A Cloud Based Automated Anomaly Detection Framework

by

PRATHIBHA DATTA KUMAR

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2014

To my family.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt thanks to Mr. David Levine, my supervisor who inspired and motivated me to do this thesis. This thesis would not have been possible without his encouragement. I extend my sincere gratitude to Mr. David Levine for his invaluable support and supervision.

I would like to extend my sincere thanks to Dr. Giacomo Ghidini, my mentor. He has been very supportive and has guided me throughout my thesis. I have learned a lot from Dr. Ghidini. My heartfelt thanks to him. I would like to thank another mentor, Mr. Steve Emmons for his precious support and guidance. His invaluable comments have helped me greatly. I would also like to thank Mr. Gustavo Puerto for his generous inputs and thoughts.

I am grateful to Numerex Corporation, Dallas, TX and Dr. Jeff Smith, the CTO for giving me the opportunity and support to work on this project.

My heartfelt thanks to my committee members Dr. Matthew Wright, and Prof. Ramez Elmasri, and the computer science department, UTA for their invaluable co-operation, and support.

My heartfelt thanks and appreciation to my husband and family for their continuous encouragement and inspiration. Finally, I would like to thank all my friends who helped me through this journey.

October 16, 2014

ABSTRACT

A Cloud Based Automated Anomaly Detection Framework

PRATHIBHA DATTA KUMAR, MS

The University of Texas at Arlington, 2014

Supervising Professor: David Levine

A machine-to-machine (M2M) communications network hosts millions of heterogeneous devices such as for vehicle tracking, medical services, and home automation and security services. These devices exchange thousands of messages over cellular networks. These messages are Signaling System No. 7 (SS7) messages of various types like authentication, mobility management, and many more, resulting in tera bytes of SS7 signaling traffic data over a period of days. The data generated is diverse, depending on several factors like device activity, hardware manufacturers, and radio / tower interaction. This inherent diversity makes anomaly detection in a M2M network challenging. With millions of messages to analyze, high computation machines are necessary.

In this thesis, an automated data mining framework on the cloud, to detect anomalous devices in the traffic data is presented. Unsupervised learning such as cluster analysis is a useful tool here given the lack of static behavioral patterns of devices and numerous ways in which they can fail. Datasets extracted from a leading M2M service provider's network featuring millions of devices are analyzed. The case

studies illustrate the anomaly patterns found. One case study identified 27% of devices with an unusual behavior in a dataset of 23k devices. A second case study spotted 0.07% of devices with anomalous behavior in a dataset with 350k devices. This places the spotlight on a small subset of devices for further investigation. In order to achieve the goal of finding anomalous devices, the clusters derived from cluster analysis are labeled based on a few assumptions which are part of unsupervised anomaly detection and cluster analysis techniques. Then, k-nearest neighbors (k-NN) binary classification is applied to evaluate the labelling. This categorizes each device as either "anomalous" or "normal" with quantitative results like accuracy. To generate actionable information, the devices identified are analyzed in light of domain expertise, past events and auxiliary information like the type of the device and its purpose among others.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction and background

Machine-to-Machine (M2M) refers to machines, typically small sensors exchanging or relaying data to other machines over a cellular network using the simple SS7 protocol. M2M has a wide range of applications and is an integral part of the broader sector, Internet of Things (IoT). It offers connectivity and communication capabilities to the things in Internet of Things (IoT). M2M is a burgeoning sector; by 2020, the M2M and IoT world is predicted to interconnect about 50 billion devices [1].

M2M includes devices (such as a sensor) which capture events (such as temperature) information about which is relayed via a network (cellular) to applications (software program) that translate and represent the event information in a meaningful format [2]. Fig. 1.1 demonstrates the vital elements that make up M2M - Devices, Network and Applications (DNA).

