# Investigation of Mobile Network Traffic Using Hadoop and Mahout Machine Learning Methods

by

Man Si

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

## Abstract

Since the emergence of mobile networks, the number of mobile subscriptions has continued to increase year after year. To efficiently assign mobile network resources such as spectrum (which is rare and expensive), the network operator needs to process and analyze information and statistics about each base station and the traffic that passes through it.

This thesis focuses on processing and analyzing two datasets provided by our industrial partner, Ericsson, Canada. A detailed approach that uses Apache Hadoop and the Mahout machine learning library to process and analyze the datasets is presented. The analysis provides insights to the network operator about the resource usage of network devices. This information is of great importance to network operators for efficient and effective management of resources and user experience. Furthermore, an investigation has been conducted that evaluates the impact of executing the Mahout clustering algorithms with various system and workload parameters on a Hadoop cluster.

## Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Dr. Chung-Horng Lung and Dr. Samuel A. Ajila, for their instructive advice and valuable guidance throughout my studies. I am deeply impressed by their professionalism, patience, and strong work ethic. I would like to also thank Mr. Wayne Ding from Ericsson, Canada for supplying the experimental datasets and providing feedback and expertise throughout the project. In addition, this research project is partly sponsored by NSERC through an Engage research grant.

I would like to thank Norman Lim and Duo Liu, who as experts in this research area, were always willing to help and give their best suggestions. Many thanks to Chengchao, Xiaolin, Zhiyuan, Fikirte, Vanessa, Mana, and my other friends. My research life would have been lonely without their company and friendship.

Finally, I would like to thank my parents and relatives for their unconditional support and continuous encouragement. This thesis would not have been finished without them.

# Table of Contents

## List of Tables

## List of Figures

## List of Abbreviations

| | |
|---|---|
| AMD | Advanced Micro Devices |
| CPU | Central Processing Unit |
| DL | Downlink |
| DRB | Data Radio Bearer |
| eNodeB | Evolved Node B |
| GB | Gigabyte |
| GFS | Google File System |
| HD | High Definition |
| HDFS | Hadoop Distributed File System |
| LAN | Local Area Network |
| LTE | Long-Term Evolution |
| PC | Principle Components |
| PCA | Principle Component Analysis |
| PDCP | Packet Data Convergence Protocol |
| QoS | Quality of Service |
| RRC | Radio Resource Control |
| SE | Schedule Entity |
| SRB | Signal Radio Bearer |
| SVD | Singular Value Decomposition |
| TB | Terabyte |
| TTI | Transmission Time Interval |
| UE | User Equipment |
| UL | Uplink |
| VM | Virtual Machine |

# Chapter 1: Introduction

With the emergence of mobile networks, more and more users are connected to the network and have access to the Internet through their mobile devices. Each connected user occupies the limited radio resources of mobile network (e.g. spectrum) for a certain period of time. Since the radio resources are rare and expensive, mobile network operators spend a great deal of time and money to determine how to efficiently allocate these resources. More specifically, the mobile network operators aim to maximize profits and ensuring that the users are satisfied with the quality of service (QoS). To support resource allocation, the mobile network operators collect the statistics of the traffic (e.g. throughput, average number of connections/cell) from base stations. This information can be analyzed to determine how the resources are used and to find traffic patterns from the data. However, the base stations can collect a wide variety of information for a long period and hence the amount of data collected becomes vast. Analyzing such a large amount of data becomes a challenge and this problem can be solved efficiently by using methods in Big Data Analytics.

Hadoop [2] and its machine learning library called Mahout [3] are well-known techniques used for analyzing Big Data. This thesis focuses on exploring and comparing the popular clustering algorithms of Mahout to analyze real base station data provided by our industrial partner, Ericsson, Canada. In addition, Hadoop parameters, e.g., the number of Map tasks, the number of Reduce tasks are tuned to achieve the optimal performance of running K-means clustering on a Hadoop cluster. The experiments are performed on a Hadoop cluster composed of Virtual Machines within a server.

## 1.1 Motivation

There are two main motivating factors for performing this research, which are described in the following sub-sections 1.1.1 and 1.1.2.

### *1.1.1 Limited Resources*

With the popularity of mobile devices, more and more users are connected to the mobile network. For example, a report [1] has stated that total number of mobile subscriptions in the first quarter of 2015 was approximately 7.2 billion, which included 108 million new subscriptions. It is estimated that mobile subscriptions, globally, are growing by 1.5 percent each quarter, and approximately 5 percent every year. The graph displayed in Figure 1.1 depicts how the number of different subscribers and subscriptions changed from 2010 to 2014 and how these numbers are expected to increase in the future years to come. As can be observed from the graph, mobile broadband subscriptions are expected to grow rapidly in the next few years. Conversely, it is observed that the number of mobile subscribers is not expected to increase substantially compared to the number of mobile subscriptions. There will not be as many new mobile subscribers, but the number of mobile subscriptions will increase because more and more users are subscribing secondary devices, such as tablets, to the mobile network.

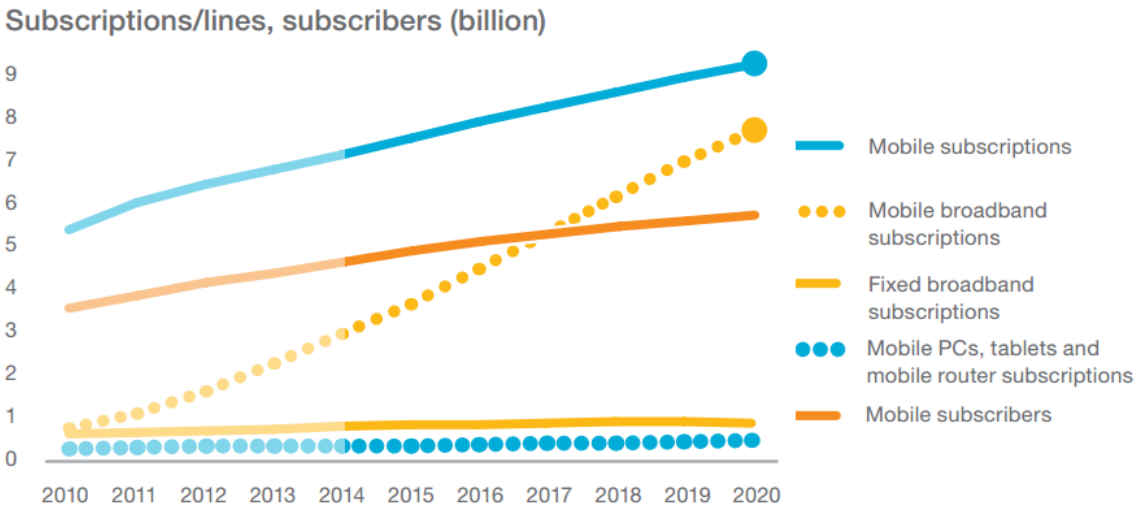


**Figure 1.1:** Number of Subscribers and Subscriptions by 2020 [1]

The continuous growth of mobile subscriptions brings a big challenge for mobile network operators to provide a robust and scalable mobile network. In addition, the demand for high QoS is continuing to grow. However, a mobile network operator has limited wireless resources, including spectrum, and these wireless resources are rare and expensive. Devising efficient and effective techniques to allocate these limited wireless resources while also maximizing the user experience has become the focus of attention for a great deal of research.

System data collected by base stations consists of valuable information that can reveal the current state and quality of the system. With the emergence of LTE (Long-Term Evolution) networks and the increasing demand for QoS, monitoring the network system and analyzing system data has become an important way for mobile network operators to oversee the health, quality, and stability of the mobile networks.

### *1.1.2 Large Volume of Data*

The base stations in a mobile network are always collecting and storing a wide variety of information about the mobile network, including the average number of connected users and the uplink and downlink throughputs. In addition, a mobile network can have hundreds or thousands of base stations, and thus, the amount of data that needs to be analyzed to find meaningful information and useful patterns can be substantial. Understanding the relationship between the different statistics and the effect that the statistics can have on the quality of mobile networks is essential. However, one of the main difficulties with analyzing the system data collected by the base stations is the huge volume. The traditional data analysis techniques that use a single machine cannot efficiently process the large volume of data.

## 1.2 Contributions

The main contributions of this thesis, which are performed in collaboration with our industrial partner, Ericsson, Canada are summarized below:

- Presenting an approach to processing and analyzing mobile network data for more than 5000 base stations. The approach consists of multiple steps, including identification of the relationship between features, adoption of Principle Component Analysis (PCA) [4] for feature reduction, and application of clustering algorithms for categorizing data records.
- Applying Mahout k-means and fuzzy k-means algorithms on the experimental datasets to group together data records that have similar attributes. Various experiments have been conducted to find the values to use for the required parameters (e.g., *K* value that specifies the number of clusters to sort the data into)

of the k-means and fuzzy k-means algorithms that generate the highest quality clustering results. Furthermore, experiments have been performed using PCA to improve execution time as a result of feature reduction, while still maintaining an accurate clustering solution.

- Detailed investigation of a critical feature to mobile network operators: the Average RRC (Radio Resource Control) Connections/Cell that specifies the average number of user connected to the base station. The analysis reveals how the Average RRC Connections/Cell changes throughout a 24-hour period and how the base station cells are affected.
- Investigating how the performance of a Hadoop cluster is affected when executing the k-means clustering algorithm with different system and workload parameters, including dataset sizes, the number of Map tasks, the number of Reduce tasks, and the number of processing cores.

## 1.3 Dissertation Outline

The rest of the thesis is organized as follows:

Chapter 2 provides background information on the clustering algorithms and the Apache Hadoop platform. In addition, related works that use Hadoop and clustering techniques to analyze data are reviewed.

Chapter 3 describes the proposed approach and the steps to analyze the historical base station data provided by our industrial partners, Ericsson, Canada.

Chapter 4 presents the experimental results and a performance analysis of executing the Mahout clustering algorithms on a Hadoop cluster using a variety of different system and workload parameters.

Lastly, Chapter 5 concludes the thesis and discusses possible directions for future research.

# Chapter 2: Background and Related Works

This chapter describes the background information and related works in the area of Big Data analytics using clustering methods in Apache Mahout, which is a machine learning library built on top of the Apache Hadoop platform. The chapter starts with an introduction to Apache Hadoop which is the fundamental platform to enable scalable processing of large datasets. Following that, Section 2.2 presents basic concepts of clustering algorithms and introduces the clustering solutions provided by Mahout. Other research related to mobile traffic data analysis and research using methods in Apache Mahout is reviewed in Section 2.3.

## 2.1 Introduction to Apache Hadoop

Apache Hadoop [2] is a software platform used for analyzing and processing large datasets. It is an open-source software framework implemented using Java and allows for the distributed processing of large datasets using a cluster of computers. Hadoop was formally introduced by the Apache Software Foundation in the fall of 2005 as part of the Nutch subproject of Lucene. Hadoop was started by Doug Cutting who also created Apache Lucene—a widely used Text Search library. Nutch is an open-source Web crawler and web search system [5]. Hadoop is inspired by Google File System (GFS) [6] and the MapReduce programming model [7], both were introduced by Google in 2003 and 2004, respectively. Hadoop consists of two core components: the Hadoop Distributed File System (HDFS) and the MapReduce Software Framework. These components are discussed in more detail in the following sections 2.1.1 - 2.1.2.

### *2.1.1 Hadoop Distributed File System (HDFS)*

HDFS is a distributed file system that runs on a cluster of commodity hardware and is designed to store very large files that have a streaming data access pattern [8]. Streaming implies that it can offer a constant bitrate above a certain threshold when transferring the data, in order to avoid having the data come in bursts or waves. HDFS is similar to existing distributed file systems; however, there are some key advantages for HDFS. First of all, HDFS is designed to have a very high fault-tolerant characteristic, where each data file is replicated three times among the cluster of computers. Secondly, HDFS is designed to be deployed on low-cost commodity hardware. Lastly, HDFS provides high throughput, so that a large dataset of an application can be accessed in a short amount of time.

Files stored in HDFS are split into blocks of which the default size is 64MB. These chunks of data are spread among the computer nodes in the cluster and will be read in parallel when the data needs to be accessed. Since the dataset can be very large, transferring it between cluster nodes is not efficient and can cause high network traffic. In order to solve this problem, Hadoop is designed to transfer the application code (e.g. Hadoop program code) to the cluster nodes instead of transferring the dataset.

The HDFS is managed by two main processes which are called Hadoop daemons: NameNode and DataNode. NameNode works as a master and manages the file system namespace. NameNode stores the metadata for the data files, which are the locations of where each data block of a file is stored. In Hadoop, the data blocks are stored on the local storage of the machines that DataNodes run on. DataNodes act as slave nodes in a HDFS file system. DataNodes hold the actual data blocks and each block is replicated three times

on the cluster by default. The procedure of reading and writing files in HDFS is shown in Figure 2.1.

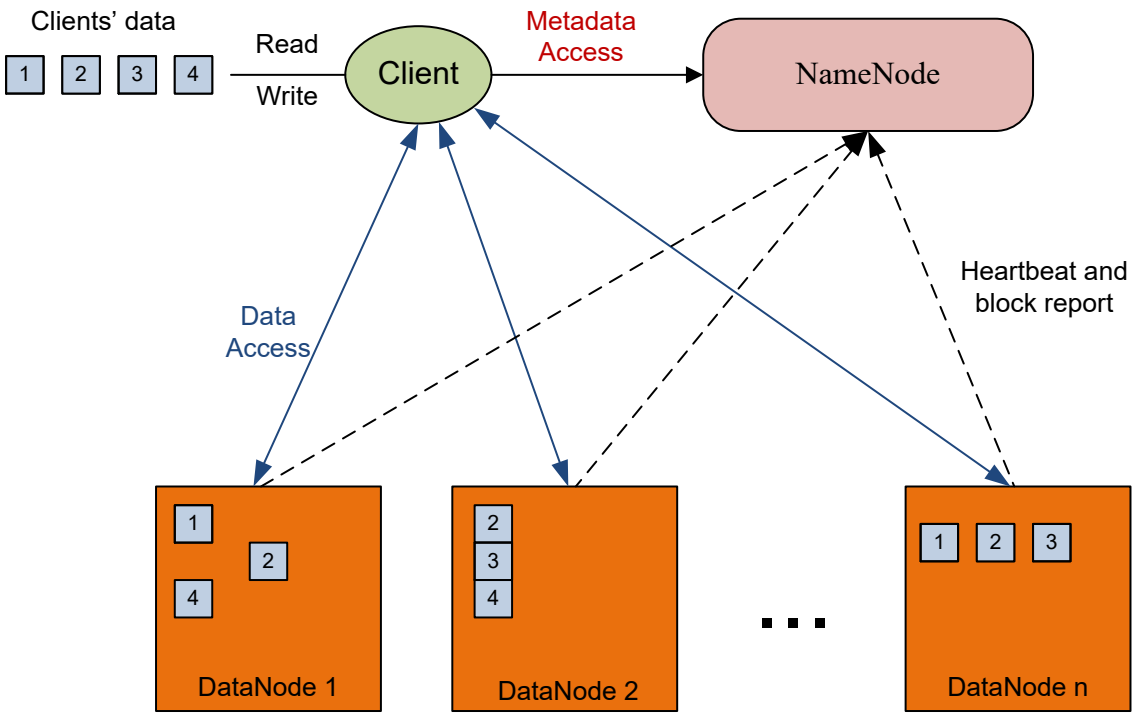


**Figure 2.1:** HDFS [9]

When a client application wants to read a file, it first communicates with the NameNode to get the information of which blocks made up the file and which DataNodes store those files. It then communicates with DataNodes directly to read the data blocks. When a client application wants to write a file to HDFS, it consults the NameNode and writes a block directly to the DataNode. This block will be replicated three times and stored in DataNodes. This process will be repeated for each of the file's data blocks. A DataNode sends its heartbeat to the NameNode regularly to keep the NameNode informed of its livness.

### *2.1.2 MapReduce*

MapReduce [7] is a programming model and technique for distributing a job across multiple nodes. It was created by developers at Google in 2004 aiming at building production search indexes. MapReduce has since then been used in many other applications by various companies and institutions. It works by separating the processing into two phases, the Map phase and Reduce phase.

The Map, with processing function Mapper, takes original input data and produces intermediate data as the input for the Reduce phase. In the Reduce phase, the function Reducer accepts intermediate data and merge together those intermediate values which have the same key. An illustration of MapReduce is shown in Figure 2.2. The input data are split into multiple subsets based on the `dfs.blocksize` which is the default block size for new files in HDFS. Each subset is fed into a Mapper which performs the desired operation. The Map tasks may finish at different times and the Reduce task starts to copy the Map outputs as soon as each Map completes. This is the *copy phase* of a Reduce task. After all the Map outputs have been copied, the Reduce task moves into the *merge phase* (also known is sort phase). During *merge phase*, key-value pairs with the same key are grouped together and are merged to generate one key with a list of values. The merged results are fed to the Reduce function in the last phase, the *reduce phase* for post-processing.

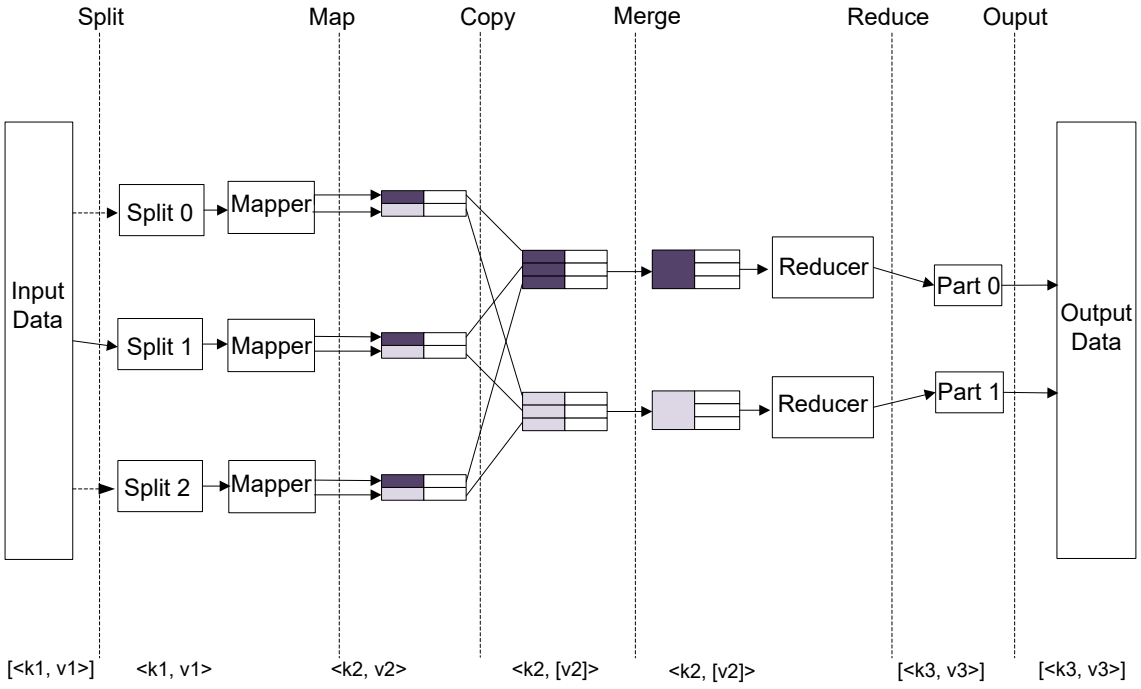


**Figure 2.2:** MapReduce Processing Procedure [8]

In order to meet the varieties of raw data types, key/value pairs are used as data units in the MapReduce framework. The data types of key and value can either be basic data structures (e.g. int, float, string) or complex data structures (e.g. list, array and self-defined data structures). In both the Map and Reduce stages, key/value pairs are used as input and output. In the formulas shown below: <…> stands for key/value pair, and […] stands for list:

$$\text{Map: } \langle k1, v1\rangle \rightarrow [\langle k2, v2\rangle] \tag{2.1}$$

$$\text{Reduce: } \langle k2, [v2]\rangle \rightarrow [\langle k3, v3\rangle] \tag{2.2}$$

Based on the above formula, the process of a MapReduce job is described as follows:

(1) The MapReduce application passes a [<k1, v1>] list as an input to the MapReduce processing model, e.g. [<String filename, String fileContent>]. The

processing model of MapReduce then splits the [<k1, v1>] list into single <k1, v1> key/value pairs and then assign those key/value pairs to Mapper to process.

(2) Mapper function processes <k1, v1> pairs with a user defined program and produces a [<k2, v2>] list. Each Mapper function is independent from each other, which means that the execution of one Mapper does not influence other Mappers.

(3) All the results of produced by the Mapper function are collected to form a large [<k2, v2>] list. Those results which have the same key are merged together to generate a new key/values pair, <k2, [v2]>, of which [v2] stands for a list of v2.

