and 5% training sample over 2% initial training data respectively. We also found that although KDSD is not as fast as KDS due to the final anomaly removal operation, it outperforms KD, which is simply a combination of K-Means and DBSCAN. In our experiments, it takes KD 268.62 seconds to complete the clustering process while it takes KDSD 237.836 seconds in the worst case.

5.2 FUTURE WORK

Both KDS and KDSD involve SKNN, which is essentially a classification algorithm. Currently, SKNN constructs the training set by randomly selecting a few data points in the cleaned clusters. Despite the fact that this approach leads to low complexity cost, it might result in a skewed training set. In the future, a distance calculation method could be utilized by SKNN to determine the distance between all cluster members and the centroids of the cleaned clusters before it is used to train the model. Then a training set that includes data points of varied distances could be constructed. We expect that this approach will lead to cleaner clusters. This work is only used for time series data set, extending it to data from other domains will also be interesting.

BIBLIOGRAPHY

[1] C. Lin, C. Chen, H. Lee, and J. Liao, “Expert Systems with Applications Fast K-means algorithm based on a level histogram for image retrieval,” *Expert Syst. Appl.*, vol. 41, no. 7, pp. 3276–3283, 2014.

[2] F. Mcloughlin, A. Duffy, and M. Conlon, “A clustering approach to domestic electricity load profile characterization using smart metering data,” vol. 141, pp. 190–199, 2015.

[3] Y. Lu, T. Zhang, and Z. Zeng, “Adaptive weighted fuzzy clustering algorithm for load profiling of smart grid customers,” *2016 IEEE/CIC Int. Conf. Commun. China*, pp. 1–6, 2016.

[4] J. Wang, R. Cardell-Oliver, and W. Liu, “Knowledge-Base d Systems An incremental algorithm for discovering routine behaviors from smart meter data,” vol. 113, pp. 61–74, 2016.

[5] F. Mcloughlin, A. Duffy, and M. Conlon, “Evaluation of time series techniques to characterize domestic electricity demand,” *Energy*, vol. 50, pp. 120–130, 2013.

[6] J. Schleich, C. Faure, and M. Klobasa, “Persistence of the effects of providing feedback alongside smart metering devices on household electricity demand,” *Energy Policy*, vol. 107, no. April, pp. 225–233, 2017.

[7] P. Winters, B. Data, and S. Energy, “Big Data, Smart Energy, and Predictive Analytics Time Series Prediction of Smart Energy Data,” *KNIME Whitepaper Big Data, Smart Energy, Predict. Anal.*, pp. 1–37, 2013.

[8] K. Le Zhou, S. L. Yang, and C. Shen, “A review of electric load classification in a smart grid environment,” *Renew. Sustain. Energy Rev.*, vol. 24, pp. 103–110, 2013.

[9] V. Le and S. Kim, “K-strings algorithm, a new approach based on Kmeans,” no. 2, pp. 15–20.

[10] X. Zhang, J. Liu, Y. Du, and T. Lv, “Expert Systems with Applications A novel clustering method on time series data,” vol. 38, pp. 11891–11900, 2011.

[11] I. P. Panapakidis, M. C. Alexiadis, and G. K. Papagiannis, “Application of competitive learning clustering in the load time series segmentation,” *Proc. Univ. Power Eng. Conf.*, 2013.

[12] A. Al-wakeel and J. Wu, “K-means based cluster analysis of residential smart meter measurements,” vol. 88, pp. 754–760, 2016.

[13] A. Al-wakeel, J. Wu, and N. Jenkins, “k -means based load estimation of domestic smart meter measurements q,” vol. 194, pp. 333–342, 2017.

[14] P. J. Flynn, “Data Clustering : A Review,” vol. 31, no. 3, 2000.

[15] Merriam-Webster, “Cluster Analysis.” [Online]. Available: https://www.merriam-webster.com/dictionary/cluster+analysis. [Accessed: 30-Jun-2017].

[16] A. D. E. R. Aftery, “How Many Clusters ? Which Clustering Method ? Answers Via Model-Based Cluster Analysis,” vol. 41, no. 8, 1998.

[17] H. S. H. Sun, Z. L. Z. Liu, and L. K. L. Kong, “A Document Clustering Method Based on Hierarchical Algorithm with Model Clustering,” *22nd Int. Conf. Adv. Inf. Netw. Appl. - Work. (Aina Work. 2008)*, pp. 1229–1233, 2008.