M2M network hosts a plethora of devices like remote monitoring devices for various purposes such as patient monitoring and health care, vehicle tracking, inventory tracking, supply chain tracking, agriculture, and home security among several other emerging domains. These devices transfer messages in form of Short Message Service (SMS) which are routed using SS7 [3] signaling protocol over the roaming cellular network owned by the network partners. M2M provider owns the devices deployed in the network and has access to the data or messages those devices relay. The data serves as the single point of access to a wealth of information about the devices such as status, types of messages relayed, behaviors, events happening in the

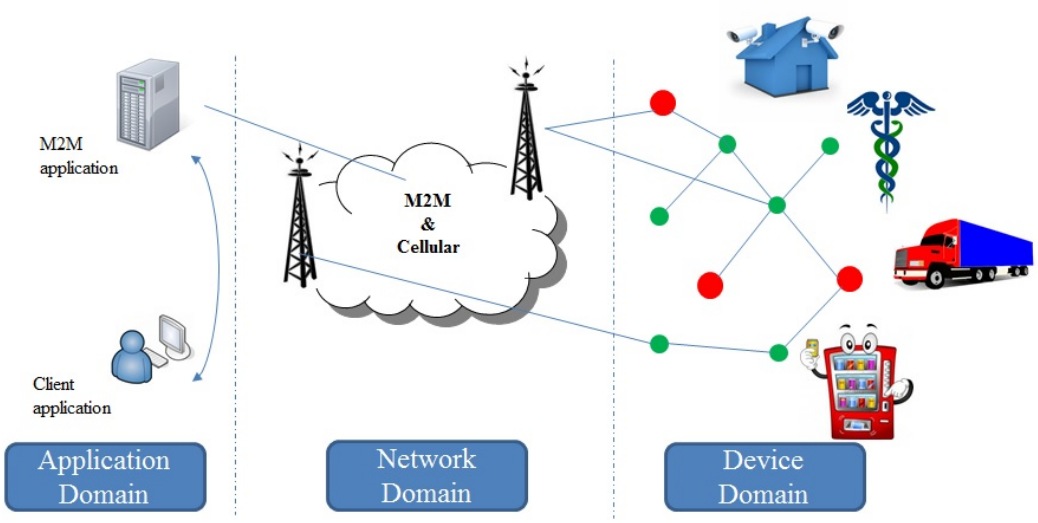


Figure 1.1: M2M with Devices, Network and Applications

network and hidden communication patterns. When all the communication of a single device is considered, we are looking at high dimensional data, concealing diverse and unique communication patterns. Let us consider an example. A pattern could demonstrate the authentication requests and responses of a particular type of device like a vehicle tracking device. It is seen that these devices usually have a high volume of authentication type messages since they are mobile and thus hop from one tower to another and from one network to another. The devices serve many different purposes. This diversity can be identified in form of patterns. These patterns provide insightful knowledge about the devices, their activities and environment.

An automated proactive framework to identify anomalous devices will significantly benefit the M2M network and services. A framework which can aid to get a better understanding of the devices and their activities in the cellular network will be very useful. This is a stepping stone to detecting the problematic ones. The communication patterns of problematic devices can be seen as anomalies. An anomaly can be anything that represents an inconsistency. In our case, an anomalous device is one

which stands out by exhibiting inconsistent behavior when compared to remaining devices in the dataset.

Identifying anomalous devices among millions or the forecasted billions is challenging. There can be numerous causes for an anomaly. Malfunctioning devices is one of them. There is no standard definition for a malfunction because a malfunction in one context can be treated as normal behavior in another context. For example, a vehicle tracking device's normal behavior is to produce data representing its changing locations since it is mobile, whereas in the context of an immobile home security device, data indicating mobility is considered anomalous. With the absence of ground truth and numerous (unknown) inconspicuous ways in which the devices can fail, it is hard to spot the malfunctioning ones.

In this thesis, an automated machine learning based framework on the cloud to identify anomalous devices in the traffic data is presented. A proactive approach is taken rather than a reactive one, where a problematic device is brought to attention before a customer reports it. Unsupervised learning technique called cluster analysis is leveraged to identify clusters of devices with similar behaviors. Then, the clusters are validated using well known techniques like silhouette co-efficient. The clusters are then labelled as either normal or anomalous under a few assumptions which are part of unsupervised anomaly detection and cluster analysis techniques. This labelling is evaluated by applying k-NN, a supervised learning technique. Further analysis of categorized devices is done to identify malfunctioning ones and generate actionable information. Cloud computing resources and technologies are leveraged to make the framework cost-effectively scalable to the increasing amounts of data being generated. The continuous evolution of the M2M / IoT world and their ubiquitous use necessitates a dependable M2M network with proactive fault detection and early corrective actions.

1.2 Motivation behind the thesis

Detecting anomalies in a network hosting millions of devices is an unreasonable task for the human eye alone. Patterns cannot be identified by just looking at the numbers and figures that make up the communication data of all the devices on the network. This is true especially with heterogeneous devices and diurnal datasets. Instead, we need to leverage machines that can learn from the data (old and new) and identify communication patterns that might indicate anomalies.