(4) The Reducer function processes <k2, [v2]> with a user defined program and generates the final results as a [<k3, v3>] list.

The MapReduce model includes two types of nodes for controlling a job execution [8]:

(1) The JobTracker is responsible for coordinating the job execution in the Master node of the MapReduce model. It assigns tasks to TaskTrackers, monitors the job running and will reschedule a task on a different TaskTracker if the task fails. There is only one JobTracker in a Hadoop cluster.

(2) The TaskTracker accepts tasks from the JobTracker and runs the tasks. The TaskTracker keeps sending the JobTracker a heartbeat message indicating it is alive and it also sends the progress reports of a task which keeps the progress record.

## 2.2 Clustering Analysis and Mahout Solutions

This section includes three sub-sections. In Section 2.2.1, the basic concepts of clustering analysis are presented and in Section 2.2.2 the clustering techniques in Mahout

are discussed. Lastly, Section 2.2.3 outlines how to evaluate the quality of clustering results.

### *2.2.1 Clustering Analysis*

The focus of this section is on describing the basic concepts and purpose of clustering analysis. In addition, a brief overview of the k-means [10] clustering technique is provided to show an example of how clustering is performed.

#### 2.2.1.1 Basic Concepts of Clustering Analysis

Clustering analysis [10] refers to the process of organizing items into groups based on their similarity [11]. The generated clusters consist of a set of items that are similar to one another in the same group but dissimilar from the items belonging to other groups. Clustering analysis is regarded as a form of classification since both techniques are used to divide items into groups. However, clustering analysis categorizes unlabeled items based only on the data and is thus referred to as unsupervised classification. On the contrary, classification techniques which assign new class labels to unlabeled items based on a labeled model are regarded as supervised classification.

There are several types of clustering approaches: hierarchical versus partitional, and exclusive versus fuzzy. Hierarchical clustering, as shown in Figure 2.3, is a set of nested clusters organized as a tree. It builds the hierarchy from the individual elements by progressively merging clusters. In this example, there are six elements from {1} to {6}. Node 1 and 2 are firstly merged together, because they are the closest elements. A sequence of merging is followed. Partitional clustering, as shown in Figure 2.4, is simply dividing elements into different clusters, and there is no overlap between clusters. A hierachical clustering can be regarded as a sequence of partitional clusters.

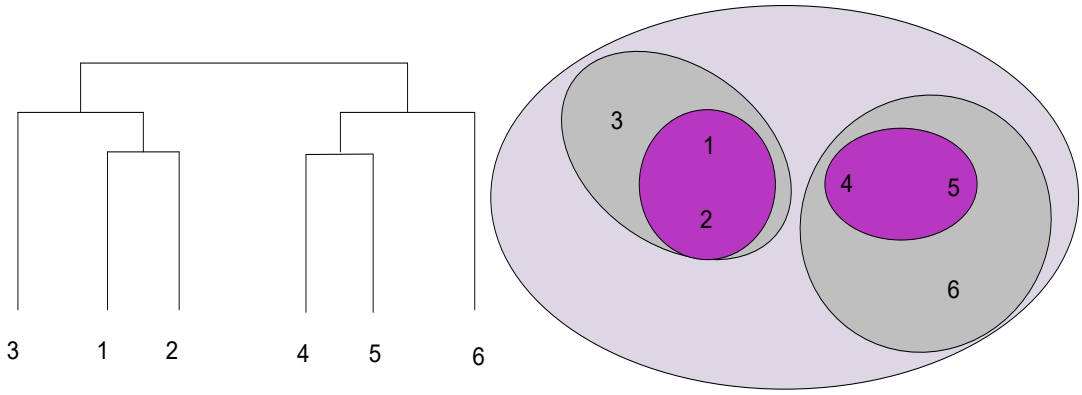


**Figure 2.3:** Hierarchical Clustering: an Example [10]

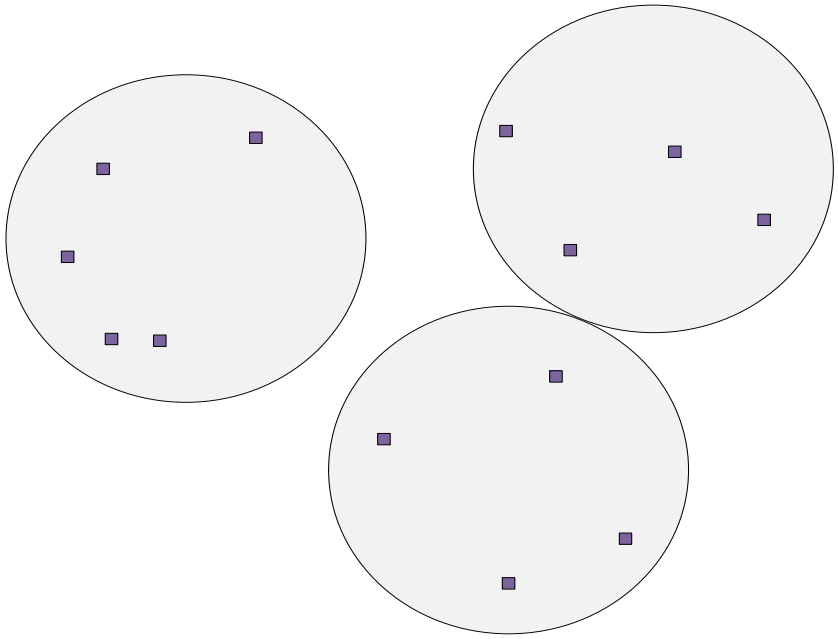

**Figure 2.4:** Partitional Clustering: an Example [10]

In exclusive clustering, each item belongs to just one cluster. However, during the process of fuzzy clustering, each item belongs to multiple clusters with a membership weight and the sum of membership weights is 1. An example of fuzzy clustering is fuzzy k-means [10].

#### 2.2.1.2 The Purpose of Clustering Analysis

Clustering analysis divides items into groups which can provide meaningful and useful insights into the data. Some applications use clustering analysis to achieve a better understanding of their dataset. For example, biologists can apply clustering techniques to find out a group of genes that have similar functions. Clustering can also be used by other applications to preprocess the data. For example, clustering can be used to summarize the dataset. Some data analysis techniques have a high complexity when processing a large dataset. Instead of analyzing the whole dataset, these techniques can be applied to a smaller dataset, which only consist of the clustering results (i.e. summarization of the data). The resulting accuracy of the analysis on the summarized dataset can still be comparable to that of analyzing the whole dataset. Clustering can also be used to efficiently find the nearest neighbors. Finding the nearest neighbors requires comparing each pairwise distance of all the points. Applying clustering analysis before finding the nearest neighbors can reduce the number of distance computations that need to be performed. The distances between objects residing in different clusters do not need to be computed since objects belonging to different cluster prototypes cannot be nearest neighbors of each other. Generally speaking, to apply clustering analysis formerly is more efficient than without it when finding nearest neighbors. Whether for understanding or utility, clustering analysis has been widely used in a variety of practical problems.

#### 2.2.1.3 K-means Clustering: An Example

K-means [10] is a partitional and exclusive clustering technique which creates a one-level partitioning of the data objects. It is one of the most widely used clustering algorithms. The description of k-means algorithm is depicted in Algorithm 2.1.

**Algorithm 2.1:** Basic K-means Algorithm

---

**1:** **Select** *K* points as initial centroids.

**2:** **Repeat**

**3:** Form *K* clusters by assigning each point to its closest cluster centroid.

**4:** Re-compute the centroid of each cluster with the points assigned to each cluster.

**5:** **until** Converge condition is satisfied

---

The user first chooses *K* initial centroids, where *K* is the number of clusters. In the k-means algorithm, *centroid* is defined as the mean of a group of points. Step 2 is to assign each point to the cluster which has the nearest centroid to this point. Closest is quantified by a proximity measure such as Euclidean distance, Manhattan distance and cosine similarity. Figure 2.5 illustrates an example of using the Euclidean distance and the Manhattan distance to measure the distance between two points, whereas Figure 2.6 presents an example of using the cosine similarity for distance measurement. The centroids are then updated. Next, steps 2 and 3 are repeated until the result *converges* which means that no more changes occur.

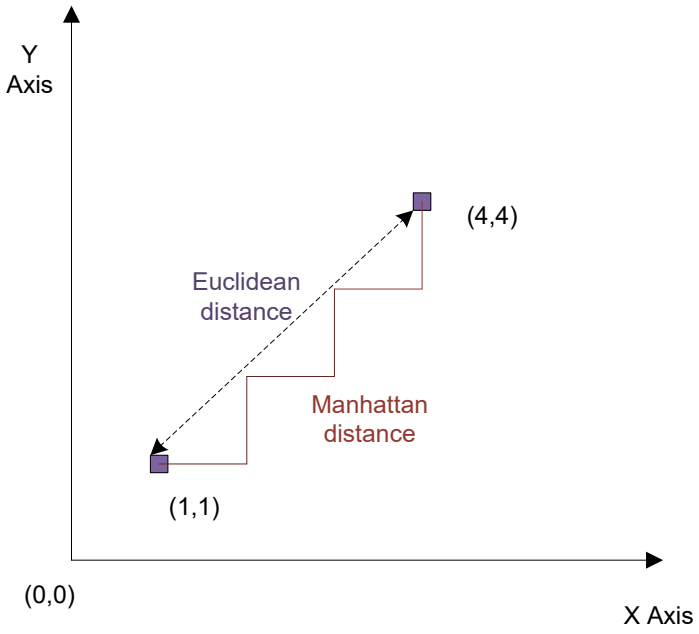


**Figure 2.5** Euclidean Distance and Manhattan Distance [11]

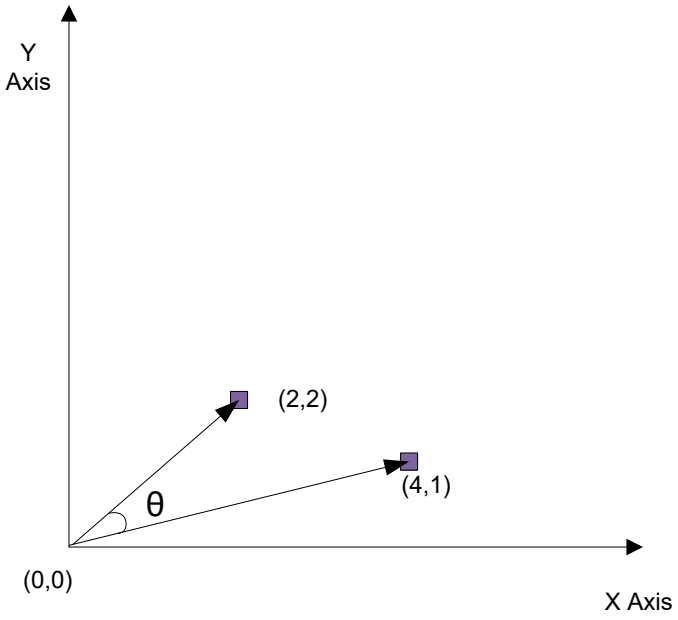


**Figure 2.6:** Cosine Similarity [11]

### *2.2.2 Clustering Methods in Mahout*

Mahout [3] is a machine learning library built on top of the Hadoop platform and uses the MapReduce paradigm. Mahout contains a collection of algorithms to solve recommendation, classification and clustering problems. It is an open source project and is currently still being developed and maintained. Mahout contains the implementation of various clustering algorithms, including k-means, fuzzy k-means and canopy clustering.

K-means aims to partition *n* items into *k* clusters in which each item belongs to the cluster with the nearest centroid. Fuzzy k-means [3] discovers soft clusters where a particular point can belong to more than one cluster. The k-means clustering algorithm in Mahout takes the following 5 parameters as input:

- The `SequenceFile` containing the input `Vectors`. `SequenceFile` is a Hadoop class which allows users to write arbitrary key/value pairs into it.
- The `SequenceFile` containing the initial centroids, and these centroids can either be randomly selected or set by users.
- The similarity method to be used. Mahout has implemented a variety of distance measurements, such as `EuclideanDistanceMeasure`, `SquaredEuclidean DistanceMeasure` and `ManhattanDistanceMeasure`.
- The `convergenceThreshold`, by which a clustering job will stop if each centroid changes less than this value after an iteration.
- The number of iterations. A clustering job will stop when it reaches this value even if the `convergenceThreshold` has not been reached.

The fuzzy k-means clustering algorithm also requires the same six input parameters as that of the k-means algorithm; however, it also requires an additional parameter, *fuzziness*, which is a coefficient normalization factor.

By using the MapReduce paradigm, clustering can be run on multiple machines with each Mapper processing a subset of the data points. The process of a k-means job in MapReduce paradigm is demonstrated in Figure 2.7.

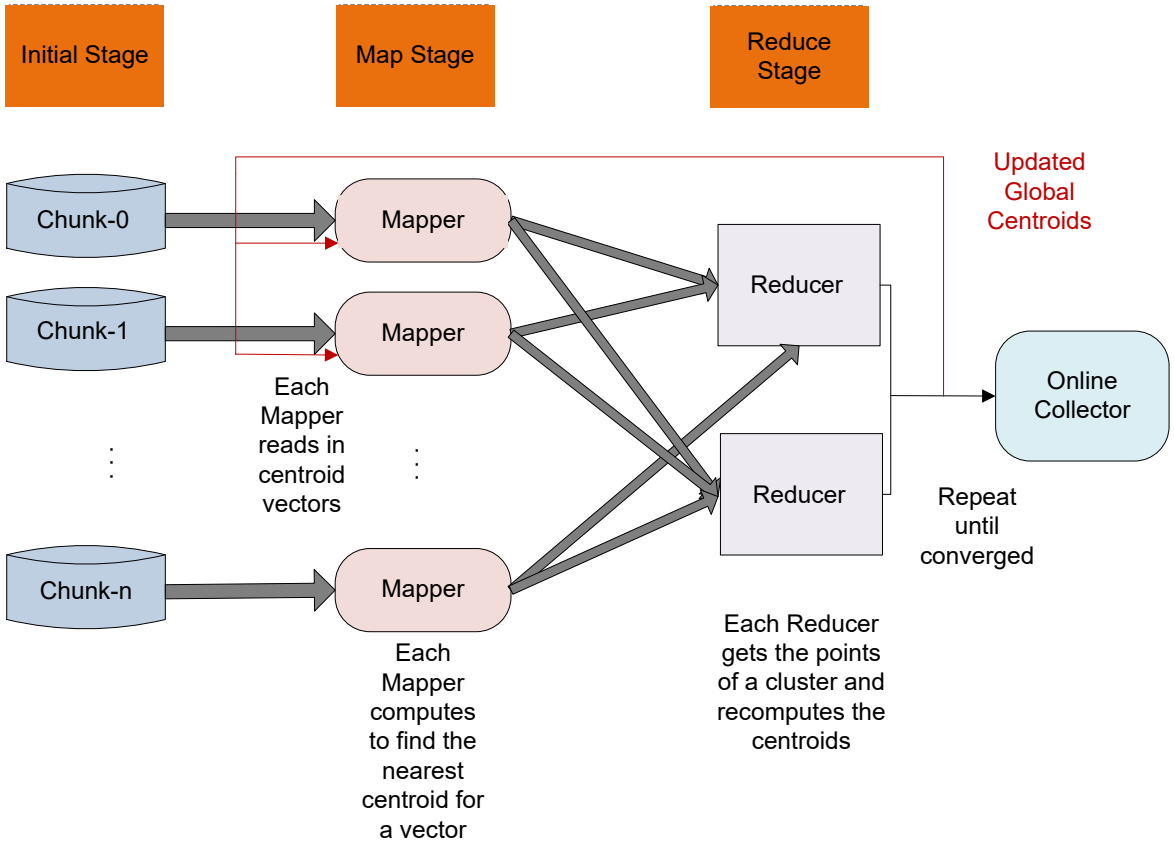


**Figure 2.7:** Procedure of K-means running as a MapReduce Job

In the initial stage, the input file is segmented into chunks based on the HDFS block size. Each chunk will be replicated and transferred to other computers. These chunks will be assigned to be processed by Mapper.

In the Map stage, each Mapper reads in the centroid vectors and then computes the distances between each point and the centroids to find the nearest centroid. Next, in the Reduce stage, each Reducer gets the partial sums of all points of a cluster from each Mapper and recalculates the centroid of each cluster. The result of the centroid calculations are fed back to the Map stage. This process continues until the threshold of convergence is reached.

### *2.2.3 Clustering Evaluation*

For supervised classification, clustering evaluation is part of the process of developing a classification model. However, for clustering that has unsupervised classification, cluster evaluation [10] is not commonly used as part of the clustering analysis. Even so, clustering evaluation, also referred to as clustering validation, is still important and necessary to be applied. Some evaluation criteria are shown below:

1. To check if the data has non-random structure.
2. To determine the correct number of clusters.
3. To evaluate the quality of the clustering analysis results.
4. To compare different sets of clusters to check which one is the best.

For unsupervised classification, not enough external information (e.g. the label of a data point) is provided as for supervised classification. Unsupervised measures thus are also called internal indices, since only the information exists in the dataset will be used. There are two widely used classes of measurements: the *cluster cohesion* (also called compactness, tightness, etc.) and *cluster separation* (also called isolation), which can measure the quality of clustering.

#### 2.2.3.1 Evaluation Metrics

Mahout [3] has implemented the measurements of using *inter-cluster* and *intra-cluster* distances to evaluate the clustering quality. Inter-cluster distance (as shown in Figure 2.8) is the distance between two cluster centers. A large inter-cluster distance represents that two clusters are more separated compared to clusters with a small inter-cluster distance.

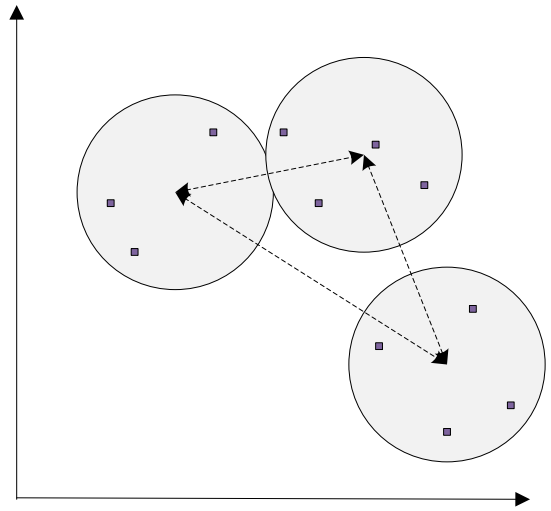

**Figure 2.8:** Inter-cluster Distance [11]

Since different distance measures (e.g. Euclidean distance, Squared Euclidean distance, and Manhattan distance, etc.) can be applied to find the similarity between points during clustering, these distance measures are used for cluster evaluation too. However, comparing the inter-cluster distances across two distance measures without normalizing them first is not meaningful [11]. For example, a pair of clusters with Euclidean distance of 10 will have a Square Euclidean distance of 100. Thus, Mahout implemented a measurement named *scaled average inter-cluster distance* for all centroid pairs to reduce the difference brought by distance measurements. The formula is shown in Eq. (2.3):

*Scaled average inter-cluster distance*$= (sum/count - min)/(max - min)$ (2.3)

*Sum* stands for the sum of all distances between each unique pair of cluster centroids. *Min* is the minimum distance among all the distances of two cluster centroids and *max* is the maximum distance among all the centroid distances. Lastly, *count* is the number of centroid pairs. For example, *n* centroids have *n(n-1)/2* centroid pairs.

Intra-cluster distance, as shown in Figure 2.7, is the distance between members of a cluster. A small value of intra-cluster distance means a tighter cluster.