[18] H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh, and W. L. Nowinski, “Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm,” *2006 IEEE Southwest Symp. Image Anal. Interpret.*, pp. 61–65, 2006.

[19] T. C. Fu, “A review on time series data mining,” *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, 2011.

[20] A. K. Jain, “Data clustering: 50 years beyond K-means,” *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[21] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. 2012.

[22] L. Rokach and O. Maimon, “Chapter 15— Clustering methods,” *Data Min. Knowl. Discov. Handb.*, p. 32, 2010.

[23] M. Martinez-Pabon, T. Eveleigh, and B. Tanju, “Smart meter data analytics for optimal customer selection in demand response programs,” vol. 107, no. September 2016, pp. 49–59, 2017.

[24] W. Qiu, F. Zhai, Z. Bao, B. Li, Q. Yang, and Y. Cao, “Clustering Approach and Characteristic Indices for Load Profiles of Customers Using Data from AMI □,” no. Ciced, pp. 10–13, 2016.

[25] A. Albert and R. Rajagopal, “Smart Meter Driven Segmentation: What Your Consumption Says About You,” vol. 28, no. 4, pp. 4019–4030, 2013.

[26] Strategy N. M. G, “Advanced metering infrastructure,” *US Dep. Energy Off. Electr. Energy Reliab.*, no. February 2008.

[27] R. Rashed Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, “A survey on Advanced Metering Infrastructure,” *Int. J. Electr. Power Energy Syst.*, vol. 63, pp. 473–484, 2014.

[28] S. Meters, A. C. Study, I. Vassileva, and J. Campillo, “Consumers ’ Perspective on Full-Scale Adoption of,” 2016.

[29] F. L. Quilumba, W. J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, “Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities,” *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 911–918, 2015.

[30] T. K. Wijaya, K. Aberer, and D. P. Seetharam, “Consumer Segmentation and Knowledge Extraction from Smart Meter and Survey Data ∗,” pp. 226–234.

[31] “Smart Grid Legislative and Regulatory Policies and Case Studies,” no. December 2011.

[32] Wikipedia, “Smart Meters.” [Online]. Available: https://en.wikipedia.org/wiki/Smart_meter. [Accessed: 30-Jun-2017].

[33] U. Greveler, B. Justus, and D. Loehr, “Multimedia Content Identification Through Smart Meter Power Usage Profiles.”

[34] G. Chicco, R. Napoli, and F. Piglione, “Comparisons among clustering techniques for electricity customer classification,” *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 933–940, 2006.

[35] R. Granell, C. J. Axon, and D. C. H. Wallom, “Impacts of Raw Data Temporal Resolution Using Selected Clustering Methods on Residential Electricity Load Profiles,” *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3217–3224, 2014.

[36] R. Pereira *et al.*, “Electrical Power and Energy Systems A fuzzy clustering approach to a demand response model,” vol. 81, pp. 184–192, 2016.

[37] J. Han and M. A. Piette, “Solutions for Summer Electric Power Shortages : Demand Response and its Applications in Air Conditioning and Refrigerating Systems,” vol. 29, no. 1, pp. 1–4, 2008.

[38] V. Ford, “Clustering of Smart Meter Data for Disaggregation,” pp. 507–510, 2013.

[39] A. M. Learning and L. B. Key, “Securing Metering Infrastructure of Smart Grid :,” 2016.

[40] G. Editor, *Identification of Outliers*.

[41] F. Angiulli and F. Fassetti, “Very efficient mining of distance-based outliers,” *Proc. Sixt. ACM Conf. Conf. Inf. Knowl. Manag. - CIKM ’07*, p. 791, 2007.

[42] E. M. Knox and R. T. Ng, “Algorithms for Mining Datasets Outliers in Large.”

[43] O. Analysis, *Outlier Analysis*.

[44] D.-H. Bae, S. Jeong, S.-W. Kim, and M. Lee, “Outlier detection using centrality and center-proximity,” *Proc. 21st ACM Int. Conf. Inf. Knowl. Manag. - CIKM ’12*, p. 2251, 2012.

[45] V. Hautam, “Improving K-Means by Outlier Removal,” pp. 978–987, 2005.