Identification of anomaly patterns and anomalous devices can further bring to light malfunctioning devices which wastefully utilize network resources, and consume time and money. Such malfunctioning devices can be corrected early and brought back to track with early detection techniques. Anomaly patterns can also disclose important activities in the network, the various causes and consequences of certain specific events which further help to take precautions when such an activity repeats in the future. A proactive, automated approach to identify anomalies should definitely replace the reactive approach where an action is taken only after a customer report and a tedious manual inspection.

1.3 Goals of the thesis

The broad goal of this thesis is to implement a proactive automated framework to detect anomalies in the SS7 traffic data and categorize devices as either "anomalous" or "normal" with quantifiable results by leveraging machine learning techniques. By leveraging machine learning techniques this thesis aims to handle the irregular nature of the communication data. The initial time of this thesis was spent in implementing a proof of concept which is detailed in Section 5.3

This research facilitates near-real time insights into the raw data that was incomprehensible at the start of the analysis. The knowledge learned from the framework can help in intelligent decision-making process, fault diagnosis and device monitoring.

1.4 Organization of the thesis

This thesis starts with an introduction and background, motivation and goals of the thesis. Chapter 2 gives an overview of related work. Chapter 3 defines the problem statement. Chapter 4 details the data sets which is the backbone of this thesis. Chapter 5 provides a detailed description of the preliminary analyses, proof of concept and observations. Chapter 6 gives a description of the framework implemented and the cloud implementation of the framework follows in Chapter 7. Chapter 8 details the experiments, analyses and case studies. This thesis is then concluded in Chapter 9 with future work outlined in Chapter 10.

CHAPTER 2

RELATED WORK

An anomaly in a dataset is defined as an observation that appears to be inconsistent with the remainder of the dataset [4]. This inconsistency can have numerous causes such as a faulty sensor, an intrusion among others. In this thesis, the focus is on anomaly detection to identify faulty sensor devices. Anomaly detection has been researched and applied in diverse research areas and application domains such as intrusion detection, pattern recognition in cellular M2M networks, traditional wired networks, and peer-to-peer environments. Previous work related to our goals and processes can be found in the literature. Rajasegarar et al. present a distributed anomaly detection scheme based on clustering [5]. They consider sensors that are deployed in a homogeneous environment where they are time synchronized and the measurements taken have the same unknown distribution. Bandyopadhyay et al. present a technique for clustering homogeneously distributed data in a peer-to-peer environment based on the principles of k-means algorithm [6] [7]. Wang et al. use self-organizing map (SOM) neural network and k-means clustering [8]. Yu et al. present a rule based machine learning framework in wireless sensor networks using a training set [9].

An unsupervised anomaly detection scheme was developed by Garette in the context of intrusion detection [10]. Although the context covered in our research is very different, the techniques used are similar and the goals seem to converge; The goal is to identify anomalies with little prior knowledge about the dataset. In the scheme presented, the author demonstrates the application of many clustering algorithms including k-means, to identify normal and anomalous traffic and also sug-

gests improvements to the algorithms. A strategy is presented where a percentage of the clusters are labeled as "normal" and the author also uses width of clusters for labelling. Here the author supplies values for percentage and width inputs.

The limited prior knowledge about the datasets and the inherent diversity do not allow the generation of a training set directly from our raw dataset. This makes our task of classifying the devices into "normal" or "anomalous" challenging as there is no reference dataset which can help validate our findings. In this thesis, our clusters are labelled without being dependent on any external input. The labelling is based on basic assumptions that are part of clustering and unsupervised anomaly detection techniques and how these techniques work. These assumptions are similar to the ones made in the research work by Garette in [10]. Then, we apply k-NN, a supervised learning algorithm to label the rest of the dataset. Our framework detects anomalies in heterogeneous datasets as opposed to homogeneous datasets which are considered by some authors mentioned in this section. It is important to avoid making any assumptions about data distribution, especially with diurnal datasets. Our framework detects anomalous devices without making any assumptions about the data, devices, or its environments.