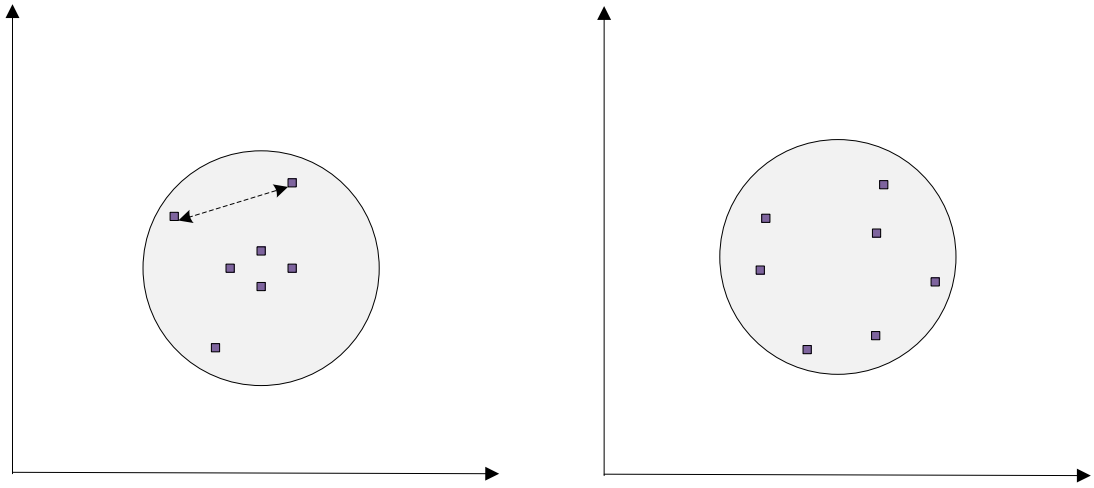

**Figure 2.9:** Smaller Intra-cluster Distance (Left) and Larger Intra-cluster Distance (Right) [11]

Scaled average intra-cluster distances is also defined in Mahout using Eq. (2.4) shown below:

*Scaled average inter-cluster distance*$= (sum/count - min)/(max - min)$ (2.4)

In Eq. (2.4), *sum* stands for the sum of all distances between each two points in one cluster. *Min* is the minimum distance among all the distances of two points and *max* is the

maximum distance among all the distances of two points. Lastly, *count* is the number of unique pairs of points within a cluster. For example, a cluster that has *n* points will have *n(n-1)/2* unique pairs of points.

## 2.3 Related Works

The work discussed in Section 2.3.1 concentrates on analyzing mobile network traffic data collected from a service provider. Following that, Section 2.3.2 introduces an article that tests the clustering methods implemented in Mahout. Lastly, Section 2.3.3 examines related work that describe how to setup Hadoop and tune its parameters to improve system performance.

### *2.3.1 Network Traffic Analysis*

With the popularity of mobile devices, more and more users are accessing the Internet using mobile devices (e.g. phones and tablets) constantly. To improve the performance of wireless networks and developing of the 5G network, network operators need to monitor and analyze the network traffic.

The authors in [12] analyzed the mobile Internet traffic data collected from a capital city in southern China. They analyzed the information of the operating system equipped in the mobile devices and manufactures of those devices to detect the popularity of different kinds of mobile devices. Besides, they analyzed the user behavior in 2G and 3G networks to find out the distribution of applications in 2G and 3G networks. They also analyzed the distribution of different services over one day time, and demonstrated the analysis results.

The authors in [13] examined a twelve-week trace of a building-wide local-area wireless network. They analyzed some overall user behavior such as how many users are active at a time and how much users move between access points. They also analyzed the

overall network traffic characteristics of this wireless network, such as network throughput and symmetry. These experimental results can help determine how wireless hardware and software should be optimized to handle traffic generated for this specific local-area network.

The authors of [14] developed a support vector machines (SVM) based approach to enable accurate re-identification of LTE transmitters. The data used in [2] are collected from towers in Ottawa, Canada and some useful features of LTE network are extracted and applied to the SVM model. Their result shows an accuracy of 98% for re-identification when SVM parameters are tuned and the training bin size is no less than 45 vectors.

### *2.3.2 A Comparison between K-means and Fuzzy K-means in the Cloud*

The authors of [15] compare k-means clustering and fuzzy k-means clustering algorithms in Mahout to perform clustering of Wikipedia's latest articles. From the experiment results, they notice that with fixed initial centroids, fuzzy k-means uses less time to converge compared with k-means and thus runs more quickly. However, when the initial centroids are selected randomly, they did not observe that fuzzy k-means executes more quickly compared to the k-means. The authors also compare the clustering quality of k-means and fuzzy k-means when applied to their dataset. They observed that fuzzy k-means produced lower quality clustering results. Based on their research, the authors commented that Mahout is a promising tool for performing clustering but the preprocessing tools need to be further developed.

### *2.3.3 A Mahout Test*

The authors of [16] have studied the performance of Mahout by using a large dataset comprising of tcpdump (a common packet analyzer) data collected from a US Air Force

local area network (LAN). The size of the dataset used in their experiment is 1.1 GB. They used Amazon EC2 with 4 EC2 Compute Units for their tests. The Mahout k-means clustering algorithm was executed in their work to test the performance of Mahout. They tested the scalability of Mahout with regards to the size of data size, and also compared the performance of processing the dataset on a single node versus multiple nodes. The results of their experiments showed that when the dataset is smaller than 128 MB, the advantage of distributing the process among multiple nodes cannot overcome the overhead of job initialization. Thus, for small dataset, it is better to use a single node. As the data size increases, the overhead can be compensated by distributing the process among multiple nodes and the performance gain reaches 351% for a 1.1 GB dataset.

## 2.4 Comparison with Related Works

This section presents the differences of this thesis compared to the related works described in Section 2.3.

The data used for this thesis was statistical traffic data provided by our industrial partner, Ericsson, Canada. Unlike the data analyzed in [12] and [13] which do not have base station information, experiment datasets in this thesis include records from more than 5000 base stations and each base station has 3 cells. To analyze the large volume of data from different base stations is a challenge. In order to have general information of all those base stations, we applied clustering algorithms to categorize the base stations with respect to the statistical information. Base stations with similar characteristics are grouped together which makes it easier to perform further analysis of the data.

Unlike experimental data in [14] that includes transmitter identification, our datasets do not have enough information that can be used as a label for each base station. Without

a label, supervised machine learning algorithms such as support vector machines (SVM) cannot be applied to our datasets. Hence, we applied clustering techniques, which is an unsupervised machine learning approach, to our datasets.

Authors in [15] and [16] have tested the Mahout clustering methods on Amazon EC2. In this thesis, we performed additional experiments using a network traffic dataset to compare the performance of Mahout's k-means and fuzzy k-means clustering algorithms. The results are presented in Section 4.3. Besides testing the performance of changing the size of the input dataset, this thesis also varies the number of Map and Reduce tasks in a Hadoop job (see Section 4.6). The experiments conducted to evaluate the performance of the Hadoop cluster are performed on a local high-performance server comprising of multiple virtual machines. Both the Hadoop master node and slave node are run on these virtual machines. A more detailed discussion of the experimental setup used in this research is provided in Section 4.2. Additional experiments are also conducted where the slave node is configured to have a different number of cores (discussed in Section 4.6).

# Chapter 3: Methodology and Approach

This chapter discusses the methodology and approach used in this thesis. It starts with the description of the experimental datasets in Section 3.1, and then goes on to describe the procedure used for data processing and analysis in Section 3.2.

## 3.1 Dataset Description

This section describes the base station datasets that are analyzed and experimented within this thesis. Section 3.1.1 and 3.1.2 demonstrates the data format and feature meanings of Dataset 1 and Dataset 2, respectively.

### *3.1.1 Description of Dataset 1 (2015 dataset)*

The first dataset, referred to as the 2015 dataset, used in this thesis is anonymized mobile network traffic data collected over a one week period from January 25, 2015 to January 31, 2015 (see Table 3.1). The data were collected from the cells of base stations and each base station has three unique cells. There are 5840 unique cells in this dataset. Each row in Table 3.1 is a record which contains the unique Cell ID and the values of the 24 features (shown in Table 3.2 and discussed below) collected from the cells over a one hour period from Start Time to End Time. Each unique cell (e.g. Cell ID 80004c2) collects the values of the 24 features every hour for one week. Therefore, each unique cell should have 168 records (24 hours per day × 7 days per week). However, there are some missing records for some cells at certain time slots. A discussion on how the missing records were dealt with is provided in Section 3.2.6. As shown in Table 3.1, the records are sorted by Cell ID, i.e., the 168 records for one unique cell are shown before the records of the next cell is displayed (see row 169 in Table 3.1).

**Table 3.1:** Sample Records of Dataset 1

| Row No. | Cell ID | Start Time | End Time | Feature 1 | Feature 2 | … | Feature 23 | Feature 24 |
|---|---|---|---|---|---|---|---|---|
| 1 | 80004c2 | 1/25/2015 0:00 | 1/25/2015 1:00 | 177 | 288 | … | 0 | 0 |
| 2 | 80004c2 | 1/25/2015 1:00 | 1/25/2015 2:00 | 86 | 208 | … | 0 | 0 |
| | … | … | … | … | … | … | … | … |
| 24 | 80004c2 | 1/25/2015 23:00 | 1/26/2015 0:00 | 27 | 25 | … | 0 | 0 |
| 25 | 80004c2 | 1/26/2015 0:00 | 1/26/2015 1:00 | 1 | 110 | … | 362 | 7 |
| | … | … | … | … | … | … | … | … |
| 168 | 80004c2 | 1/31/2015 23:00 | 1/31/2015 24:00 | | | … | | |
| 169 | 80024c1 | 1/25/2015 0:00 | 1/25/2015 0:00 | 23 | 30 | … | 1 | 2 |
| | … | … | … | … | … | … | … | … |

**Table 3.2:** Feature Description for Dataset 1

| Feature ID | Feature Name | Description |
|---|---|---|
| **1** | ActiveUeDlSum | Number of Active UE at DL |
| **2** | ActiveUeUlSum | Number of Active UE at UL |
| **3** | PdcpBitrateDlDrbMax | Max PDCP DRB Bit Rate in Mbps at DL |
| **4** | PdcpBitrateDlDrbMin | Min PDCP DRB Bit Rate in Mbps at DL |
| **5** | PdcpBitrateUlDrbMax | Max PDCP DRB Bit Rate in Mbps at UL |
| **6** | PdcpBitrateUlDrbMin | Min PDCP DRB Bit Rate in Mbps at UL |
| **7** | PdcpLatPktTransDl | PDCP Latency for Packet Transmission at DL |
| **8** | PdcpLatTimeDl | PDCP Latency Time |
| **9** | PdcpPktReceivedDl | Number of PDCP Packets Received at DL |
| **10** | PdcpPktReceivedUl | Number of PDCP Packets Received at UL |
| **11** | PdcpVolDlDrb | PDCP DRB Volume at DL |
| **12** | PdcpVolDlDrbLastTTI | PDCP DRB Volume at DL for the Last TTI |
| **13** | PdcpVolDlSrb | PDCP SRB Volume at DL |
| **14** | PdcpVolUlDrb | PDCP DRB Volume at UL |
| **15** | PdcpVolUlSrb | PDCP SRB Volume at UL |
| **16** | AvgRrcConn/Cell | Number of  Average RRC Connections/Cell |
| **17** | RrcConnMax | Number of Peak Connected Users/Cell |
| **18** | SchedActivityCellDl | Scheduling Time in Seconds per Cell at DL |
| **19** | SchedActivityCellUl | Scheduling Time in Seconds per Cell at UL |
| **20** | SchedActivityUeDl | Scheduling Time in Seconds per UE at DL |
| **21** | SchedActivityUeUl | Scheduling Time in Seconds per UE at UL |
| **22** | UeThpTimeDl | UE Throughput Time at DL |
| **23** | UeThpTimeUl | UE Throughput Time at UL |
| **24** | UeThpVolUl | UE Throughput Volume at UL |

Table 3.2 presents the 24 features used in the 2015 dataset including the unit used for each feature. The following list outlines the abbreviations used in Table 3.2:

- UE: User Equipment
- DL: Downlink
- UL: Uplink
- PDCP: Packet Data Convergence Protocol
- RRC: Radio Resource Control
- DRB: Data Radio Bearer
- SRB: Signal Radio Bearer

An example of UE is a mobile phone. PDCP is one of the user plane protocols used in LTE mobile networks. This protocol sends and receives packets from User Equipment and eNodeB (abbreviation for Evolved Node B), which is the hardware used to communicate directly with UEs in a mobile phone network. There are two kinds of PDCP bearers, which are data carriers in LTE systems, the SRB and DRB. SRB is used for carrying signals and DRB is used to carry User Plane content on the Air Interface. The Average RRC Connections is calculated by dividing the total number of RRC Connections with the number of samples.

### *3.1.2 Description of Dataset 2 (2013 dataset)*

The second dataset that is analyzed in this thesis, referred to as the 2013 dataset, is also wireless traffic data. The dataset has a similar format of Dataset 1 (shown in Table 3.3 and discussed in Section 3.1.1). The records were collected from 2808 Base Stations and each base station has 3 cells. Thus, this dataset should have 8424 (2808 × 3) unique cells. But some base stations did not have records for all three cells, this dataset only has records for 7592 unique cells. The data of these unique cells were collected every 15 minutes in

the morning from 6:00 to 7:00 and in the afternoon from 14:00 to 15:00 on November 18, 2013. As shown in Table 3.3, each row is a record which contains the Cell ID, the values of the 17 features (shown in Table 3.4 and described next), and the Start Time and the End Time of this recording. The description of the 17 features in the 2013 dataset is presented Table 3.4.

**Table 3.3:** Sample Records of Dataset 2

| Row No. | Cell ID | Start Time | End Time | Feature 1 | Feature 2 | … | Feature 17 |
|---|---|---|---|---|---|---|---|
| 1 | L0002c2 | 18/11/2013 6:00 | 18/11/2013 6:15 | 177 | 288 | … | 0 |
| 2 | L0002c2 | 18/11/2013 6:15 | 18/11/2013 6:30 | 86 | 208 | … | 0 |
| | … | … | … | … | … | … | … |
| 4 | L0002c2 | 18/11/2013 6:45 | 18/11/2013 7:00 | 27 | 25 | … | 0 |
| 5 | L0002c2 | 18/11/2013 14:00 | 18/11/2013 14:15 | | | | |
| | … | … | … | … | … | … | … |
| 8 | L0002c2 | 18/11/2013 14:45 | 18/11/2013 15:00 | | | | |
| 9 | L00012c1 | 18/11/2013 6:00 | 18/11/2013 6:15 | 1 | 110 | … | 7 |
| | … | … | … | … | … | … | … |

The following list outlines the abbreviations used in Table 3.4:

- UE: User Equipment
- TTI: Transmission Time Interval
- DL: Downlink
- UL: Uplink
- RRC: Radio Resource Control
- DRB: Data Radio Bearer
- SRB: Signal Radio Bearer

TTI is a parameter used in digital communication networks that refers to the duration of a transmission on the radio link. Occupancy denotes how much percentage the TTI has been occupied for data transmission. Note that the other terms, which are also used in the 2015 dataset (see Table 3.2) are explained in Section 3.1.1.

**Table 3.4:** Feature Description of Dataset 2

| Feature ID | Feature Name | Description |
|---|---|---|
| **1** | ActiveUeDlMax/TTI/Cell | Number of Maximum Active UE at DL/TTI/Cell |
| **2** | ActiveUeUlMax/TTI/Cell | Number of Maximum Active UE at UL/TTI/Cell |
| **3** | RrcConnMax/Cell | Number of Maximum RRC Connection/Cell |
| **4** | RrcConnAvg/Cell | Number of Average RRC Connection/Cell |
| **5** | ActUeDl/TTI (incl. Idle time) | Number of Active UE at DL/TTI (include idle time) |
| **6** | ActUeUl/TTI (incl. Idle time) | Number of Active UE at UL/TTI (include idle time) |
| **7** | ActUeDl/TTI (SE/TTI) | Number of Active UE at DL/TTI (Schedule Entity/TTI) |
| **8** | ActUeUl/TTI (SE/TTI) | Number of Active UE at UL/TTI (Schedule Entity/TTI) |
| **9** | DlDrbCellTput | DL DRB Cell Throughput in Mbps |
| **10** | UlDrbCellTput | UL DRB Cell Throughput in Mbps |
| **11** | DlDrbUeTput | DL DRB UE Throughput in Mbps |
| **12** | UlDrbUeTput | UL DRB UE Throughput in Mbps |
| **13** | DlTtiOccupancy | DL TTI Occupancy |
| **14** | UlTtiOccupancy | UL TTI Occupancy |
| **15** | TotalAvgTput (Vol/ROP) | Total Average Throughput Volume/ Recording Output |
| **16** | AvgServiceLength(s)/CU | Average Service Length/Connected User |
| **17** | AvgSessionLength(s)/CU | Average Session Length/Connected User |

## 3.2 Data Processing and Analysis Procedure

This thesis applies a sequence of methods to analyze the mobile network traffic datasets. The procedure shown in Algorithm 3.1 and illustrated in Figure 3.1 outlines the key steps used for analyzing the mobile network traffic datasets. Each of the steps shown in Algorithm 3.1 is discussed in a subsection that follows.

**Algorithm 3.1:** Data Analysis Procedure

---

**1:** Preprocess data (Standardization)
**2:** Find feature correlation (using R package)
**3:** **If** analyze with all feature information **then**
**4:** Apply k-means and fuzzy k-means clustering method
**5:** **Else if** analyze with reduced feature dimension **then**
**6:** Apply PCA
**7:** Apply k-means clustering method
**8:** **Else**
**9:** Select one feature per time
**10:** Convert data format (include handling missing time slots)
**11:** Apply k-means clustering method
**12:** **End if**
**13:** Evaluate and analyze results

---

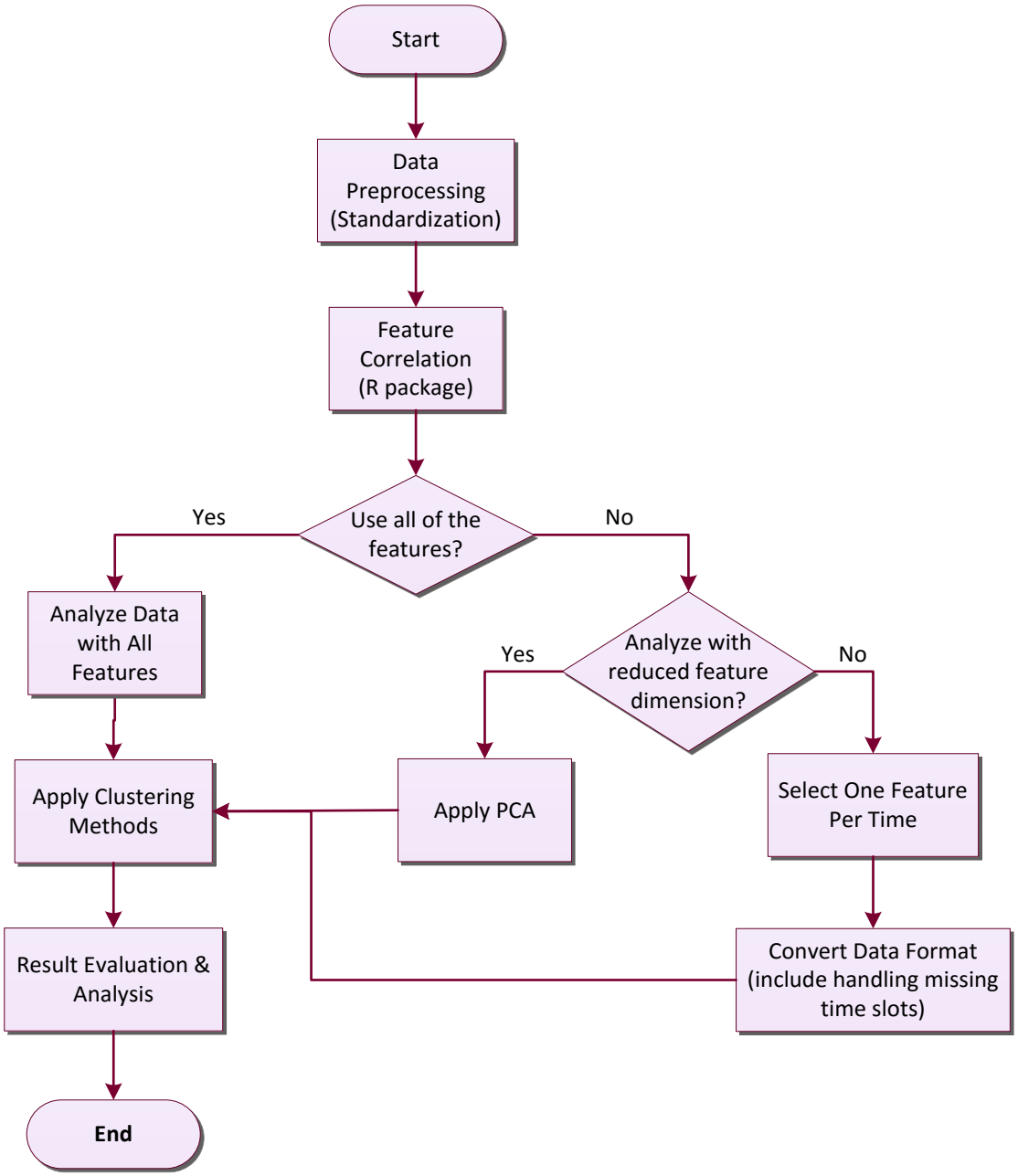


**Figure 3.1:** Flowchart of Data Analysis procedure.

### *3.2.1 Data Preprocessing*

Different features in the experimental dataset have different units, which means the data are collected using various measurements. Before applying any analysis method (e.g. checking the correlation of different features or clustering the dataset), we first applied a *standardization method* to eliminate the effect of different parameter units. Without the same scale, different parameter units will cause an unfair comparison when applied to some data mining techniques, for example, clustering. The standardization method that is used in this thesis is the `Scale()` function from one of the R packages: the 'caret' package (in short for classification and regression training) [17]. *R* [18] is a programming language and a free software environment which has been widely used for statistical computing. The `Scale()` function is implemented to create standardized scores (also known as Z-scores) and a standardized score indicates how many standard deviations an element is from the mean. A standardized score is calculated using the following formula in R:

$$\textit{Standardized score:} \qquad Z = \frac{x-\mu}{\sigma} \tag{3.1}$$

where Z is the standardized score, $x$ is the value of an element, μ is the mean value of all elements, and $\sigma$ is the standard deviation. If an element $x$ has a negative standardized score, it means that the element is less than the mean, while a positive standardized score denotes that an element has a value greater than the mean. An element with a standardized score of *1* (one) signifies that the value of the element is *1* standard deviation greater than the mean. After standardization, we get a preprocessed dataset with standardized data where each feature has the same weight.