[46] M. Ester, H. Kriegel, X. Xu, and D.- Miinchen, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” 1996.

[47] C. Procopiuc and J. S. Park, “Fast Algorithms for Projected Clustering,” pp. 61–72, 1999.

[48] R. E. F. E. Rences and N. O. T. Es, “Clustering by fast search and find of,” vol. 344, no. 6191, 2014.

[49] J. Zheng, “Research on Optimization of Clustering by Fast Search and Find of Density Peaks *,” no. 61173130, pp. 129–133, 2016.

[50] T. F. Ghanem, W. S. Elkilani, H. M. Abdelkader, and M. M. Hadhoud, “Fast Dimension-based Partitioning and Merging clustering algorithm,” *Appl. Soft Comput. J.*, vol. 36, pp. 143–151, 2015.

[51] L. Gunisetti, “Outlier detection and visualization of large data sets,” *Proc. Int. Conf. Work. Emerg. Trends Technol. - ICWET ’11*, p. 522, 2011.

[52] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases,” *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.

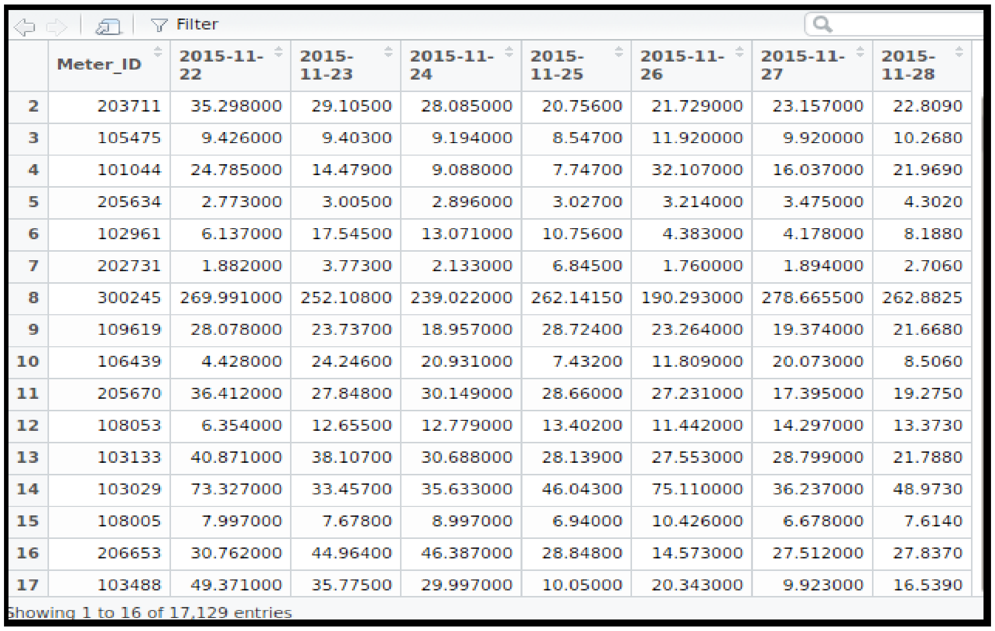
[53] T. Warren Liao, “A clustering procedure for exploratory mining of vector time series,” *Pattern Recognit.*, vol. 40, no. 9, pp. 2550–2562, 2007.

[54] S. Vijayalaksmi, “A Fast Approach to Clustering Datasets using DBSCAN and Pruning Algorithms,” vol. 60, no. 14, pp. 1–7, 2012.

[55] Elbow method (clustering), *Wikipedia*. 29-May-2016.

APPENDIX: SAMPLE CONSUMPTION DATA

The Figures below include the power consumption of a few sample smart meters in November 2015 and December 2015.