CHAPTER 3

PROBLEM STATEMENT

The problem this thesis addresses is the detection of anomalous devices in a diverse M2M network hosting over 2M devices [11]. These devices contribute to approximately 60M daily SS7 transactions. Each transaction consists of two messages: request and response. Each transaction has something to say about the device, its environment and events taking place in the network. The volume and variety of data make anomaly detection a challenging task. The devices function on a variety of hardware tuned to various vendors' businesses. Such factors add to the diversity.

On the brighter side, such rich data can be analyzed to reveal insightful information about devices and network. This research attempts to identify anomalous devices by analyzing the SS7 message volumes relayed by the devices. Analyzing the devices' SS7 volumes in a perspective brings to light many behavioral patterns, both anomalous and normal patterns which provide a good ground to analyze and enhance our knowledge about the data. Let us consider an example. In the event of a maintenance shutdown carried out by a cellular network partner, the devices affected by this event behave differently than unaffected devices. This behavior is recorded in the SS7 messages they exchange.

Continuous analyses of datasets require an automated analytical framework. Cloud resources are leveraged for distributed processing and storage of the large volume of data. Like many real-life situations, we face the absence of ground truth. The diversity introduced by many factors makes it difficult to characterize good and bad behavior. These definitions can also vary depending on the context. Therefore, an

unsupervised learning technique called cluster analysis is applied to discover hidden behavioral patterns in our unlabeled data. Clustering technique groups devices in such a way that devices in the same cluster are more similar to each other in some way than the devices in other clusters. Anomalies are detected in unlabeled data under the assumption that majority of the devices behave normally and the ones that do not fit most devices' behavior is considered an outlier and hence an anomaly.

The analysis of SS7 traffic distribution and message types can lead towards anomalous devices which might probably be misbehaving or faulty. A proactive identification of such anomalous devices out of hundreds of thousands of devices is analogous to picking a needle in a haystack.

CHAPTER 4

DATA

As part of this thesis, the signaling traffic volumes captured at the Home Location Register (HLR) and Message Switching Center (MSC) of a 2G cellular network belonging to a leading M2M service provider is analyzed. The M2M communications network features well over 2M devices that include sensors to capture and quantify events happening in their surroundings. Fig. 4.1 illustrates the application domain and extraction of data. The MSC in Fig. 4.1 is responsible for routing SMSs. The Base Station Subsystem (BSS) is responsible for handling traffic and signaling between devices and MSC. The HLR is a central database that contains details of each device subscriber that is authorized to use the GSM network. The Visitor Location Register (VLR) is a database of the subscribers who have roamed into the jurisdiction of the MSC which the VLR serves [12].

Each device in the M2M network belongs to a single account and exhibits a specific state. An account indicates the purpose or type of the device (e.g.: Vehicle Tracking) and state indicates the status of the device; 13 different states exist such as active, and inactive.

Table 4.1 provides examples of raw transactions. The traffic data extracted over a period of one week is analyzed. We consider a weekly time granularity because most machine learning algorithms work well on large sized datasets, and one week is a reasonable time period to differentiate between usual and unusual events in the network. Each device is associated with an International Mobile Subscriber Identity (IMSI) and a Mobile Subscriber Integrated Services Digital Network Number (MSISDN) [13]