### *3.2.2 Identification of Feature Correlation*

In order to have a better understanding of the relationship between different features, the next step is to calculate the correlations between each pair of features by using the `cor()` function [19] from the 'stats' package of R. The prototype of the `cor()` function is displayed below:

```
cor(x, y = NULL, use = "everything",
     method = c("pearson", "kendall", "spearman"))
```

where:

- `'x'` should be a numeric vector, matrix or data frame
- `'y'` by default is null, which means that `y` is set to `x`, and the self-correlation of matrix `x` should be calculated.
- `'use'` is an optional string which gives a method for computing covariance when there are missing values. It has several options, for example, `'use'` equals to "`everything`" means missing data will propagate to the end (retained in the final output file), `'use'` equals to "`all.obs`" means the presence of missing observations will produce an error, and `'use'` equals to "`complete.obs`" means the missing values are handled by deletion. In this thesis, we choose `'use'` equals to "`everything`".
- `'method'` is a string indicating the correlation coefficient that is to be computed. For our experiment, we choose the Pearson Coefficient [20] method.

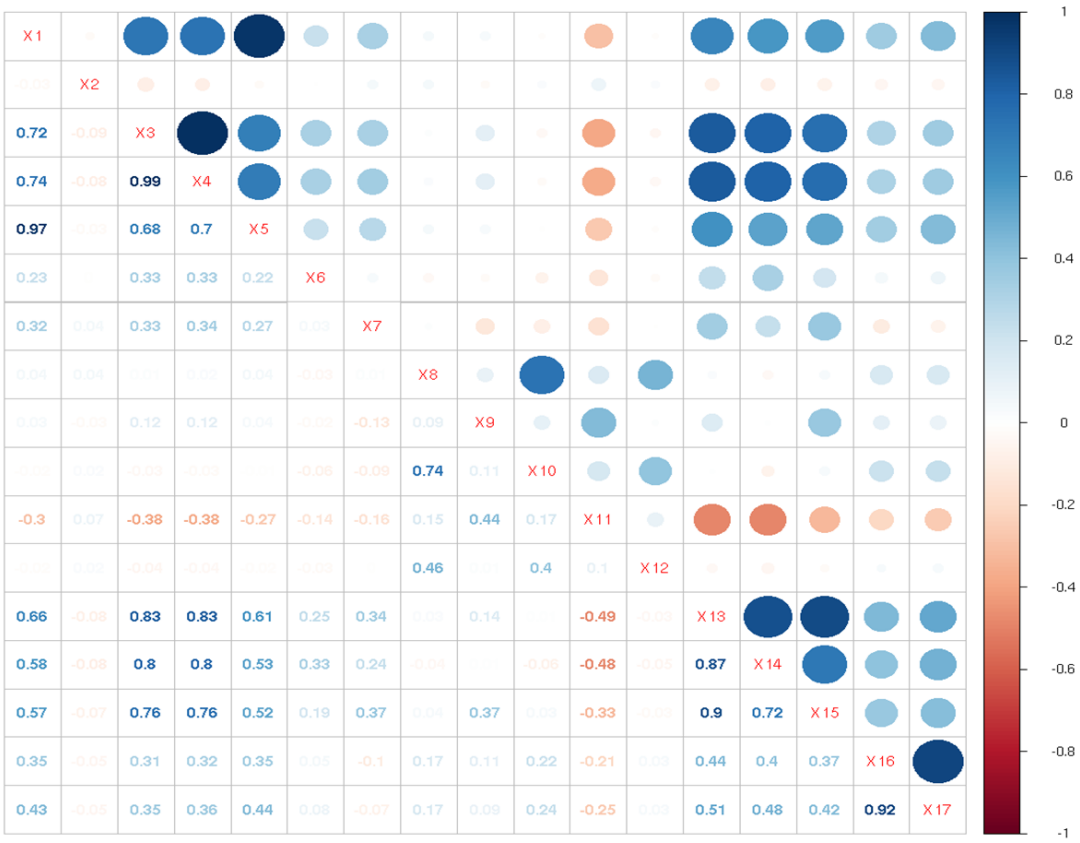


**Figure 3.2:** Sample Result of Correlation Matrix

A sample result of a correlation matrix for the Dataset 2 is presented in Figure 3.2. The diagonal values that start with an 'X' indicate the feature number. For example, X1 indicates the first feature, X2 represents the second feature, and so forth. The rest of the values of the matrix which are not the diagonal numbers represent the correlation between each feature pair. A value *in blue* (positive value) stands for a positive correlation between a pair of features (e.g. X1 and X3), which means that as one feature value increase, the value of the other feature will probably increase as well. The darker the color or the bigger the circle indicates the higher correlation between two features. Conversely, a *red* value (negative value), indicates a negative correlation between two features, which means that when the value of one feature increases, the value of another feature has a large chance to

decrease (e.g. X11 and X13). A large absolute value indicates that two features are strongly correlated, either positively or negatively.

### *3.2.3 Applying Clustering Methods*

Cluster Analysis (discussed in detail in Section 2.2.1) has been widely applied in a variety of areas, such as Data Mining, Pattern Recognition and Machine Learning. Clustering is used to organize items into groups from a large set of items based on their similarity. After the clustering, an item will share higher similarity with other items within the same cluster it belongs to compared to items belonging to different clusters. The k-means algorithm (discussed in detail in Section 2.2.1.3) is a widely used partitional clustering method that groups *n* items into *k* clusters in order to achieve high similarity among items inside each cluster. The similarity is calculated by measuring the distance between each item of a cluster and the average value (cluster center) of those items in the cluster. The k-means method first chooses *k* random points to be treated as the average value of a cluster, which is called the *cluster center*. After this, the k-means algorithm assigns the rest of the points to the nearest cluster based on the distances between this point and different cluster centers. Cluster centers are then recalculated and updated. The sequence of operations will be repeated until the no more changes occur, which means that the solution has converged. The steps of applying Mahout k-means/fuzzy k-means clustering method in this thesis are shown in Algorithm 3.2.

**Algorithm 3.2:** Steps of Applying Mahout K-means Clustering

**1:** Prepare data file and convert the data format to Mahout Vector.
**2:** Convert Mahout Vector to Hadoop SequenceFile.
**3:** Randomly select K centroids.
**4:** Copy the SequenceFile to HDFS.
**5:** Set Hadoop configuration parameters: `NameNode` IP
**6:** Set k-means clustering parameters: threshold,
**7:** Run Mahout k-means clustering.
**8:** Use ClusterDumper/SequenceFileDumper to get the clustering result.

### *3.2.4 Analysis of Clustering Results*

The final results of a clustering job reside in two files in the HDFS (Hadoop Distributed File System). One file named `output/clusters-n-final/part-r-00000` includes the information of the final iteration. The other file named `output/clusteredPoints/part-m-0000` contains the information of the clustered points describing which point each cluster is assigned to. All the files in HDFS are stored in the `SequenceFile` format, and need to be converted into text files when getting downloaded from the HDFS to a local machine. The format of getting a sequence file from HDFS to a local machine is shown below:

```
$mahout seqdumper -i /user/hdfs/mix_data/result/clusteredPoints –
o /home/mansi/Desktop/clusteredPoints-iris
```

The method `seqdumper` (in `SequenceFileDumper` utility) is used to dump a sequence file, where –i denotes the input file and –o determines the output file. The sample clustering result from analysis of the Iris Dataset [21] is shown in Figure 3.3. The Iris dataset is one of the most famous databases used in the area of pattern recognition and it was first

introduced by Fisher in 1936 [21]. The dataset contains 3 types of iris plants and gives the measurements in centimeters of the flowers' four attributes which include sepal length, sepal width, petal length, and petal width. Each iris data sample has a 4 dimensional vector that represent these four attributes, and a label to denote its flower type.

| Row ID | Seqdumper Output Result |
|---|---|
| **1** | Input Path: hdfs://namenode:54310/user/hdfs/mix_data/result/clusteredPoints/part-m-00000 |
| **2** | Key class: class org.apache.hadoop.io.IntWritable Value Class: class org.apache.mahout.clustering.classify.WeightedPropertyVectorWritable |
| **3** | Key: 90: Value: wt: 1.0 distance: 0.14694216549377498<br>vec: 4 = [5.100, 3.500, 1.400, 0.200] |
| **4** | Key: 90: Value: wt: 1.0 distance: 0.4381689171997474<br>vec: 4 = [4.900, 3.000, 1.400, 0.200] |
| **5** | Key: 90: Value: wt: 1.0 distance: 0.41230086102263686<br>vec: 4 = [4.700, 3.200, 1.300, 0.200] |
| **…** | … |
| **152** | Key: 144: Value: wt: 1.0 distance: 0.8357241788096976<br>vec: 4 = [6.200, 3.400, 5.400, 2.300] |
| **153** | Key: 51: Value: wt: 1.0 distance: 0.8345274136673622<br>vec: 4 = [5.900, 3.000, 5.100, 1.800] |
| | Count: 150 |

**Figure 3.3:** Sample Output File of Seqdumper Method

The first row in Figure 3.3 of the result shows the input path of the file. The second row displays the class types of the key and value parameters. Starting from the third row, the file begins to show the information of the clustered points. As shown in Figure 3.3, the value for the *key* parameter is a cluster identifier (integer value) and the *value* parameter contains a *weight* (*wt:* double value), a *distance* (double value) and a VectorWritable *vector* (*vec*). The *weight* indicates the probability that the vector (data point) is a member of the cluster. The *distance* is the distance measured between the cluster center and this vector by using a chosen distance measurement (e.g. Euclidean distance as discussed in Section

2.2.1.3). Lastly, *vector* is the vector of data point. If two vectors have different key values (cluster ID), it specifies that the two vectors are assigned to different clusters (e.g. row 3 and row 152). Vectors with the same key value, e.g. vectors in rows 3, 4, 5, means that they are close in distance and are assigned to one cluster.

Since the result from `seqdumper` does not have enough information on the clusters (e.g. cluster center vector, number of vectors assigned to the cluster and so on), we also applied `clusterdump` (from `ClusterDumper` utility) to inspect the clusters. The command used for dumping the cluster information is shown below:

```
$mahout clusterdump --input /user/hdfs/mix_data/result/clusters-100-final --pointsDir /user/hdfs/mix_data/result/clusteredPoints --output /home/mansi/Desktop/iris-k3-Euclidean.txt -e –dm
```

The `clusterdump` command requires the following parameters: an input file containing the cluster points (--input or -i), an output file where to store the results (--output or -o), whether to execute the algorithm to evaluate the clustering quality (-e), and the distance measurement to use (-dm) for evaluating the clustering quality.

A sample output file after applying the `clusterdump` method is shown in Figure 3.4. Rows 1, 69, 109 display the information of three clusters. *VL*- means the cluster has converged, while *CL*- (which does not occur in this sample result) means the cluster has not converged at the last iteration. The number following VL- is the cluster ID. The value of *n* is the number of items assigned to the cluster; whereas, *c* indicates the vector of cluster centers, and *r* shows the radius of the cluster.

| Row ID | Clusterdump Output Result |
|---|---|
| **1** | VL-51{n=62 c=[5.902, 2.748, 4.394, 1.434] r=[0.463, 0.294, 0.505, 0.295]} |
| **2** | Weight : [props - optional]:  Point: |
| **3** | 1.0 : [distance=1.2269752492315513]: 4 = [7.000, 3.200, 4.700, 1.400] |
| **4** | 1.0 : [distance=0.684140999639805]: 4 = [6.400, 3.200, 4.500, 1.500] |
| **5** | 1.0 : [distance=0.7315365210170275]: 4 = [5.500, 2.300, 4.000, 1.300] |
| **…** | … |
| **69** | VL-144{n=38 c=[6.850, 3.074, 5.742, 2.071] r=[0.488, 0.286, 0.482, 0.276]} |
| **…** | … |
| **109** | VL-90{n=50 c=[5.006, 3.418, 1.464, 0.244] r=[0.349, 0.377, 0.172, 0.106]} |
| **…** | … |
| **160** | 1.0 : [distance=0.1413930691370697]: 4 = [5.000, 3.300, 1.400, 0.200] |

**Figure 3.4:** Sample Output File of Clusterdump Method

There are a few differences between the output files generated by `Seqdumper` and `Clusterdumper`. First, the `Seqdumper` output file maintains the order of the input data vectors. For example, the vectors from Row 3 to Row 152 are the same vectors from vector 1 to vector 150 in the original input data file. On the other hand, the `Clusterdumper` output file has the information of a cluster (e.g. cluster center, number of points in this cluster). It also has the information of clustered points assigned to each cluster; however, the sequence of the vectors in the original input file is no longer maintained.

#### 3.2.4.1 Approach to Retrieval of Vector Information

An algorithm (see Algorithm 3.3), which is implemented in Java, is devised to output all the clusters (identified by cluster ID) and the data points that are assigned to the cluster (identified by the row ID from the original input dataset). The input to Algorithm 3.3 is the `Seqdumper` output file. A sample output file is shown in Figure 3.5.

**Algorithm 3.3:** The Proposed ClusterIdToIndex Algorithm

---

**1:** Create a `TreeMap` instance called `clusterIdToIndex` to store the final result.

**2:** Read in the `Seqdumper` output file one line at a time.

**3:** Split the record and get the cluster ID of each line.

**4:** Use a variable `index` to keep track of the row index and store the result in `clusterIdToIndex` by using cluster ID as a key and the index as a value.

**5:** **Repeat** 2-4 until all lines of the file are read.

**6:** Output item indices corresponding to the cluster ID.

---

| Row ID | Output Result |
|---|---|
| **1** | Cluster ID: 144:<br>53 78 101 103 104 105 106 108 109 110 111 112 113 116 117 118 119 121 123 125 126 129 130 131 132 133 135 136 137 138 140 141 142 144 145 146 148 149 |
| **2** | Cluster ID: 51:<br>51 52 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 102 107 114 115 120 122 124 127 128 134 139 143 147 150 |
| **3** | Cluster ID: 90:<br>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 |

**Figure 3.5:** Sample Output File of ClusterIdToIndex Algorithm

As shown in Figure 3.5, there are three clusters in the output result (identified by cluster ID). Data points which are assigned to each cluster are displayed by index following the cluster ID in the same row.

An illustration of the clustered points generated by the R tool is displayed in Figure 3.6. Note that the illustration shows the points plotted using the first three features. The three ellipses in different colors correspond to the three clusters.

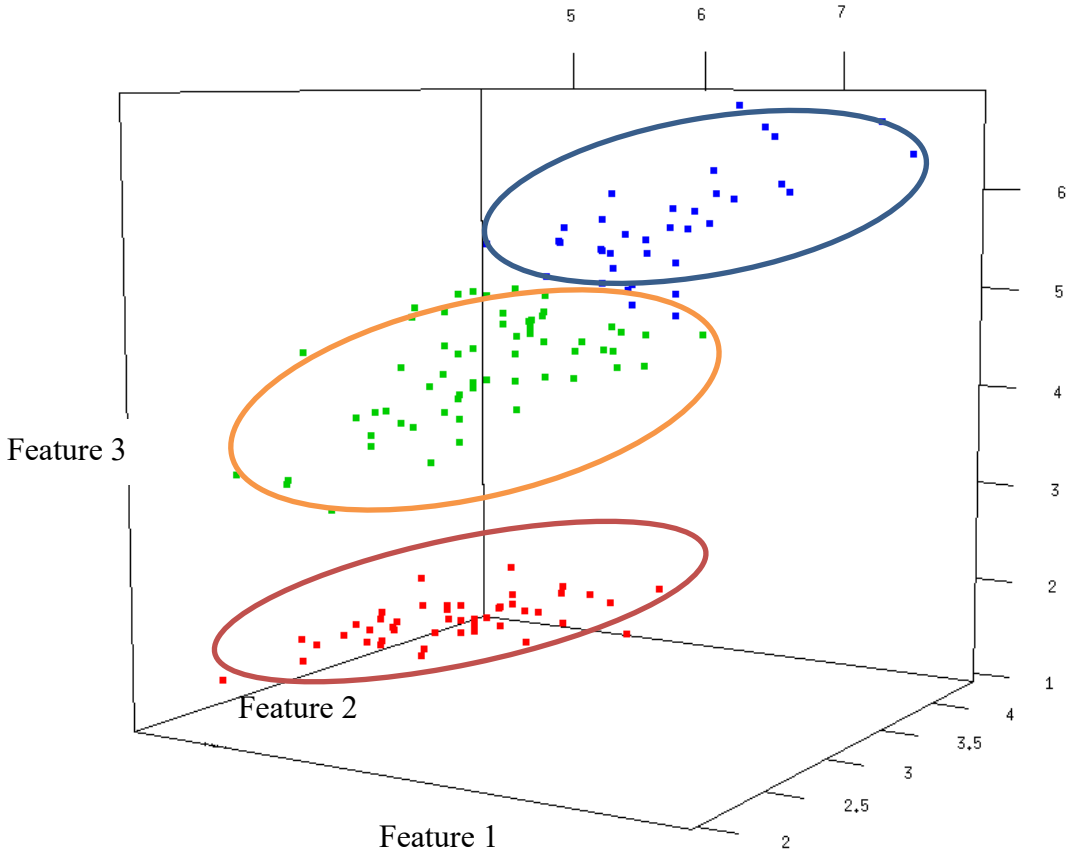


**Figure 3.6:** Illustration of Clustered Points from Iris Dataset [21].

### *3.2.5 Evaluation Result Matrix (Benchmark)*

As discussed in 2.2.3.1, scaled average *inter-cluster distance* (can be regarded as *separation*) and scaled average *intra-cluster distance* (*cohesion*) are two evaluation metrics

of clustering quality provided by the Mahout Library. A large scaled average inter-cluster distance represents a good clustering quality, since good clusters mostly do not have centroids that are close to each other. Scaled average intra-cluster distance computes the distance between members within a cluster. A small intra-cluster distance value shows that the points within the cluster are close to one another (i.e. a cohesive cluster), which is desired. Some research has used one or both of the two metrics to evaluate the clustering quality. For example, [15] uses inter-cluster distances to compare the clustering quality after applying k-means and fuzzy k-means algorithms on Wikipedia's Latest Articles. This thesis *combines* the two evaluation metrics together by using Eq. (3.2):

$$\frac{scaled\ average\ inter\_cluster\ distance}{scaled\ average\ intra\_cluster\ distance} \tag{3.2}$$

as a validity function. A large validity value is desirable, since a large inter-distance and a small intra-distance indicate a good quality clustering result.

To prove the utility of this validity function, this function was evaluated by applying it to the clustering result of the Iris Dataset. The steps used for testing the validity function (Eq. 3.2) is shown in Algorithm 3.4.

**Algorithm 3.4:** Steps of Testing the New Validity Function (Eq. 3.2)

---

**1:** Apply Mahout k-means clustering method to the Iris Dataset. Various experiments are performed for different *K* values from 3 to 10.

**2:** Use `ClusterDump` functions to get the *separation* and *cohesion* of each result.

**3:** Calculate the validity by using *separation/cohesion* (Eq. 3.2).

**4:** Compare and analyze the results.

---

The k-means clustering results of the Iris Dataset is shown in Table 3.5. From the results, we can see that $K$=3 provides the largest validity value and thus this value of $K$ should be chosen to cluster the Iris Dataset. Using $K$=3 also matches the structure of the Iris Dataset because the items are in 3 groups.