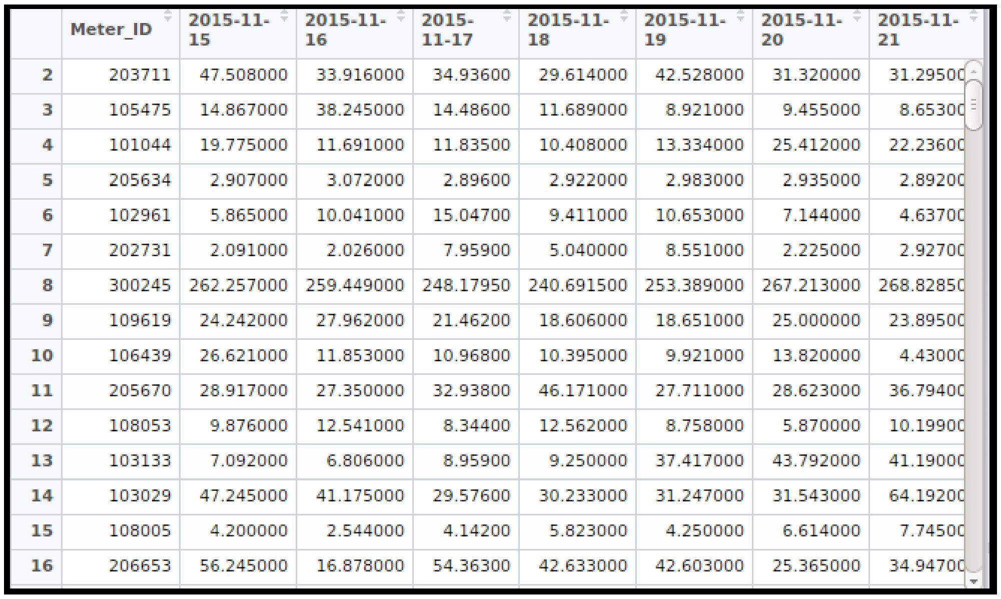


Filter

	Meter_ID	2015-11-22	2015-11-23	2015-11-24	2015-11-25	2015-11-26	2015-11-27	2015-11-28
2	203711	35.298000	29.10500	28.085000	20.75600	21.729000	23.157000	22.8090
3	105475	9.426000	9.40300	9.194000	8.54700	11.920000	9.920000	10.2680
4	101044	24.785000	14.47900	9.088000	7.74700	32.107000	16.037000	21.9690
5	205634	2.773000	3.00500	2.896000	3.02700	3.214000	3.475000	4.3020
6	102961	6.137000	17.54500	13.071000	10.75600	4.383000	4.178000	8.1880
7	202731	1.882000	3.77300	2.133000	6.84500	1.760000	1.894000	2.7060
8	300245	269.991000	252.10800	239.022000	262.14150	190.293000	278.665500	262.8825
9	109619	28.078000	23.73700	18.957000	28.72400	23.264000	19.374000	21.6680
10	106439	4.428000	24.24600	20.931000	7.43200	11.809000	20.073000	8.5060
11	205670	36.412000	27.84800	30.149000	28.66000	27.231000	17.395000	19.2750
12	108053	6.354000	12.65500	12.779000	13.40200	11.442000	14.297000	13.3730
13	103133	40.871000	38.10700	30.688000	28.13900	27.553000	28.799000	21.7880
14	103029	73.327000	33.45700	35.633000	46.04300	75.110000	36.237000	48.9730
15	108005	7.997000	7.67800	8.997000	6.94000	10.426000	6.678000	7.6140
16	206653	30.762000	44.96400	46.387000	28.84800	14.573000	27.512000	27.8370
17	103488	49.371000	35.77500	29.997000	10.05000	20.343000	9.923000	16.5390