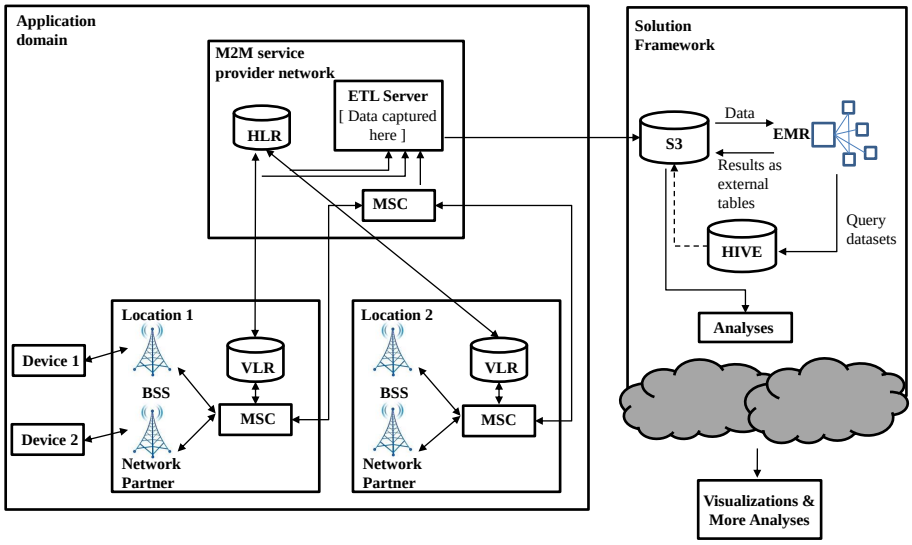


Figure 4.1: Application domain and cloud framework architecture

which uniquely identify a subscription in the network. Each SS7 message comprises of the following three parameters: TCAP Type [14], GSM MAP Component, and GSM MAP value [15]. These parameters define message types in SS7. Our analysis is focused on 14 important message types and the sum of other message types as 'SumofOtherMessageTypes' is also considered so as to not ignore any communication made by the device. The 14 message types along with corresponding parameter values are listed in Table 4.2.

The raw dataset is preprocessed such that the messages of each device are represented in one row with 15 columns each representing the message types detailed previously. Table 4.3 details a few example rows of the preprocessed datasets. The datasets have around 20 columns and a few hundred thousand rows in each account.

Based on the observations from experiments conducted on the preprocessed datasets, it was observed that a few message types like authentication types are dominant. While this observation tells us something worthwhile about the devices,

we do not want a few message types or columns to bias our results. Therefore, the datasets are normalized by row and by column.

4.1 Normalize by row

The preprocessed datasets are normalized by row with respect to the sum of the message volumes for each device such that the new values in each row fit add up to 1. Therefore, all the new values lie in range [0,1]. The message volumes in 15 columns are totaled to represent the sum of volumes for each row. The new value is computed as a ratio of original value which represents the message volume for each message type and sum of each row.

$$NewValue = \frac{RawVolume}{Sum}$$

where NewValue is the normalized value,
RawVolume is the message volume for a single message type,
Sum is the sum of volumes of all message types for each device

Recollecting that each row represents communications of a single device, the new values represent the contribution of each type of message (column) to the overall communication.

4.2 Normalize by column

Preprocessed datasets are normalized by column with respect to the minimum and maximum values in each column. Each column is transformed such that the new values fit in the range [0,1] with new minimum being zero. New value is computed as a ratio of (old value - minimum) and range. This eliminates the unfair bias introduced

by dominating message types.

$$NewValue = \frac{RawVolume - Min\_vol\_column}{Max\_vol\_column - Min\_vol\_column} \times Scale + Translation$$

where New value is the normalized value,

Raw volume is the original message volume,

Min_vol_column is the minimum raw (original) volume in each column,

Max_vol_column is the maximum raw (original) volume In each column,

Scale is the factor for scaling the output range = 1 for range [0,1],

Translation is the translation of the output range = 0 for range [0,1]

With normalization, the influence of individual message types on the computation of Euclidean distance is evened out. Euclidean distance is the main distance function used in the framework. The results of cluster analysis on normalized datasets are detailed in chapter 8.

Table 4.1: Description of raw datasets

Account Id	State Id	IMSI	MSISDN
Account1	3	Imsi1	Msisdn1
Account1	9	Imsi2	Msisdn2
Account2	5	Imsi3	Msisdn3

TCAP Type	GSM MAP Component	GSM MAP Value	MessageCount
1	1	56	2
3	2	56	3
3	3	1	1

Table 4.2: Important SS7 message types

TCAP Type	GSM MAP Component	GSM MAP Value	Meaning
1	1	56	InvokeSendAuthenticationInfo
3	2	56	ReturnSendAuthenticationInfo
3	3	1	ReturnErrorUnknownSubscriber
1	1	2	InvokeUpdateLocation
3	2	2	ReturnUpdateLocation
1	1	3	InvokeCancelLocation
3	2	3	ReturnCancelLocation
2	1	7	InvokeInsertSubscriberData
2	2	7	ReturnInsertSubscriberData
1	1	23	InvokeUpdateGprsLocation
3	2	23	ReturnUpdateGprsLocation
1	1	67	InvokePurgeMs
3	2	67	ReturnPurgeMs
4	0	0	TcapAbort

Table 4.3: Description of preprocessed datasets

Generic Heading	IMSI	MSISDN	Message Type 1	Message Type 2	Message Type 3	...	Message Type 14
Detailed Heading	IMSI	MSISDN	Invoke Send AuthInfo	Return Send AuthInfo	ReturnError... Unknown Subscriber		Return PurgeMs
	Imsi1	Msisdn1	2	2	0		0
	Imsi4	Msisdn4	3	3	0		2
	Imsi5	Msisdn5	1	1	0		1