**Table 3.5:** Clustering Results of Iris Dataset

| K | Avg. Inter | Avg. Intra | Avg. inter/ Avg. Intra |
|---|---|---|---|
| **3** | 0.663 | 0.609 | 1.089 |
| **4** | 0.559 | 0.610 | 0.916 |
| **5** | 0.509 | 0.534 | 0.953 |
| **6** | 0.433 | 0.596 | 0.727 |
| **7** | 0.317 | 0.608 | 0.521 |
| **8** | 0.364 | 0.612 | 0.595 |
| **9** | 0.435 | 0.583 | 0.746 |
| **10** | 0.292 | 0.587 | 0.497 |

### *3.2.6 Dimension Reduction Using Principle Component Analysis (PCA)*

Dimension reduction refers to the process of reducing the number of variables (also called dimension) in dataset. With advances in data observation and collection, high-dimensional datasets (e.g. datasets with many features) are common. However, high-dimensional data analysis is mathematically challenging and time consuming. Some redundant information and noise can be present in the original dataset and thus impact the analysis results. With dimension reduction, the original data is mapped from a high-dimensional space to a lower dimensional space to simplify the analysis of the dataset.

Principle Component Analysis (PCA) is a well-known linear dimension reduction technique [4]. It reduces the dimension of a dataset by finding a few principle components with large variances. Principle components (PCs) are orthogonal and generated by a series of linear combinations of the variables. The first PC has the largest variance and the second PC has the second largest variance and so on. PCs are orthogonal to each other.

The PCA method used in this thesis is the `prcomp()` function the R library. The calculation is done by using a singular value decomposition (SVD) of the data matrix. The standard deviation, variance of each principle component, and the new matrix of variables (variables after conversion) are returned by this function. A certain number of PC variables are selected based on the variances of the PCs (discussed in Section 4.4.1). Following this, the k-means clustering algorithm is applied to the new dataset with reduced dimension and the original dataset with same K value. The clustering results of two datasets are compared using the Algorithm 3.5 shown as below.

**Algorithm 3.5:** Algorithm of Calculating Similarity of Two Clustering Results (Accuracy)

---

**1:** Use `ClusterDumper` to convert two output sequence files to text files.
**2:** Read in the two converted files and store the clustering results in two `ClusterResult` instances, named *result_a* and *result_b*. `ClusterResult` extends `TreeMap` class, and in `ClusterResult` the `key` is the cluster ID and the `values` is an array which stores the indices of points.
**3:** **ForEach** `key` (cluster ID) in *result_a*
**4:** **ForEach** value (point index) belongs to this key
**5:** **If** this value belongs to the same key in *result_b*
**6:** Increase count[key] by one
**7:** **End If**
**8:** **End ForEach**
**9:** Output count[key]

---

$$Accuracy = \frac{count[key_1] + count[key_2] + \cdots count[key_n]}{total\ number\ of\ Points} \quad (3.3)$$

### *3.2.7 Procedure to Analyze One Feature at a Time (Feature by Feature Analysis)*

For the analysis mobile network traffic, sometimes we want to see the distribution of the base stations based on just one feature. Some possible reasons for doing this are listed below:

1. Mobile network service providers sometimes are interested in one specific feature and want to see the behavior of base stations based on that feature.
2. Feature by feature analysis can provide us a more detailed result compared to using values of all features in the same experiment.

To perform feature by feature analysis, we first select one feature, and use a Java program to convert the data format. The algorithm of this conversion program is presented in Algorithm 3.6. The goal of the conversion program is to transpose the *start time* of the records to become their own columns. In other words, the *start time* now becomes the features of the record. A sample input dataset of this algorithm is shown in Table 3.6 (Feature 3, Max PDCP DRB bit rate at DL, is selected as an example). The data displayed in Table 3.7 are generated from the Dataset 1 after executing the conversion program. As shown, the features (column titles) are now the start times and each row corresponds to a unique cell ID that contains a record for one cell. For each cell, the value of the feature chosen to be analyzed (e.g. Feature 3 in Table 3.2), is shown at various times. Therefore, by analyzing this converted dataset, we can analyze how one feature changes at different points in time.

**Algorithm 3.6:** Data Conversion Algorithm

---

**1:** Sort data records by Base Station ID (does not exist in Dataset 1) and Cell ID.

**2:** Create an array to store the values of the current cell processed. The size of the array is equal to the number of time slots in the original dataset (e.g. the array size is 24 for 24 time slots)

**3:** Read in one data record at a time.

**4:** Get the Base Station ID, Cell ID, Start Time of the Record, and the value of the feature to analyze.

**5:** Store the feature value to the appropriate time slot in the array. Print the values of the array when all time slots the current Cell ID is read.

**6:** **Repeat** 3-5 until reaching the end of this file.

---

**Table 3.6:** Original Dataset before Conversion

| Row No. | Cell ID | Start Time | Feature 3 |
|---|---|---|---|
| 1 | L0002c2 | 18/11/2013 6:00 | null |
| 2 | L0002c2 | 18/11/2013 6:15 | null |
| | … | … | … |
| 4 | L0002c2 | 18/11/2013 6:45 | 1 |
| 5 | L0002c2 | 18/11/2013 14:00 | |
| | … | … | … |
| 8 | L0002c2 | 18/11/2013 14:45 | |
| 9 | L00012c1 | 18/11/2013 6:00 | 1 |
| | … | … | … |

**Table 3.7:** A Sample Result of Feature 3 (see Table 3.2) After Conversion

| Unique Cell ID | Cell ID | 6:00 | 6:15 | 6:30 | 6:45 | 14:00 | 14:15 | 14:30 | 14:45 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | L0002c2 | null | null | null | 1 | 1 | 1 | 1 | 1 |
| 2 | L0012c1 | 1 | 1 | null | 1 | 1 | 1 | 1 | 1 |
| 3 | L0013c1 | 1 | 1 | null | 1 | 1 | 1 | 1 | 1 |
| 4 | L0032c1 | … | … | … | … | … | … | … | … |
| 5 | L0032c2 | … | … | … | … | … | … | … | … |
| 6 | L0032c3 | … | … | … | … | … | … | … | … |
| … | | | | | | | | | |
| 48 | 2 | 1 | 1 | 1 | 2 | 36 | 26 | 16 | 19 |
| … | | | | | | | | | |
| 87 | 3 | 1 | 1 | 1 | 2 | 36 | 34 | 29 | 43 |
| … | | | | | | | | | |
| 185 | 1 | 1 | 1 | 1 | 1 | 18 | 23 | 20 | 19 |
| … | | | | | | | | | |

As we can see from the sample result in Table 3.7, there are some missing values for some cells at certain time slots. For example, the cell with ID L0002c2 has a missing value at the 6:00 time slot. To fill in these missing values so that further analysis (e.g. clustering) can be performed, we use the average value of the particular column (time slot) of where the missing value is located. For example, the missing value that is located in the 6:00 time slot is replaced with the average value calculated from all values in the 6:00 time slot. This method can minimize the impact of missing values since the average value will not influence the clustering result (the other values within the time slot will still contribute to the clustering result) [10].

# Chapter 4: Experimentation and Analysis

This chapter presents and discusses the results of analyzing Dataset 1 and Dataset 2. Section 4.1 describes the correlation of the features in Dataset 1 and Dataset 2 using correlation matrices. In Section 4.2, the setup and configuration of the Hadoop cluster used to execute the clustering algorithms in Mahout is explained. Following that, Section 4.3 presents a comparison of two standard clustering methods (k-means [10] vs. fuzzy k-means [10]) applied to our datasets. In addition, different *K* values are tested to see which *K* value can generate highest quality clustering result (i.e., largest value for the validity function defined in Eq. (3.2) and described in Section 3.2.5). Section 4.4 applies PCA to reduce the dimension of Dataset1 to investigate the improvement of the clustering execution time. In section 4.5, the feature Average RRC Connections/Cell is chosen for single feature analysis since it is a critical feature that is influential on performance. An analysis of how the value of this feature changes over a 24-hour period is provided in section 4.5. Lastly, Section 4.6 evaluates the performance of executing the k-means clustering technique on a Hadoop cluster.

## 4.1 Analysis of the Correlation of Features

To analyze the correlation of features, a *correlation matrix* is generated using the *R* tool as discussed in Section 3.2.2. The following sub-sections: 4.1.1 and 4.1.2 discuss the correlation matrices for Dataset 1 and Dataset 2, respectively.

### *4.1.1 Correlation Matrix for Dataset 1*

The correlation matrix for Dataset 1 is displayed in Figure 4.1. Color blue indicates positive correlation, and color red indicates negative correlation. The darker the color (or

the larger the circle size and the value), the stronger the correlation represents. The result shows that most of the features in Dataset 1 are strongly correlated (e.g. X9, X10 and X11).

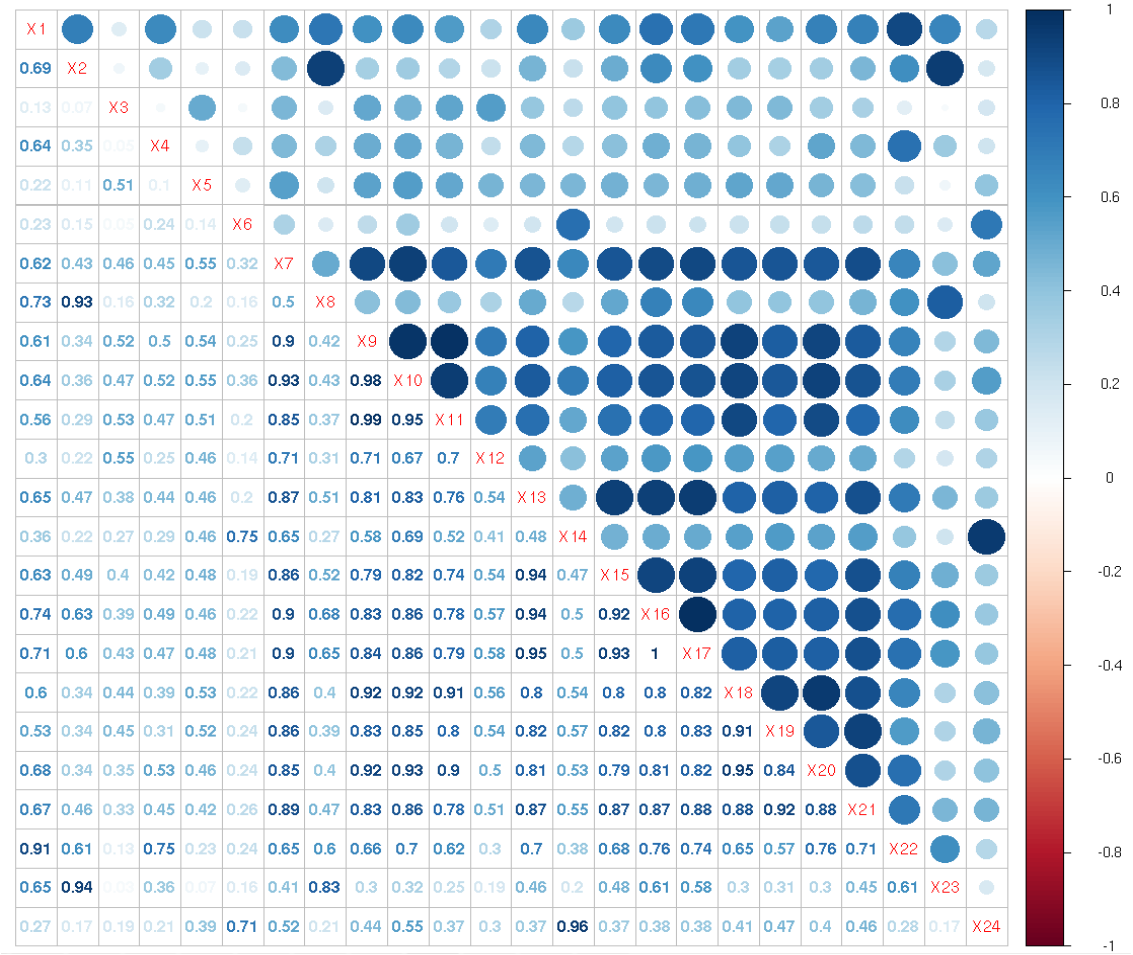


**Figure 4.1:** Correlation Matrix for Features of Dataset 1

### *4.1.2 Correlation Matrix for Dataset 2*

The correlation matrix for Dataset 2 is illustrated in Figure 4.2. The feature description of Dataset 2 is presented in Table 3.4. As shown in the figure, some feature pairs have a strong positive correlation (i.e., a large value) which means that as one feature value increases it is likely the other feature value increases as well. For example, features X3 (Maximum RRC Connection/Cell) and X4 (Average RRC Connection/Cell), and

features X1 (Maximum active UE at DL/TTI/Cell) and X5 (Active UE UL/TTI) are observed to have a strong positive correlation. Conversely, it is observed that some feature pairs (e.g. X11 and X1 and X11 and X3) are negatively correlated. X11 represents the downlink UE throughput and the negative relationship means that as the number of active UE (or the RRC Connections) increase, downlink UE throughput decreases. There are also some feature pairs that are loosely correlated or have almost no correlation (e.g. X7 and X8), and thus have a small absolute correlation value. X7 and X8 represent the number of active users at downlink and uplink, respectively. The features X7 and X8 are not strongly correlated, which indicates that downlink and uplink channels are independent in terms of user numbers.

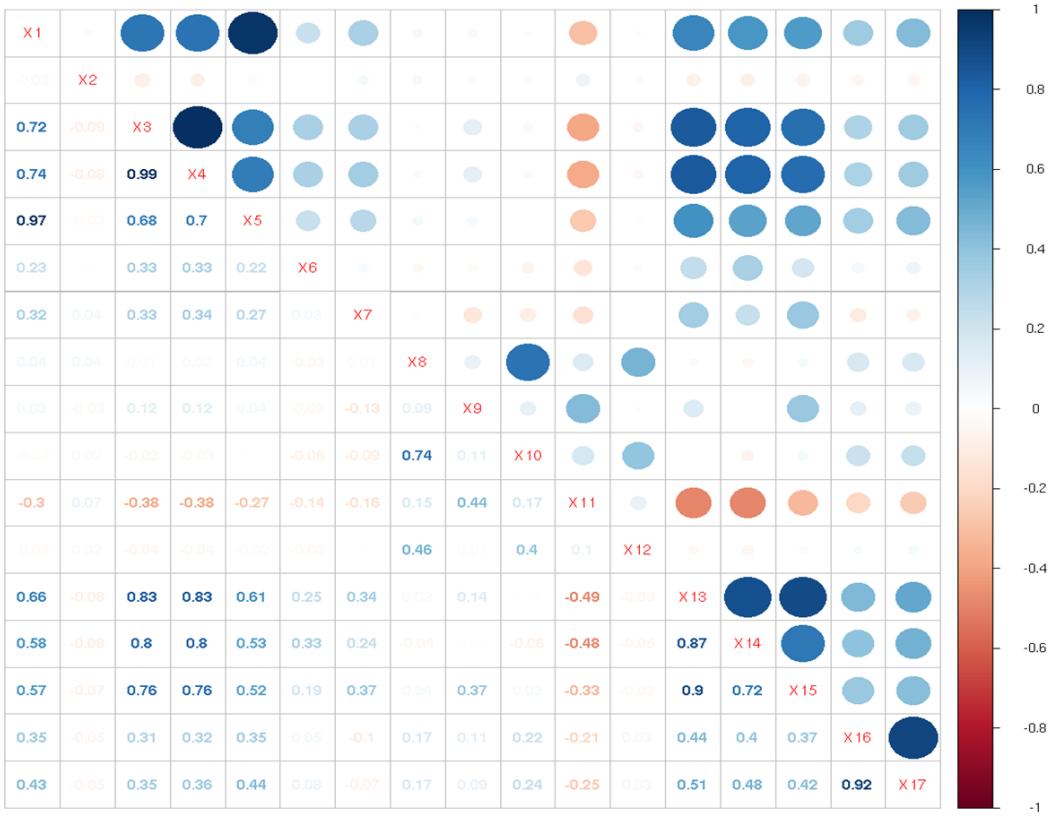


**Figure 4.2:** Correlation Matrix for Features of Dataset 2

## 4.2 Experiment Setup and Configuration of Hadoop Cluster

The Hadoop cluster is deployed on a local server with the following characteristics: 16 3.1 GHz processors (AMD Opteron Processor 4386) where each processor has 8 cores, 32 GB of memory, and 1770 GB of storage running on a 64-bit CentOS (Linux) 6.0 operating system.

As outlined in Table 4.1, five virtual machines (VMs) are deployed on the server. Table 4.1 shows the characteristics of each VM, including the number of virtual CPUs (processors), the amount of memory, storage size, and the operating system used. VM1 comprises 8 virtual CPUs and 16 GB of RAM. VM2 to VM5 each has a different number of virtual CPUs ranging from 1 to 8, but all of the VMs from VM2 to VM5 have 8GB of RAM. All the VMs have the same size of Virtual Disk of 100.73 GB, and run Ubuntu 14.04 (64-bit) operating system.

The Hadoop daemons (as discussed in Section 2.1) are executed on these VMs, and thus each VM is considered a node in the Hadoop cluster. VM1 operates as the *Master Node* of the Hadoop cluster and therefore executes the `NameNode` and JobTracker Hadoop daemons. The other VMs, VM2 to VM5, are considered the *Slave Nodes* and run the `DataNode` and `TaskTracker` Hadoop daemons. Note that the Hadoop cluster used in experiments comprises of one *Master Node* and one *Slave Node* as shown in Figure 4.3. Each time an experiment is run on the *Slave Node* that can be any of VM2, VM3, VM4, and VM5. As described in Table 4.1, each of those VMs has a different number of CPUs.

**Table 4.1:** VM Specifications inside the Host Server

| VM ID | Type | Number of Virtual CPUs | RAM Size |
|---|---|---|---|
| 1 | *Master Node* (NameNode +JobTracker) | 8 | 16384MB |
| 2 | *Slave Node A* (DataNode +TaskTracker) | 1 | 8192MB |
| 3 | *Slave Node B* (DataNode +TaskTracker) | 2 | 8192MB |
| 4 | *Slave Node C* (DataNode +TaskTracker) | 4 | 8192MB |
| 5 | *Slave Node D* (DataNode +TaskTracker) | 8 | 8192MB |

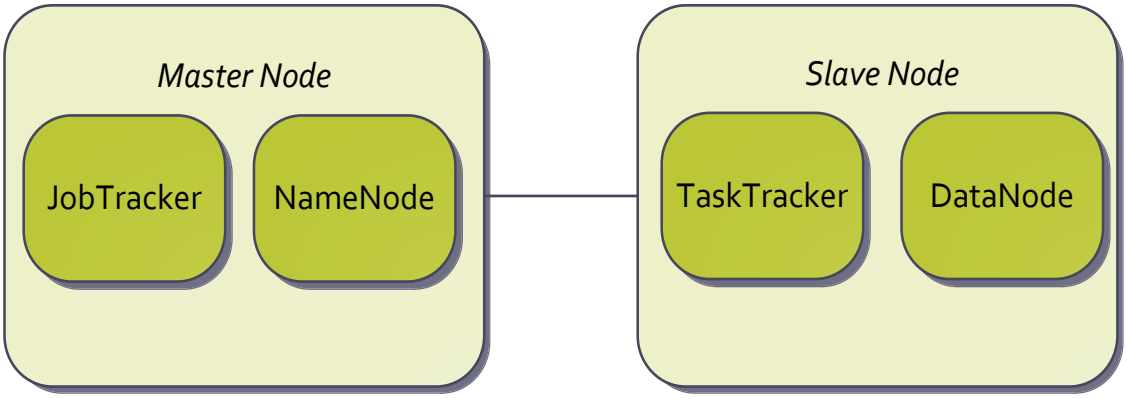


**Figure 4.3:** Cluster Set Up in the Experiment

## 4.3 Choosing Appropriate K and Fuzzy Values

This section presents and analyzes the clustering results of Dataset 1 and Dataset 2. K-means [10] and fuzzy k-means [10] techniques are used for clustering the experimental datasets in this thesis. Note that the inputs to the clustering techniques are the preprocessed versions of Dataset 1 and Dataset 2. The standardization technique that is used to preprocess the datasets is detailed in Section 3.2.1. We vary the K value for the k-means

technique to determine the K value that achieves the best validity value (refer to Eq. (3.2) in Section 3.2.5). Furthermore, additional experiments (which use the best K value identified in the previous experiment) are conducted to find the value of the fuzziness parameter for the fuzzy k-mean technique that generates the best validity value.

The Hadoop cluster used in this section is composed of the *Master Node* and the *Slave Node D* (with 8 CPUs as shown in Table 4.1). The convergence threshold (discussed in Section 2.2.2) of the k-means clustering algorithm is set to 0.01, which means that the difference between the last two iterations has to be less than 1% for the algorithm to stop. However, the maximum number of iterations is set to 150, hence even if the algorithm does not converge based on the threshold after 150 iterations the algorithm with still stop. Note that the number of iterations is set to 150 so that the experiments can complete in a reasonable amount of time if the algorithm does not converge. These configuration values are the same as those used in [16] .