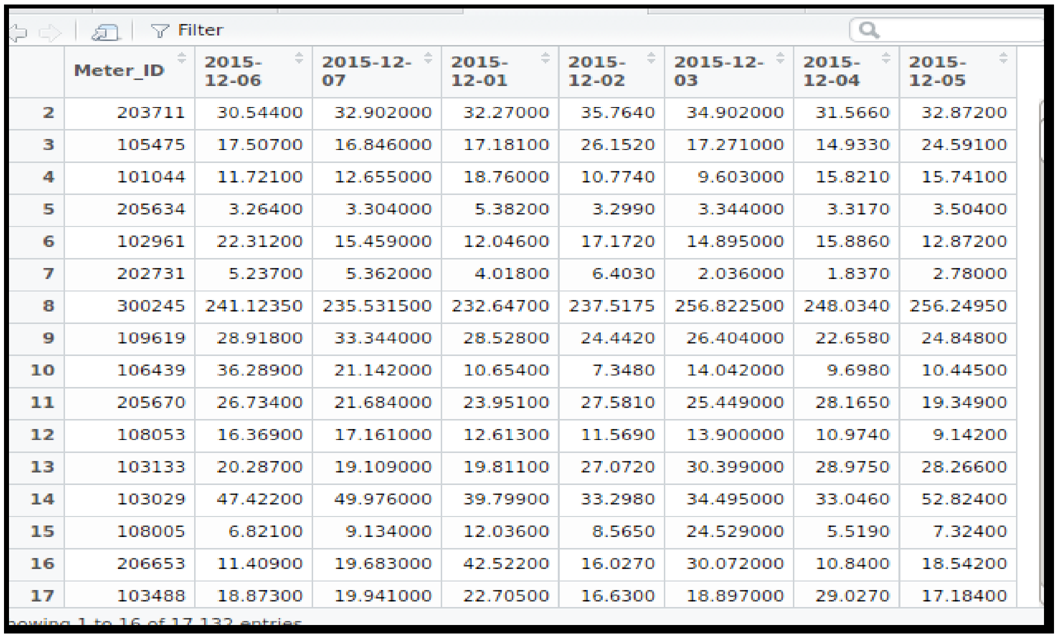
Showing 1 to 16 of 17,129 entries

Figure 1 Consumption variation 15th Nov – 21th Nov.



	Meter_ID	2015-11-15	2015-11-16	2015-11-17	2015-11-18	2015-11-19	2015-11-20	2015-11-21
2	203711	47.508000	33.916000	34.93600	29.614000	42.528000	31.320000	31.29500
3	105475	14.867000	38.245000	14.48600	11.689000	8.921000	9.455000	8.65300
4	101044	19.775000	11.691000	11.83500	10.408000	13.334000	25.412000	22.23600
5	205634	2.907000	3.072000	2.89600	2.922000	2.983000	2.935000	2.89200
6	102961	5.865000	10.041000	15.04700	9.411000	10.653000	7.144000	4.63700
7	202731	2.091000	2.026000	7.95900	5.040000	8.551000	2.225000	2.92700
8	300245	262.257000	259.449000	248.17950	240.691500	253.389000	267.213000	268.82850
9	109619	24.242000	27.962000	21.46200	18.606000	18.651000	25.000000	23.89500
10	106439	26.621000	11.853000	10.96800	10.395000	9.921000	13.820000	4.43000
11	205670	28.917000	27.350000	32.93800	46.171000	27.711000	28.623000	36.79400
12	108053	9.876000	12.541000	8.34400	12.562000	8.758000	5.870000	10.19900
13	103133	7.092000	6.806000	8.95900	9.250000	37.417000	43.792000	41.19000
14	103029	47.245000	41.175000	29.57600	30.233000	31.247000	31.543000	64.19200
15	108005	4.200000	2.544000	4.14200	5.823000	4.250000	6.614000	7.74500
16	206653	56.245000	16.878000	54.36300	42.633000	42.603000	25.365000	34.94700

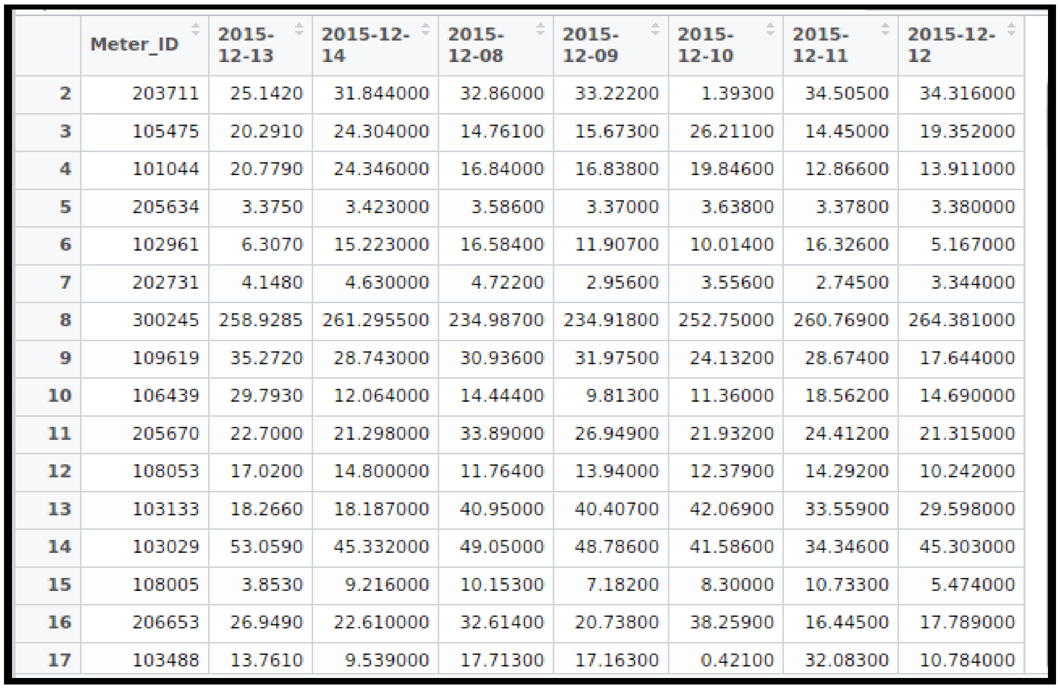
Figure 2 Consumption variation 22nd Nov – 28th Nov.