In the experiments conducted in this research, the K value varies from 3 to 10 (see Table 4.2). For each experiment the following metrics are collected or calculated: scaled average inter-cluster distance (denoted as inter-cluster density), scaled average intra-cluster distance (indicated as avg. intra-cluster density), the validity value (calculated using Eq. (3.2), the number of iterations, and the execution time of each iteration. Since the initial centroids are selected randomly, each experiment is run three times and the average value of each metric is calculated. Running the experiments three times was adequate, because only little variation was observed in final results and all three runs converge to a similar solution.

Section 4.3.1 and Section 4.3.2 discuss the k-means and fuzzy k-means clustering results of Dataset 1, respectively. Following that, the results for k-means and fuzzy k-means of Dataset 2 are presented in Section 4.3.3 and 4.3.4, respectively.

### *4.3.1 K-means Results of Dataset 1*

This section applies k-means clustering method to Dataset 1. The K value varies from 3 to 10 (see Table 4.2). The parameters used in this section are discussed at the beginning of Section 4.3. Table 4.2 displays the results of applying k-means on Dataset 1.

**Table 4.2:** K-means Results of Dataset 1

| K (No. of Clusters) | Inter-cluster Density | Intra-cluster Density | Validity | Number of Iterations | Execution Time(s) / Iteration | Total Execution Time(s) |
|---|---|---|---|---|---|---|
| 3 | 0.498 | 0.701 | 0.710 | 34 | 46.22 | 1571.48 |
| 4 | 0.501 | 0.673 | 0.744 | 35 | 46.78 | 1637.3 |
| 5 | 0.440 | 0.663 | 0.664 | 60 | 48.73 | 2923.8 |
| 6 | 0.458 | 0.651 | 0.704 | 64 | 49.98 | 3198.72 |
| 7 | 0.396 | 0.644 | 0.615 | 96 | 51.69 | 4962.24 |
| 8 | 0.372 | 0.641 | 0.580 | 122 | 53.34 | 6507.48 |
| 9 | 0.350 | 0.642 | 0.545 | 135 | 54.68 | 7381.8 |
| 10 | 0.325 | 0.632 | 0.514 | 131 | 55.95 | 7329.45 |

As the K value increases, it is observed that in general the value of Inter-cluster density tends to decrease (except when K increases from K=3 to K=4 and from K=5 to K=6). The largest inter-cluster density, which means that the clusters have good separation, is achieved when K=4. The value of intra-cluster density decreases as K increases, except when K increases from 8 to 9. A smaller value for intra-cluster density indicates higher

cohesion. The largest validity value (defined in 3.2.5) is reached when K=4. Thus K=4 is the most appropriate K value for Dataset 1. As shown in Table 4.2, it requires 35 iterations for the k-means algorithm to converge when K=4. Lastly, it is also observed that the execution time for each iteration increases with an increase in K. This is expected because k-means requires calculating the distance between each data point and each cluster center. Therefore, a higher value of K (clusters) means that more calculations need to be performed causing a higher execution time.

### *4.3.2 Fuzzy K-means Results of Dataset 1*

The experiments in this section apply fuzzy k-means to Dataset 1 to find the *fuzziness* parameter that obtains the highest validity value.

**Table 4.3:** Fuzzy K-means Results of Dataset 1

| Fuzziness | Inter-cluster Density | Intra-cluster Density | Validity | Number of Iterations | Execution Time(s) / Iteration | Total Execution Time(s) |
|---|---|---|---|---|---|---|
| **1.0** | 0.501 | 0.673 | 0.744 | 35 | 67.05 | 2346.75 |
| **1.1** | 0.485 | 0.671 | 0.723 | 83 | 66.70 | 5536.1 |
| **1.2** | 0.469 | 0.668 | 0.702 | 69 | 66.51 | 4589.19 |
| **1.3** | 0.448 | 0.668 | 0.671 | 45 | 66.69 | 3001.05 |
| **1.4** | 0.431 | 0.660 | 0.653 | 34 | 66.95 | 2276.3 |
| **1.5** | 0.414 | 0.653 | 0.634 | 27 | 66.71 | 1801.17 |
| **1.6** | 0.394 | 0.656 | 0.601 | 21 | 66.45 | 1395.45 |
| **1.7** | 0.380 | 0.658 | 0.578 | 17 | 66.49 | 1130.33 |
| **1.8** | 0.382 | 0.684 | 0.558 | 14 | 66.69 | 933.66 |
| **1.9** | 0.394 | 0.614 | 0.642 | 45 | 66.79 | 3005.55 |

It can be observed from Table 4.2 that K=4 achieves the highest validity value when k-means is used to cluster Dataset 1. Therefore, K=4 is chosen for the fuzzy k-means algorithm used in this section which aims to find the value of the *fuzziness* parameter that achieves the highest validity value. The fuzziness parameter varies from 1.0 to 1.9. The experimental results of applying fuzzy k-means clustering to Dataset 1 are displayed in Table 4.3. The largest validity value is generated when fuzziness=1.0. Note that using fuzzy k-means with fuzziness=1.0 is identical to using normal k-means clustering [10]. This result indicates that k-means clustering with K=4 mostly fits Dataset 1, meaning that the data points in Dataset 1 can be divided into four well-defined clusters. By comparing the execution time for fuzzy k-means in Table 4.3 and k-means in Table 4.2, it is observed that the execution time per iteration for fuzzy k-means is longer compared to k-means. This is reasonable, because the complexity of the fuzzy k-means algorithm is more than the k-means algorithm.

### *4.3.3 K-means Results of Dataset 2*

This section applies k-means clustering method to Dataset 2. The parameters (except the K value for fuzzy k-means) used in this section are the same as those used in Section 4.3.1. Table 4.4 displays the results of applying k-means on Dataset 2. The results show that using k-means clustering with K=3 on Dataset 2 achieves the highest validity value.

**Table 4.4:** K-means Results of Dataset 2

| K (No. of Clusters) | Inter-cluster Density | Intra-cluster Density | Validity | Number of Iterations | Execution Time(s) / Iteration | Total Execution Time(s) |
|---|---|---|---|---|---|---|
| **3** | 0.531 | 0.679 | 0.782 | 93 | 25.64 | 2384.52 |
| **4** | 0.498 | 0.679 | 0.733 | 103 | 25.40 | 2616.2 |
| **5** | 0.433 | 0.680 | 0.637 | 111 | 25.53 | 2833.83 |
| **6** | 0.390 | 0.641 | 0.608 | 24 | 25.56 | 612.72 |
| **7** | 0.396 | 0.644 | 0.615 | 96 | 25.66 | 2463.36 |
| **8** | 0.414 | 0.672 | 0.580 | 142 | 25.67 | 3645.14 |
| **9** | 0.333 | 0.653 | 0.510 | 46 | 25.75 | 1184.5 |
| **10** | 0.365 | 0.635 | 0.575 | 136 | 25.87 | 3518.32 |

### *4.3.4 Fuzzy K-means Results of Dataset 2*

The experiments in this section apply fuzzy k-means to Dataset 2 to find the *fuzziness* parameter that obtains the highest validity value. Following the results described in Table 4.4, K=3 is chosen for the fuzzy k-means algorithm. The fuzziness value is varied from 1.0 to 1.9 and the results are displayed in Table 4.5. From the results, it is observed that fuzzy k-means with fuzziness=1.7 generates a better result than k-means method.

**Table 4.5:** Fuzzy K-means Result of Dataset 2

| Fuzziness | Inter-cluster Density | Intra-cluster Density | Validity | Number of Iterations | Execution Time(s) / Iteration | Total Execution Time(s) |
|---|---|---|---|---|---|---|
| **1.0** | 0.531 | 0.679 | 0.782 | 35 | 26.08 | 912.8 |
| **1.1** | 0.515 | 0.661 | 0.779 | 5 | 26.06 | 130.3 |
| **1.2** | 0.533 | 0.667 | 0.799 | 17 | 26.09 | 443.53 |
| **1.3** | 0.544 | 0.706 | 0.771 | 24 | 26.15 | 627.6 |
| **1.4** | 0.554 | 0.706 | 0.785 | 13 | 26.16 | 340.08 |
| **1.5** | 0.573 | 0.695 | 0.824 | 14 | 26.16 | 366.24 |
| **1.6** | 0.596 | 0.671 | 0.888 | 23 | 26.17 | 601.91 |
| **1.7** | 0.638 | 0.657 | 0.971 | 45 | 26.18 | 1178.1 |
| **1.8** | 0.365 | 0.654 | 0.558 | 6 | 25.99 | 155.94 |
| **1.9** | 0.564 | 0.654 | 0.862 | 25 | 26.12 | 653 |

## 4.4 Applying Principle Component Analysis (PCA)

This section discusses the performance and accuracy of applying PCA [4] technique before clustering. PCA is a technique used for linear dimension reduction. As observed in Section 4.1.1**,** Dataset 1 has 24 features and most of these features are strongly correlated. This phenomenon implies redundant features exist in Dataset 1 and thus processing the dataset with all the features, which requires a substantial amount of time, is not required. In Section 4.4.1, PCA is applied to Dataset 1 for dimension reduction and the results of principle components are displayed. Section 4.4.2 compares the performance of applying PCA before clustering and without applying PCA before clustering.

### *4.4.1 Determining the Number of Principle Components (PCs)*

There are 24 features in Dataset 1 and after applying PCA, 24 principle components are obtained. The variances of the 24 principle components are shown in Figure 4.4. The first principle has the largest variance value of approximately 15. The variance values are then observed to decrease rapidly for the following PCs (Principle Components), e.g., variance for PC2 is only about 2.5.

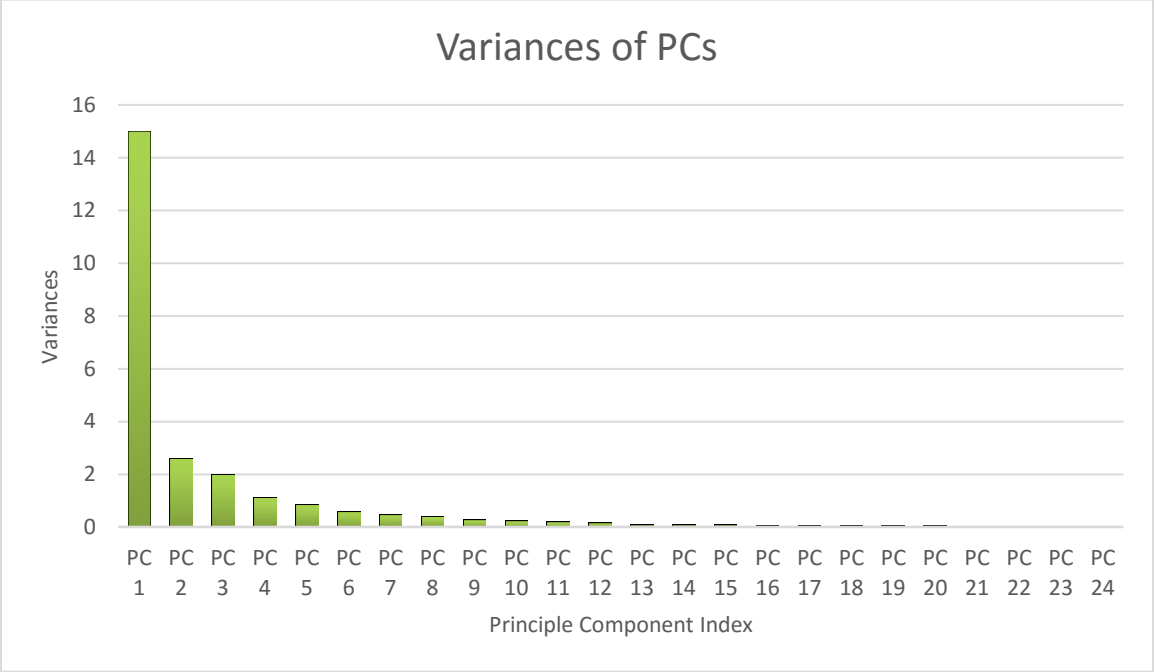


**Figure 4.4:** Variances of 24 Principle Components

The cumulative proportion of variance of the first *r* principle components are calculated using the following formula:

$$\textit{Cumulative proportion of variance:} \quad V = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{4.1}$$

where λ is the variance and *n* indicates the total number of PCs. The cumulative proportion of PCs are displayed in Figure 4.5.

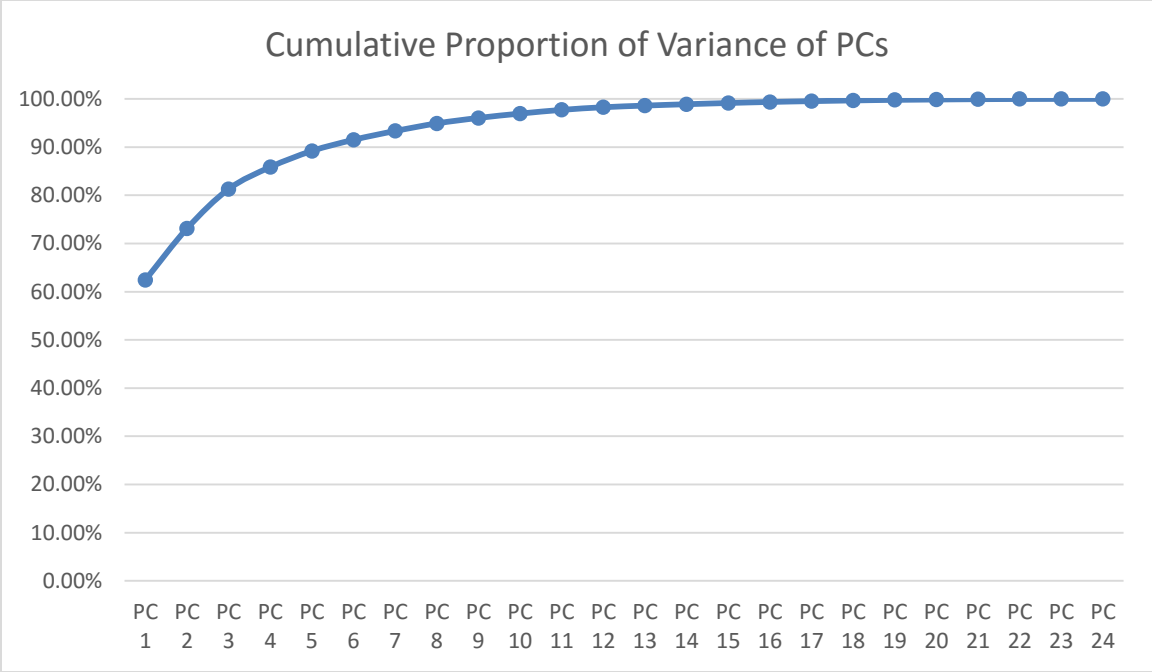


**Figure 4.5:** Cumulative Proportion of Variances of Principle Components

To determine the number of principle components to be retained, we set the threshold of $V$ to be $0.85$ [22] (demonstrated in formula 4.2).

$$V = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.85 \tag{4.2}$$

With the proposed threshold of Eq. (4.2), PCs 1 to 4 are retained as meaningful variables for further analysis.

### *4.4.2 Performance of Dimension Reduction Using PCA*

To evaluate the performance of PCA applied to our experimental datasets, an accuracy function (see Eq. (3.3)) is proposed. K-means clustering with K=4 is performed on Dataset 1 using PCA before clustering and without using PCA. Experiment *A* represents the clustering result of using all 24 features of the original Dataset 1 and experiment *B* denotes the clustering result of Dataset 1 using first 4 PCs after using PCA. Note that both

the experiment *A* and the experiment *B* are running on the Hadoop cluster composed of the *Master Node* (VM1) and the *Slave Node D* (VM 5). The number of Map tasks and Reduce tasks are both set to 1.

Table 4.6 displays the results of two clustering experiments: *A* and *B*. Since K=4 is chosen for both experiments, both A and B generate 4 clusters (identified by the cluster ID column in Table 4.6). The numbers displayed in the second, third, and fourth columns of Table 4.6 are the number of data records in each cluster. For example, there are 978108 data records in the input dataset, and 4644 of the 978108 data records are grouped in the first cluster in experiment A. A∩B (intersection of A and B) shows the number of data records that are grouped in the same cluster of experiment A and experiment B. For example, 4527 data records from *cluster 1* in A also appear in *cluster 1* in B.

**Table 4.6:** Comparison of Two Clustering Results

| Cluster ID | A | B | A∩B |
|---|---|---|---|
| 1 | 4644 | 4662 | 4527 |
| 2 | 84566 | 85371 | 84052 |
| 3 | 301373 | 302629 | 299442 |
| 4 | 587525 | 585446 | 584717 |
| Total | 978108 | 978108 | 972738 |

As proposed in Section 3.2.6, formula 3.3 is used in this section to evaluate the accuracy of using PCA. The total number of input records is 978108. The number of correctly assigned records with applying PCA is 972738. What is “correct” is judged by comparing the results with the results of A, as experiment A is based on all 24 features. Thus, accuracy of applying PCA (as shown in Eq. 4.3) is 99.45%.

Accuracy: $\frac{972738}{978108}$ =99.45% (4.3)

The comparison of time used for experiment *A* (24 features) and experiment *B* (4 PCs) are given in Table 4.7. With number of features decreases from 24 to 4, the input data size decreases from 236MB to 38MB and the average execution time/ iteration decreases from 46.78s to 32.69s. The total execution time of a k-means clustering job improves from 1637.30s to 1046.08s, or 36.11% reduction. Extrapolating these figures (i.e., data size and execution time) over a bigger data (GB or TB) indicates a substantial gain in memory space and time.

**Table 4.7:** Comparison of Execution Time for Experiment *A* and *B*

| Experiment ID | Size of Input Data (MB) | Avg. Number of Iterations | Avg. Execution Time(s) /Iteration | Avg. Total Job Execution Time(s) |
|---|---|---|---|---|
| A | 236 | 35 | 46.78 | 1637.30 |
| B | 38 | 32 | 32.69 | 1046.08 |

## 4.5 Analysis of the Impact of Avg. RRC Connections on Cells in a Day

This section discusses the results of analyzing a single feature for a whole day in order to have a better understanding of how the feature value changes in a day and how the base station cells are affected by the feature. The feature that is chosen for analysis is Feature 16 of Dataset 1 (see Table 3.2) which is the average number of radio resource control (RRC) connections per cell. The reasons for choosing to analyze Feature 16 are as follows:

1. Most features in Dataset 1 have a strong correlation with Feature 16 and therefore analyzing Feature 16 is representative of analyzing the other features of Dataset 1.

2. Average RRC Connections/Cell is a crucial parameter to mobile network operators since it can influence user experience and resource utilization.

The first step of this experiment is to select Feature 16 from Dataset 1 and convert it using Algorithm 3.6 to generate the dataset as shown in Table 4.8.

**Table 4.8:** Data for Feature 16 after Conversion Using Algorithm 3.6

| Row ID | Base Station ID | 0:00 | 1:00 | 2:00 | … | 12:00 | … | 22:00 | 23:00 |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 80004c2 | 0.014 | 0.035 | 0.017 | … | 9.561 | … | 0.029 | 0.013 |
| **2** | 80024c1 | 0.004 | 0.068 | 0.013 | … | 0.04 | … | 0.018 | 0.097 |
| **3** | 80024c2 | 0.011 | 0.011 | 0.011 | … | 0.013 | … | 0.014 | 0.008 |
| **…** | | | | | | | | | |
| **19** | 80134c3 | 19.422 | 17.432 | 1 | … | 21.603 | … | 24.350 | 23.157 |
| **…** | | | | | | | | | |
| **5840** | 99996c3 | 27.183 | 22.174 | 17.972 | … | 27.56 | … | 30.136 | 25.042 |

Each row has one unique *base station ID* and the values of Feature 16 collected every hour from the morning (0:00) to the end of a day (23:00) are shown. Note that there are 5840 unique base station cells in Dataset 1. This dataset is then applied to k-means clustering and is used to find the best number of clusters.

**Table 4.9:** Clustering Result of Feature 16

| K | Inter | Intra | Validity | Iteration |
|---|---|---|---|---|
| 3 | 0.466 | 0.631 | 0.739 | 48 |
| 4 | 0.455 | 0.603 | 0.755 | 26 |
| 5 | 0.416 | 0.589 | 0.706 | 47 |
| 6 | 0.378 | 0.561 | 0.581 | 97 |
| 7 | 0.353 | 0.591 | 0.597 | 71 |
| 8 | 0.329 | 0.623 | 0.528 | 89 |
| 9 | 0.306 | 0.552 | 0.554 | 96 |
| 10 | 0.292 | 0.587 | 0.497 | 100 |

Table 4.9 depicts the results of the clustering quality for different K values when Feature 16 in Dataset 1 (Average RRC Connections/Cell) is analyzed. Based on the results, K=4 is selected due to its high validity value. Hence the data records are grouped into 4 clusters.

**Table 4.10:** The Number of Cells for Each Cluster

| Cluster ID | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Number of Cells | 71 | 604 | 1935 | 3230 |

The k-means clustering result of Feature 16 with K=4 is displayed in Table 4.10. The number of unique cells assigned to each cluster is shown above. Cluster 1 has 71 unique cells and cluster 2 has 604 unique cells, and so on. Each cluster has a center vector that is calculated by averaging all of the points' values in a cluster. By plotting (see Figure 4.6) and analyzing the 4 center vectors, we can have a better understanding of how the base station cells in different groups are affected by Feature 16-Average RRC Connections/Cell.

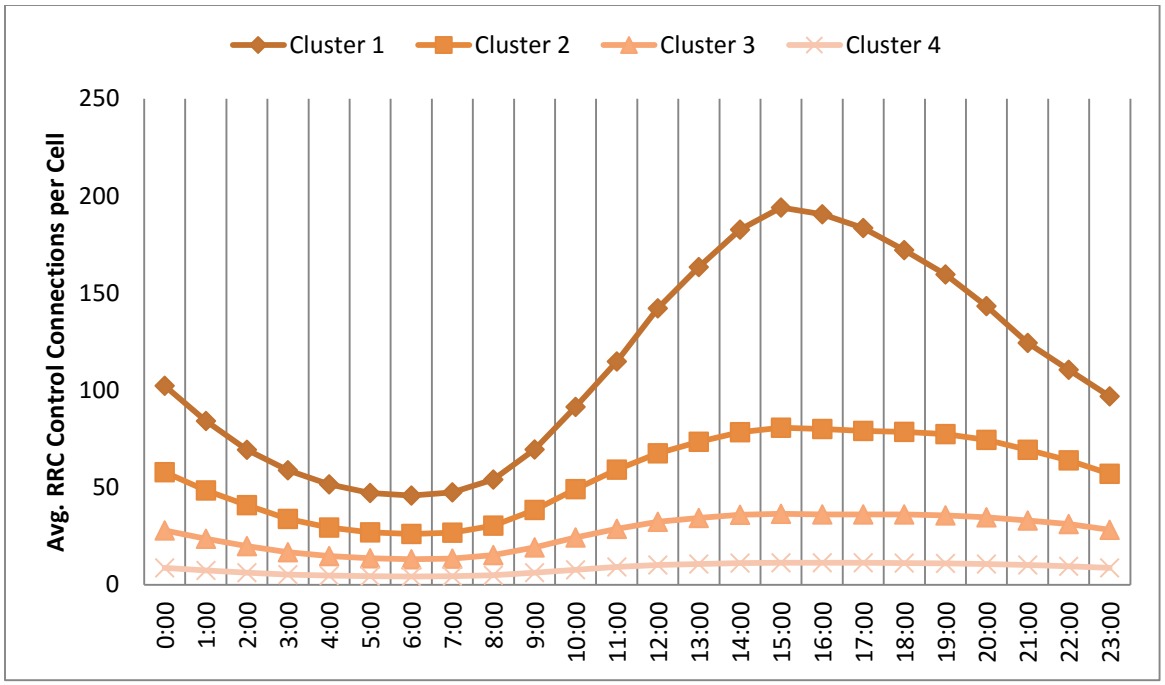


**Figure 4.6:** Average Number of RRC Connections per Cell at Different Time of a Day

Figure 4.6 shows the feature values of 4 cluster centers at different times of a day. The feature Average RRC Connections/Cell can also be denoted as average connected users/cell.

As seen in Figure 4.6, a large amount of cells have a small number of average connected users (RRC control connections) (see Cluster 4). 71 of 5840 cells have a large number of average connected users (see Cluster 1). As time changes from morning to night, we can see the number of average connected users changes. For Cluster 1, the value varies a lot during the whole day. It reaches its minimum value of 45.93 at 5:00 in the morning and achieves its peak value of 194 at 15:00. This makes sense since mobile network users are more active in the late afternoon compared to early morning. The cells in this cluster may need to be monitored closely as the number of users can be high, which may have impact on resource management and system performance. The values for Cluster 2 and

Cluster 3, however, does not vary as much as compared to Cluster 1, but it follows a similar trend. Cluster 4 has the least amount of connected users both in the morning and in the afternoon. This indicates that the cells in Cluster 4 either have only a small number of UEs or the UEs are not highly utilized.

Mobile network operators and users are concerned with QoS which can be evaluated by throughput. Since Dataset 1 does not have features related to throughput, the following experiment in this section uses Dataset 2 which has several metrics related to throughput (see Table 3.4). To understand the impact of Average RRC Connections/Cell on throughput, we select two throughput related features from Dataset 2 and analyze the relationship between these features. These two features are: (1) *downlink UE throughput* (Feature 11), (2) *uplink UE throughput* (Feature 12) (discussed in Section 3.1.2). To analyze the relationship of these two features with respect to the Average RRC Connections/Cell (Feature 4 in Dataset 2), these three features are selected and sorted in ascending order using Feature 4 (see sample data in Table 4.11). There are 60129 records in total for different base stations at different time. We group every 500 records and calculate the average value for each of these three features and then plot the result as depicted in Figure 4.7 (for Average RRC Connections/Cell and UE throughput results).

**Table 4.11:** Sample Result of Sorting Features 4, 9, 10 in Ascending Order of Feature 4

| Row ID | RrcConnAvg/Cell | DlDrbUeTput | UlDrbUeTput |
|---|---|---|---|
| **1** | 0.01 | 1.82 | 4.5 |
| **2** | 0.01 | 6 | 0.86 |
| **…** | … | … | … |
| **60128** | 578.53 | 0.59 | 0 |
| **60129** | 588.34 | 0.66 | 0 |

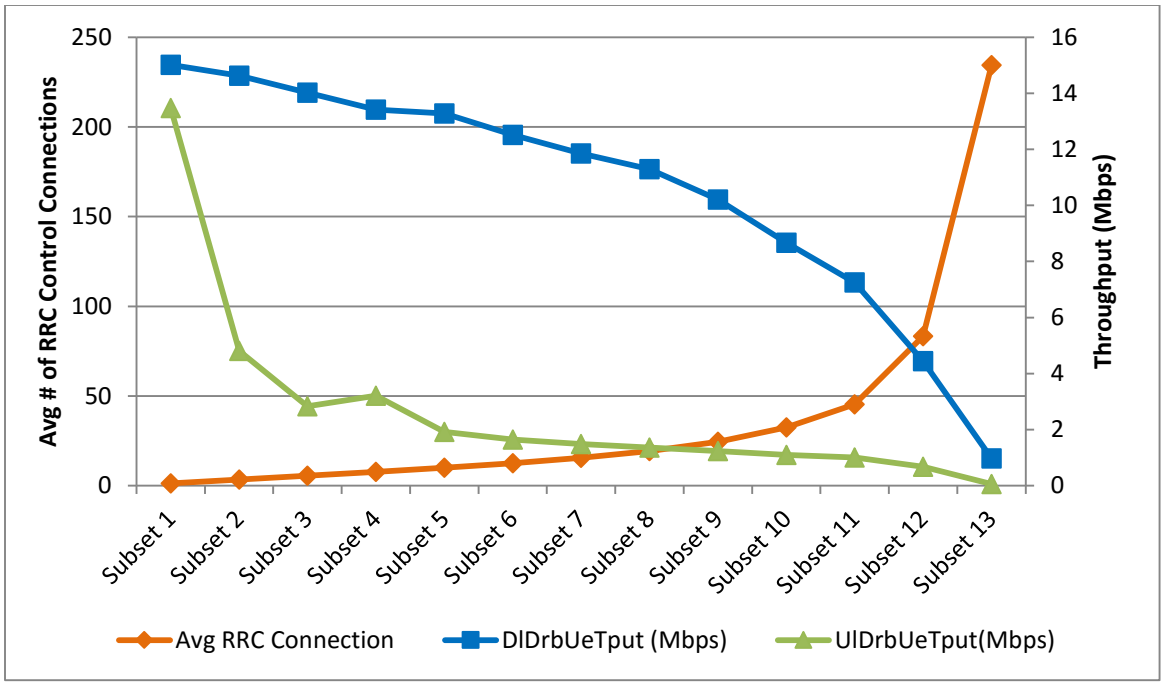


**Figure 4.7:** Relationship between Average RRC Connections per Cell and UE Throughput

Figure 4.7 has two vertical axes, the left axis is for Average RRC Connections/Cell and the right axis is for downlink and uplink throughputs. From the figure, it is observed that as the number of Average RRC Connections increases, both the downlink and uplink DRB UE throughputs decrease. The uplink DRB UE throughput drops quickly when the number of average connected users starts to increase and the downlink DRB UE throughput decreases more evenly and mildly. This figure demonstrates that a large value of the number of average connected users can lead to lower user equipment throughput (both downlink and uplink) and thus QoS can be affected. For example, Netflix, the popular video streaming service recommends at least 5 Mbps of downlink throughput for streaming HD (High Definition) quality video [23]. From this analysis, we observe that average connected users/cell has a strong negative correlation with UE throughput. Therefore, a

large number of average connected users/cell may degrade QoS because of the lower UE throughput. Conversely, a small number of average connected users/cell may not be desirable because it can lead to lower resource utilization if the service provider has a larger number of base stations.

The results of these experiments can help the network operator allocate and manage resources and improve QoS for users.

### *4.5.1 Analysis of Impact of RRC Connections for a Weekday*

To compare the distribution of base station cells between a weekend (Sunday, Jan. 25, 2015) and a weekday, an analysis of a weekday (Monday, Jan. 26, 2015) is performed. The k-means clustering technique is applied to this data and the number of clusters (*k* value) is chosen to be 4 since it best represents this dataset as described in Section 4.5. Four clusters are generated and the number of base station cells assigned to each cluster is shown in Table 4.12. For example, the first cluster includes 176 base station cells.

**Table 4.12:** The Number of Cells for Each Cluster

| Cluster ID | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Number of Cells | 176 | 853 | 2027 | 2802 |

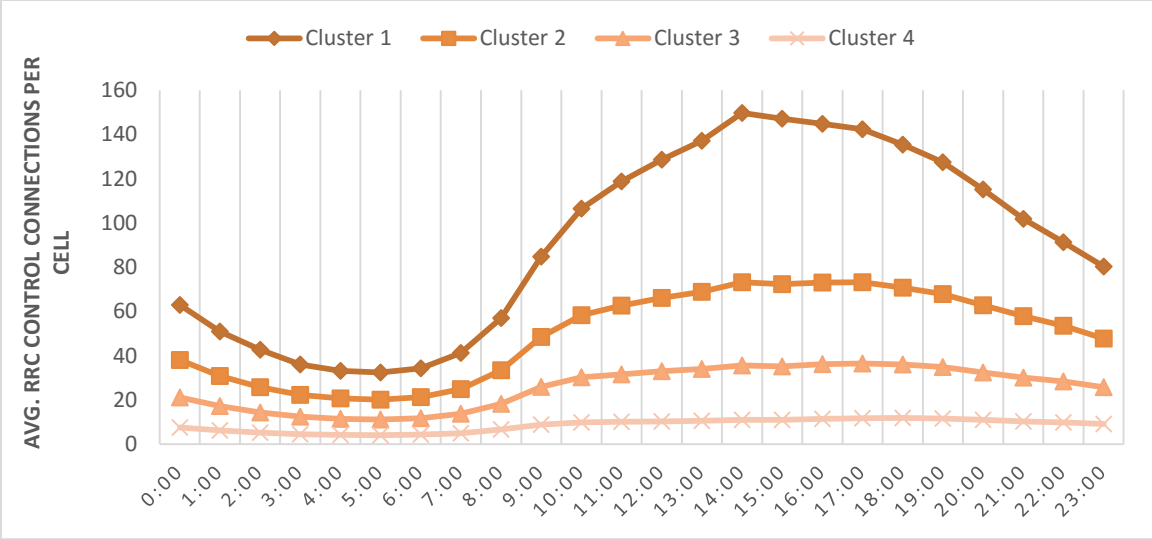


**Figure 4.8:** Average Number of RRC Connections per Cell at Different Times of a Weekday (Monday, Jan. 26, 2015)

Figure 4.8 shows the value of average RRC connections at different time slots of a weekday (Monday, Jan 26, 2015). By comparing the results of Figure 4.6 and Figure 4.8, it is observed that the overall trend is similar. However, cluster 1 of the weekday result (refer to Figure 4.8) has a slightly lower average value compared with cluster 1 of the weekend result (refer to Figure 4.6). In Figure 4.8, cluster 1 reaches the minimum of 32 at 5:00 and achieves the maximum of 150 at 14:00 compared to Figure 4.6 where cluster 1 achieves a minimum value of 45.93 at 5:00 and its peak value of 194 at 15:00. This is because cluster 1 of the weekday result has more cells and some of these cells have lower values of number of Avg. RRC Connections/cell compared with cluster 1 of the weekend result. Data of other weekdays are analyzed and the results follow the same trend as that of Monday as shown in Figure 4.8.

It was observed that the base station cells assigned to the first cluster in the results of Monday (Jan 26, 2015) and Tuesday (Jan 27, 2015) have 93 base station cells in

common, but only 3 cells are common in the results from Sunday (Jan 25, 2015) compared to Monday (Jan 26, 2015). A reason for this observation may be attributed to a majority of the base station cells in cluster 1 being located in an urban area where many individuals go to work. Thus, it would make sense that there is only a small number of common cells between the weekday and weekend results since fewer individuals will be in the same urban area on the weekend (unless there is a special event, for example). Unfortunately, the geographical location of the base stations are private and cannot be released to validate this observation.

## 4.6 Performance Evaluation of Hadoop

This section evaluates the performance of executing the k-means clustering technique on a Hadoop cluster. Section 4.6.1 investigates how the performance of the Hadoop cluster is affected when using k-means clustering with different sizes of input data, various numbers of Map tasks, and a different number of Reduce tasks. Section 4.6.2 compares the performance of running a clustering job on a Hadoop cluster that composed of execution nodes with different number of cores.

The default values for the parameters used for all clustering experiments in this section are listed below:

- Clustering method: k-means with K=4,
- convergeThreshold=0.01,
- distanceMeasurement=EuclideanDistance,
- MaximumIterations=150.

### *4.6.1 Evaluation of Parallel Execution*

Experiments in this section are executed on the Hadoop cluster composed of the *Master Node* (VM1) and the *Slave Node D* (VM5 with 8 CPUs). The k-means clustering algorithm with the same parameter values (–k 4 –c 0.01 –x 150 –d Euclidean.distance) are applied to five different experimental datasets. The first experimental dataset is Dataset 1 (discussed in 3.1.1) which comprises data for one week and is 236 MB. The other four datasets contain two to five weeks of data with sizes ranging from 472 MB to 1180 MB (with 236 MB increment). Besides changing the size of experiment dataset, the numbers of Map and Reduce tasks running in parallel are also varied to investigate how the Hadoop cluster performance is affected. Figure 4.9 and Figure 4.10 illustrate the execution time of the Map and Reduce stages, respectively, when the dataset of various sizes are used.

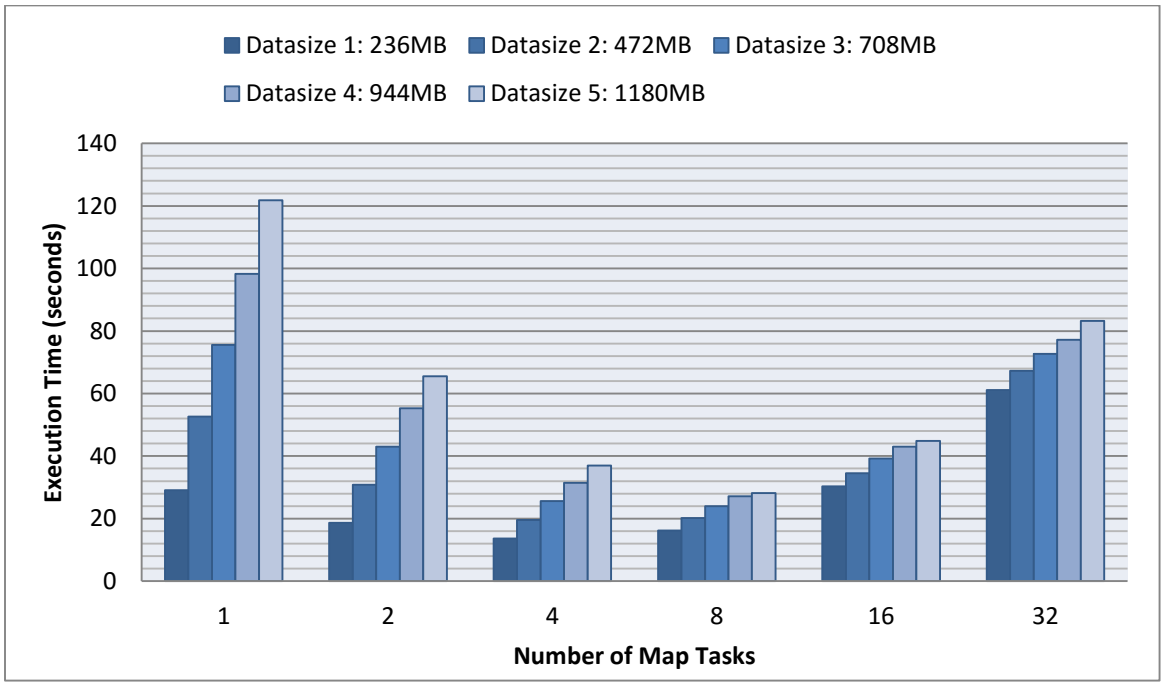


**Figure 4.9:** Effect on the Execution Time of the Map Stage When Varying the Number of Map Tasks

Figure 4.9 is generated by varying the number of Map tasks and changing the dataset size. Overall, the performance improves when the number of Map tasks increases up to 8. The execution time increases for both larger dataset sizes, as the number of Map tasks increases to 16 or 32. When the data size is small (236 MB), using a large number of Map tasks (e.g., 8, 16 or 32) does not generate a better performance, since there is an overhead for generating Map tasks and managing the splits. As the size of the input dataset increases, using a larger number of Map tasks (i.e., 8 Map tasks) improves the performance because of increased parallelism.

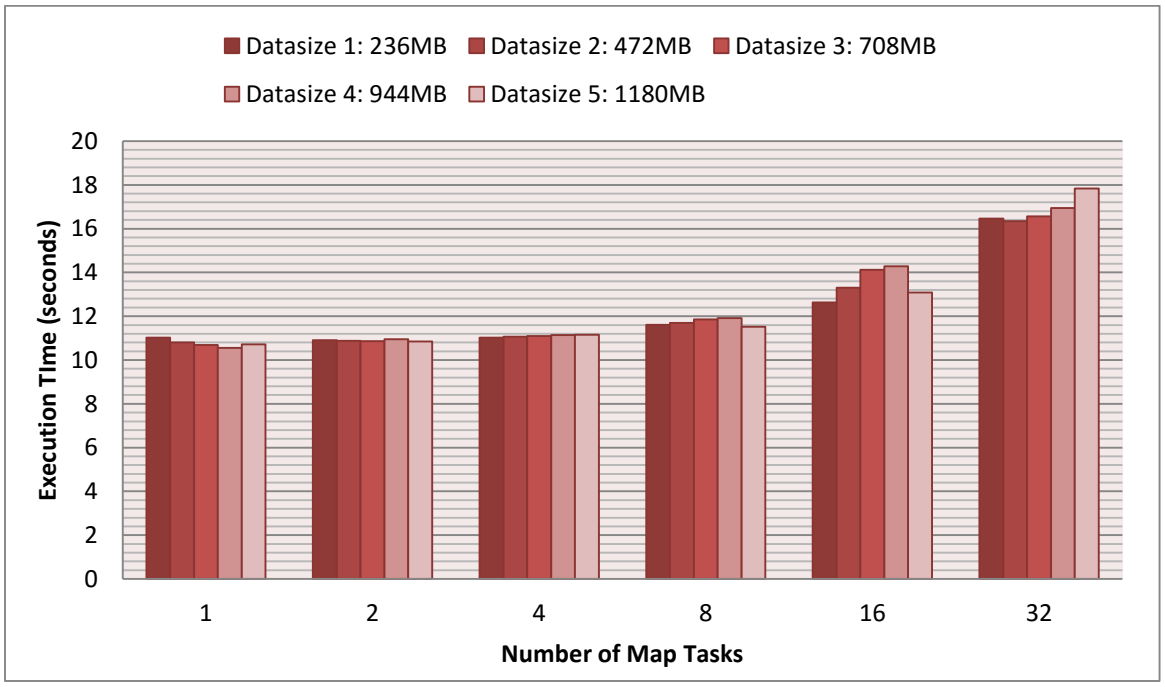


**Figure 4.10:** Effect on the Execution Time of the Reduce Stage When Varying the Number of Map Tasks

Figure 4.10 demonstrates that the execution time of Reduce stage (set to one Reduce task) is not influenced much by the number of Map tasks when the number is small

(1-8); However, the execution time increases when the number of Map tasks becomes larger, changing from 8 to 16 to 32.