Filter

	Meter_ID	2015-12-06	2015-12-07	2015-12-01	2015-12-02	2015-12-03	2015-12-04	2015-12-05
2	203711	30.54400	32.902000	32.27000	35.7640	34.902000	31.5660	32.87200
3	105475	17.50700	16.846000	17.18100	26.1520	17.271000	14.9330	24.59100
4	101044	11.72100	12.655000	18.76000	10.7740	9.603000	15.8210	15.74100
5	205634	3.26400	3.304000	5.38200	3.2990	3.344000	3.3170	3.50400
6	102961	22.31200	15.459000	12.04600	17.1720	14.895000	15.8860	12.87200
7	202731	5.23700	5.362000	4.01800	6.4030	2.036000	1.8370	2.78000
8	300245	241.12350	235.531500	232.64700	237.5175	256.822500	248.0340	256.24950
9	109619	28.91800	33.344000	28.52800	24.4420	26.404000	22.6580	24.84800
10	106439	36.28900	21.142000	10.65400	7.3480	14.042000	9.6980	10.44500
11	205670	26.73400	21.684000	23.95100	27.5810	25.449000	28.1650	19.34900
12	108053	16.36900	17.161000	12.61300	11.5690	13.900000	10.9740	9.14200
13	103133	20.28700	19.109000	19.81100	27.0720	30.399000	28.9750	28.26600
14	103029	47.42200	49.976000	39.79900	33.2980	34.495000	33.0460	52.82400
15	108005	6.82100	9.134000	12.03600	8.5650	24.529000	5.5190	7.32400
16	206653	11.40900	19.683000	42.52200	16.0270	30.072000	10.8400	18.54200
17	103488	18.87300	19.941000	22.70500	16.6300	18.897000	29.0270	17.18400