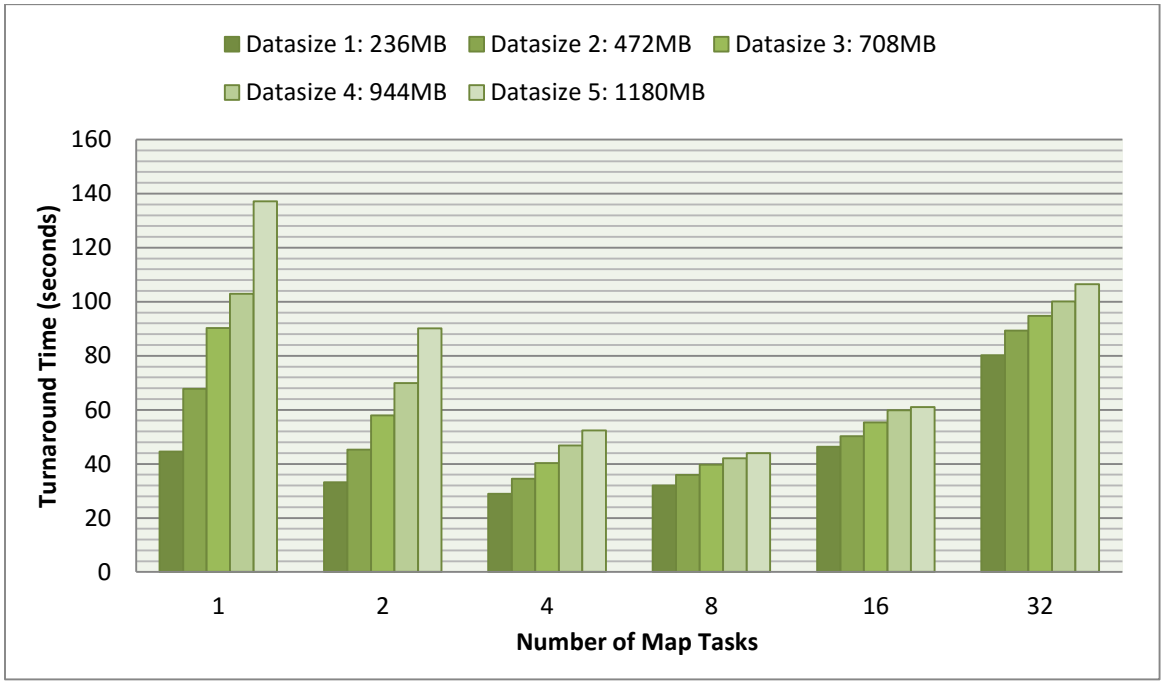


**Figure 4.11:** Effect on Turnaround Time of the Clustering Job When Varying the Number of Map Tasks

The turnaround time includes the following stages of a Hadoop job: job set up, Map stage, Reduce stage (shuffle, sort, Reduce execution), and job clean up. The time needed for job set up (approximately 3.4 seconds), clean up (around 3.5 seconds) and the Reduce stage (as shown in Figure 4.10) does not vary much when changing the input data size or the number of Map tasks. Hence, for our experiment, the Map stage dominates the performance of overall turnaround time. Figure 4.11 illustrates that for our experiments, 4 Map tasks are optimal for datasets with size 236 MB (1 week data) and 472 MB (2 weeks data) and 8 Map tasks are optimal for 708 MB (3 weeks data), 944 MB (4 weeks data) and 1180 MB (5 weeks data).

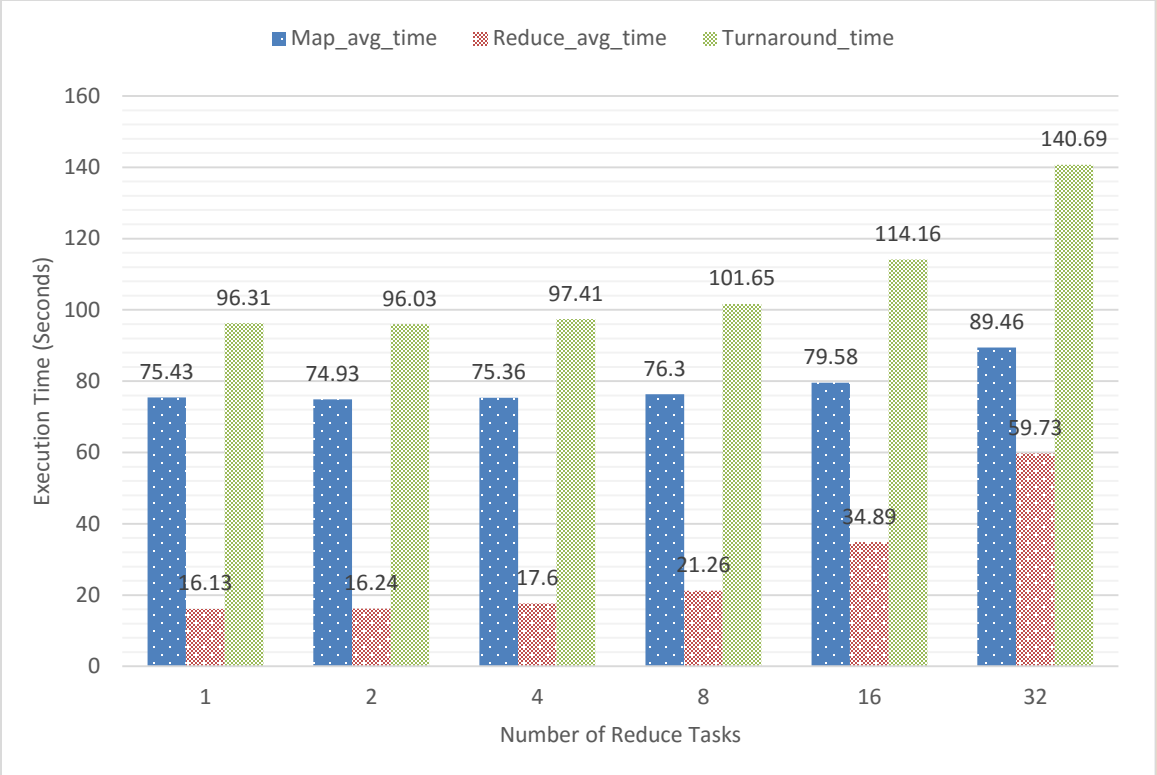


**Figure 4.12:** Effect on the Execution Time When Varying the Number of Reduce Tasks

Figure 4.12 demonstrates the result of the execution time when varying the number of Reduce tasks. The number of Reduce tasks is changed from 1 to 32. The number of Map tasks is fixed at 32, since the number of Map tasks should not be less than the number of Reduce tasks [24]. The default value for clustering parameters (listed at the beginning of Section 4.6) are chosen for these jobs. The average time used for Map stage, Reduce stage, and the average turnaround time for each job are presented in Figure 4.12.

As shown in Figure 4.12, increasing the number of Reduce tasks decreases performance. This is because the time cost for Reduce stage keeps increasing, which in turn also increases the turnaround time. The Reduce stage is composed of shuffling, sorting and Reduce processing. The time needed for each stage of a Hadoop job is listed in Table 4.13.

**Table 4.13:** Execution Time for Each Stage When Varying the Number of Reduce Tasks

| Number of Reduce | setup _time (s) | map_exec _time (s) | shuffle_exec _time (s) | sort_exec _time (s) | reduce_exec _time (s) | cleanup _time (s) |
|---|---|---|---|---|---|---|
| **1** | 3.4 | 77.6 | 13.8 | 0.7 | 1.6 | 3.5 |
| **2** | 3.4 | 77.2 | 14.1 | 0.7 | 1.6 | 3.5 |
| **4** | 3.4 | 78.0 | 15.7 | 0.9 | 1.7 | 3.4 |
| **8** | 3.4 | 79.8 | 19.4 | 1.4 | 1.8 | 3.4 |
| **16** | 3.4 | 85.1 | 33.7 | 2.9 | 2.1 | 3.5 |
| **32** | 3.4 | 98.4 | 61.4 | 4.9 | 3.1 | 3.5 |

As shown in Table 4.13, as the number of Reduce tasks increases, the shuffle time and the sort time grow rapidly. These time consuming operations make increasing the number of Reduce tasks inefficient for executing our experimental datasets. In fact, it is observed that using one Reduce task achieves the best performance when performing experiments with our datasets.

### *4.6.2 Comparison of Performance for Different Number of CPUs*

This section compares the performance of a Hadoop job executing on a slave node with different numbers of cores. The Hadoop cluster used in this section is composed of the *Master Node* (VM1) and one *Slave Node A/B/C/D* (discussed in 4.2). *Slave Node A/B/C/D* contains 1/2/4/8 CPUs, respectively. Each CPU consists of 8 cores (discussed in 4.2). For each experiment, the *Master Node* and one *Slave Node* (A/B/C/D) are used. The execution time of the Map stage the Reduce stage, and the turnaround time are depicted in Figure 4.13.

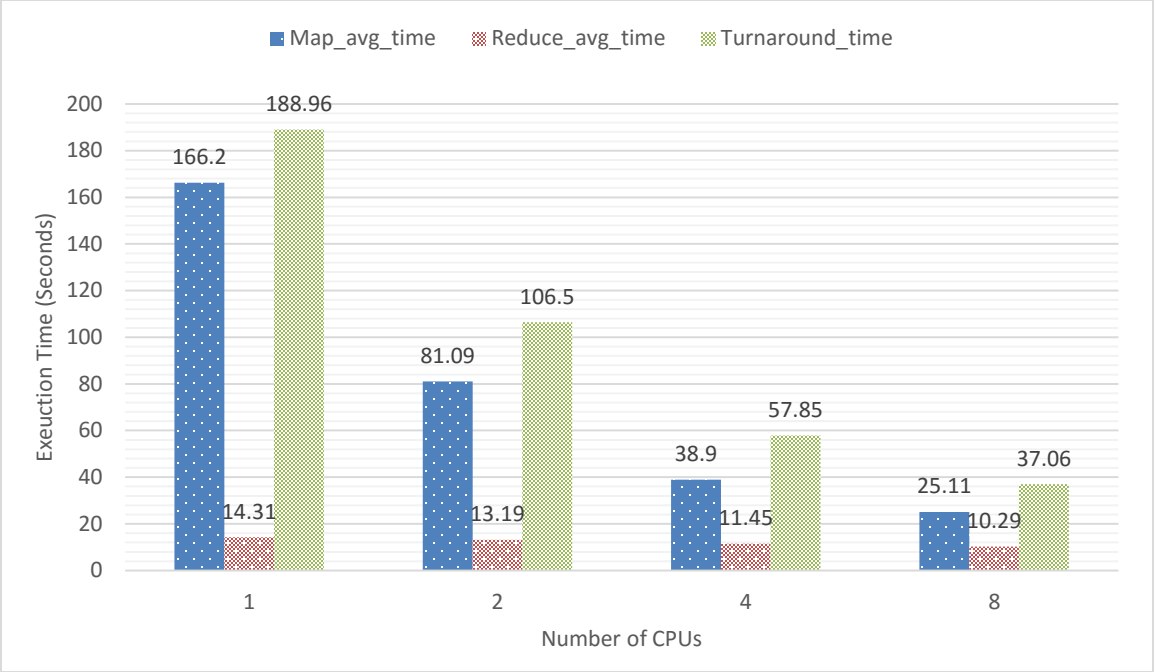


**Figure 4.13:** Execution Time for Different Hadoop Job Stages and the Job Turnaround Time When Varying the Number of CPUs

Figure 4.13 shows that by increasing the number of cores (up to 64 cores in this research), the execution times for the Map stage and Reduce stage decrease due to the increased execution parallelism that the higher number of processing cores provides. Note that the turnaround time is less than the sum of the average execution time required for the Map stage and Reduce stage because the Reduce tasks can start executing after some Map tasks finish.

# Chapter 5: Conclusions

This thesis focused on processing and analyzing mobile network data. One of the main objectives was to find information and traffic data patterns that may provide useful insight for the network operator. We presented a detailed approach to processing and analyzing the datasets collected from real mobile networks. The approach used the Apache Hadoop platform, the clustering algorithms, and other data mining algorithms of the Mahout machine learning library to analyze the datasets. Hadoop is a well-known platform that has been used for efficiently processing large datasets in a parallel and distributed manner. Furthermore, this thesis also investigated how the performance of the Hadoop cluster is affected when executing the Mahout clustering algorithms using various system and workload parameters, including the number of Map tasks and Reduce tasks, the size of the input dataset, and the number of processing cores.

A summary of our experiences and key findings from analyzing the datasets provided by Ericsson, Canada, are outlined as follows:

- The correlation results of Dataset 1 and Dataset 2 (refer to Section 4.1) illustrate the relationship (positive and negative) or lack of relationship between the various features in the datasets. For example, the correlation matrix of Dataset 2 revealed that the number of active users for downlink are independent from the number of active users for uplink. Intuitively, one may think that if there is an increase in the number of active users for downlink, there will also be an increase in the number of active users for uplink. However, there is not obvious relationship between these two features.

- A detailed analysis of the feature *Average RRC (radio resource control) connections/cell* (a critical feature for mobile networks), was performed. This feature specifies the average number of user connected to a base station.
    - The analysis (see Figure 4.6) reveals how the Average RRC Connections/Cell changes throughout a 24-hour period.
    - Using k-means clustering algorithm with K=4, it was observed that 71 out of 5840 base stations have a high value for Average RRC Connections /cell (changing from 45.9 to 194 in a 24-hour period). Conversely, 3230 out of the 5840 base stations were observed to have low values for Average RRC Connections/Cell (changing from 3.5 to 9.21 in the 24-hour period). This results show that, for the specific dataset, most base stations had an acceptable number of connection users. However, there are 71 base stations that have a high value of Average RRC Connections/Cell. The performance of those 71 base stations may be affected, and hence need to be monitored carefully for better resource management.
    - An analysis of the relationship between the Average RRC Connections/Cell and the User Equipment (UE) Uplink/Downlink Throughputs (in Mbps) (see Figure 4.7) showed that as the Average RRC Connections/Cell increases, both the UE Uplink/Downlink Throughputs decrease. The uplink throughput was observed to decrease at a faster rate than the downlink throughput.
- An investigation into the performance and accuracy of applying the PCA [4] technique before clustering for linear dimension reduction (reduction of the number

of features in the dataset for analysis) compared to that without using PCA (see Section 4.4). The results showed that applying PCA before clustering could reduce the execution time (e.g., for Dataset 1 the execution time decreased from 1637s to 1046s), but the accuracy of the clustering solution was still as high as 99.45%.

  - The advantage of using PCA is obvious; however, in some cases it is not practical to apply dimension reduction because the information of the original features needs to be analyzed. For example, in Section 4.5, we analyzed how the Average RRC Connections/Cell changes in a 24-hour period, and therefore dimension reduction cannot be applied in this scenario, because the original information at specific time periods will be lost as a result of linear combination of various dimensions.

The key observations that were made from the experiments conducted to investigate the performance of the Hadoop cluster when executing k-means clustering algorithm using various system and workload parameters are summarized next.

- By increasing the number of Map tasks, the execution time decreased because more jobs are running in parallel. However, this trend may not continue if the number of Map tasks is too high (i.e., greater than 8 for our experiments) due to the overhead of generating Map tasks. By conducting experiments with datasets of various sizes, it is observed that using 4 Map tasks achieves the lowest execution time when the size of the dataset is 236 MB (1-week data) or 472 MB (2-week data). Furthermore, using 8 Map tasks is observed to achieve the lowest execution time when the size of the dataset is 708 MB (3-week data), 944 MB (4-week data), or 1180 MB (5-

week data). The number of features also can affect the performance trend. Conducting a pilot experiment is recommended for a specific environment.

- Increasing the number of Reduce tasks does not improve the performance due to the substantial overhead of shuffling and sorting (i.e., copying data to and from nodes where different Reduce tasks are executing).
- By increasing the number of processing cores based on our experimental environment, it is observed that the execution time for the Map stage, Reduce stage, and the overall job turnaround time decreases. This is due to the increased execution parallelism that the higher number of processing cores provides (i.e., more Map and Reduce tasks can be executed in parallel).

## 5.1 Future Work

This section discusses possible directions for future work. This thesis focuses on comparing the performance of applying two of the well-known clustering algorithms: k-means and fuzzy k-means to analyze the historical system data collected by base stations. Using other machine learning algorithms such as classification algorithms in the Mahout library to analyze the datasets forms an interesting direction for future research. Furthermore, developing a tool that can automate the data analysis approach presented in this thesis can be investigated. The tool can automate the collecting, processing/analyzing, and displaying the analysis results.

## References


[1] Ericsson, "Ericsson Mobility Report: On the Pulse of the Networked Society," June 2015.

[2] Apache Hadoop, "Welcome to Apache Hadoop," [Online]. Available at: https://hadoop.apache.org [Accessed June 25 2015].

[3] Hortonworks, "Apache Mahout," [Online]. Available at: http://hortonworks.com/hadoop/mahout/ [Accessed June 25 2015].

[4] I. K. Fodor, "A Survey of Dimension Reduction Techniques," Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[5] Apache Nutch, "Apache Nutch," [Online]. Available at: http://nutch.apache.org/ [Accessed June 25 2015].

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *ACM SIGOPS Operating Systems Review*, 37(5), 2003, pp. 29–43.

[7] J. Dean, S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters," *Proc. of the 6th Symposium on Operating System Design and Implementation (OSDI04)*, 2004, pp. 137–150.

[8] T. White, *Hadoop: The Definitive Guide*, Yahoo Press, 2010.

[9] Apache Hadoop, "HDFS Architecture Guide," [Online]. Available at: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [Accessed June 25 2015].

[10] P-N. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining*, Addison-Wesley, 2006.

[11] Owen S., Anil R., Dunning T. and Friedman E., *Mahout In Action*, 2012. Manning Publications Co. ISBN 978-1-9351-8268-9.

[12] J. Yang, H. He, and Y. Qiao, "Network Traffic Analysis based on Hadoop," *Proc. of International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE),* May 2014, pp.1-5.

[13] D. Tang and M. Baker, "Analysis of a Local-Area Wireless Network," *Proc. of the 6th annual international conference on Mobile computing and networking,* August 2000, pp.1-10.

[14] F. Demers and M. St-Hilaire, "Radiometric Identification of LTE Transmitters," *IEEE Global Communications Conference (Globecom 2013)*, Atlanta, GA, USA, December 2013, pp. 4116-4121.

[15] R.M. Esteves and C. Rong, “Using Mahout for Clustering Wikipedia's Latest Articles: A Comparison between K-means and Fuzzy C-means in the Cloud,” *Proc. of International Conference on Cloud Computing Technology and Science (CloudCom),* Nov. 29 - Dec. 1 2011, pp. 565-569.

[16] R.M. Esteves, R. Pais, and C. Rong, “K-means Clustering in the Cloud -- A Mahout Test,” *Proc. of IEEE Workshops of Advanced Information Networking and Applications (WAINA),* March 2011, pp.514-519.

[17] M. Kuhn, “Variable selection using the caret package,” [Online]. Available: http://cran.r-project.org/web/packages/caret/caret.pdf [Accessed June 25 2015].

[18] R, “The comprehensive R archive Network” [Online]. Available at: http://www.r-project.org/ [Accessed June 25 2015].

[19] R, “The R Stats Package,” [Online]. Available at: http://stat.ethz.ch/R-manual/R-patched/library/stats/html/00Index.html [Accessed June 25 2015].

[20] L.Egghe and L.Leydesdorff, “The relation between Pearson’s Correlation Coefficient r and Salton’s cosine measure,” *Journal of the American Society for Information Science and Technology,* 60(5), 2009, pp. 1027-1036.

[21] R.A. Fisher, “The Use of Multiple Measurements in Axonomic Problems,” *Annals of Eugenics*, vol. 7, 1936, pp. 179-188.

[22] N. Kerdprasop, “Multiple principal component analyses and projective clustering,” *Proc. of IEEE International Workshop on Database and Expert Systems Applications. (DEXA’05)*, 2005.

[23] Netflix, “Internet Connection Speed Recommendations” [Online]. Available at: https://help.netflix.com/en/node/306 [Accessed July 15 2015].

[24] Stanford University InfoLab, “MapReduce and the New Software Stack” [Online]. Available at: http://infolab.stanford.edu/~ullman/mmds/ch2.pdf [Accessed July 20 2015].