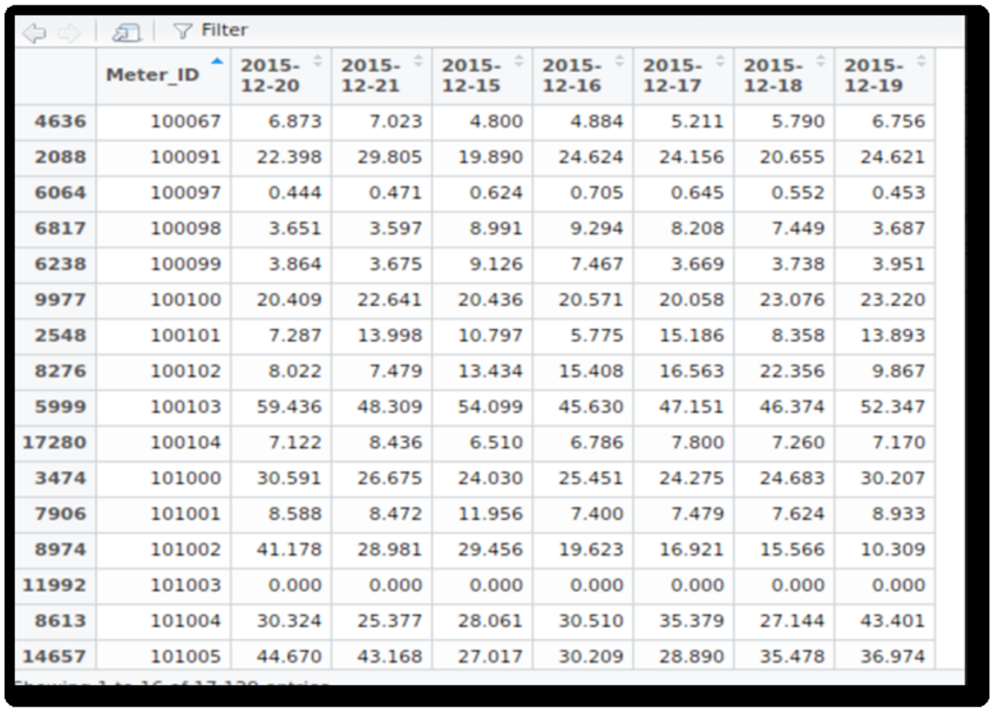
howing 1 to 16 of 17,132 entries

Figure 3 Consumption variation 01th Dec – 7th Dec.



	Meter_ID	2015-12-13	2015-12-14	2015-12-08	2015-12-09	2015-12-10	2015-12-11	2015-12-12
2	203711	25.1420	31.844000	32.86000	33.22200	1.39300	34.50500	34.316000
3	105475	20.2910	24.304000	14.76100	15.67300	26.21100	14.45000	19.352000
4	101044	20.7790	24.346000	16.84000	16.83800	19.84600	12.86600	13.911000
5	205634	3.3750	3.423000	3.58600	3.37000	3.63800	3.37800	3.380000
6	102961	6.3070	15.223000	16.58400	11.90700	10.01400	16.32600	5.167000
7	202731	4.1480	4.630000	4.72200	2.95600	3.55600	2.74500	3.344000
8	300245	258.9285	261.295500	234.98700	234.91800	252.75000	260.76900	264.381000
9	109619	35.2720	28.743000	30.93600	31.97500	24.13200	28.67400	17.644000
10	106439	29.7930	12.064000	14.44400	9.81300	11.36000	18.56200	14.690000
11	205670	22.7000	21.298000	33.89000	26.94900	21.93200	24.41200	21.315000
12	108053	17.0200	14.800000	11.76400	13.94000	12.37900	14.29200	10.242000
13	103133	18.2660	18.187000	40.95000	40.40700	42.06900	33.55900	29.598000
14	103029	53.0590	45.332000	49.05000	48.78600	41.58600	34.34600	45.303000
15	108005	3.8530	9.216000	10.15300	7.18200	8.30000	10.73300	5.474000
16	206653	26.9490	22.610000	32.61400	20.73800	38.25900	16.44500	17.789000
17	103488	13.7610	9.539000	17.71300	17.16300	0.42100	32.08300	10.784000

Figure 4Consumption variation 8th Dec – 14th Dec.



Filter

	Meter_ID	2015-12-20	2015-12-21	2015-12-15	2015-12-16	2015-12-17	2015-12-18	2015-12-19
4636	100067	6.873	7.023	4.800	4.884	5.211	5.790	6.756
2088	100091	22.398	29.805	19.890	24.624	24.156	20.655	24.621
6064	100097	0.444	0.471	0.624	0.705	0.645	0.552	0.453
6817	100098	3.651	3.597	8.991	9.294	8.208	7.449	3.687
6238	100099	3.864	3.675	9.126	7.467	3.669	3.738	3.951
9977	100100	20.409	22.641	20.436	20.571	20.058	23.076	23.220
2548	100101	7.287	13.998	10.797	5.775	15.186	8.358	13.893
8276	100102	8.022	7.479	13.434	15.408	16.563	22.356	9.867
5999	100103	59.436	48.309	54.099	45.630	47.151	46.374	52.347
17280	100104	7.122	8.436	6.510	6.786	7.800	7.260	7.170
3474	101000	30.591	26.675	24.030	25.451	24.275	24.683	30.207
7906	101001	8.588	8.472	11.956	7.400	7.479	7.624	8.933
8974	101002	41.178	28.981	29.456	19.623	16.921	15.566	10.309
11992	101003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8613	101004	30.324	25.377	28.061	30.510	35.379	27.144	43.401
14657	101005	44.670	43.168	27.017	30.209	28.890	35.478	36.974

Figure 5 Consumption variation 15th Dec – 21st Dec.