Generic Heading	SumOfOther MessageTypes	SumTotal	Year	Week	AccountId	StateId
Detailed Heading	SumOfOther MessageTypes	SumTotal	Year	Week	AccountId	StateId
	0	12	2013	43	Account1	3
	0	22	2013	43	Account1	3
	0	18	2013	43	Account1	3

CHAPTER 5

PRELIMINARY ANALYSES

This section is dedicated to the stepping stones of this thesis. The first part of the thesis involved implementing a proof of concept to detect anomaly patterns in the datasets. The proof of concept was executed locally. Some analyses were performed before the ideation of the proof of concept.

5.1 Cleaning datasets

The datasets were inspected in order to get rid of noisy data. The datasets were cleaned by getting rid of rows containing null values for IMSI and MSISDN. This step also ensured that the datasets do not contain any missing values.

5.2 Preliminary analyses

Preliminary analyses were performed on the data described in chapter 4. The frequency distribution of message volumes was visualized. Fig. 5.2 illustrates the frequency distribution of message volumes belonging to Vehicle Tracking account. It was inferred that the message volumes represent a skewed distribution. At this point, it made sense to apply unsupervised machine learning algorithms on the datasets rather than statistical procedures like Principal Component Analysis (PCA) [16] which guarantees independent principal components only when the dataset is jointly normally (gaussian) distributed.

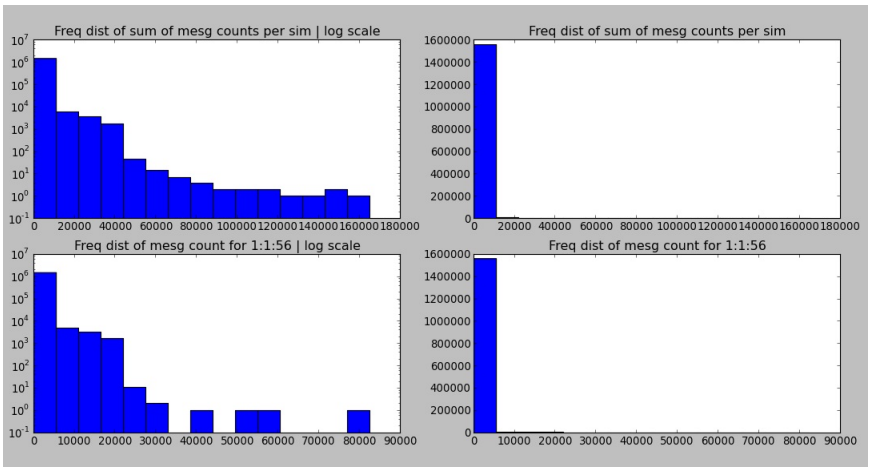


Figure 5.1: Frequency distribution of SS7 message volumes for one week

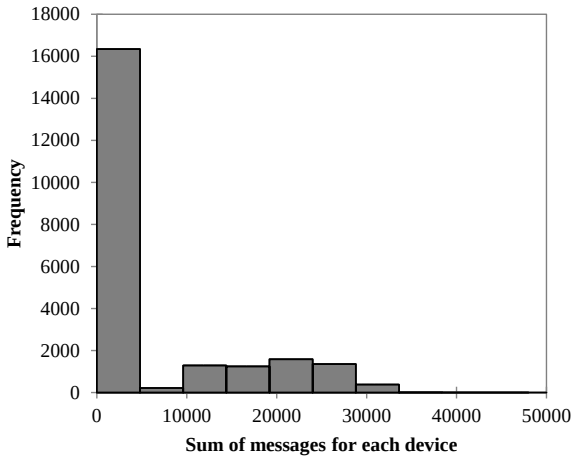


Figure 5.2: Frequency distribution of vehicle tracking dataset for one week

5.3 Proof of concept

The first step towards the implementation of this thesis was to implement a proof of concept (POC). This helped to get an understanding of the datasets and what data mining / analytics techniques to apply and how to apply. The proof of concept was implemented locally on a quad core 64-bit laptop with 8GB RAM and a virtual machine(Ubuntu) guest with dual core and 3GM RAM. Fig. 5.3 illustrates the POC workflow implemented on the local machine.

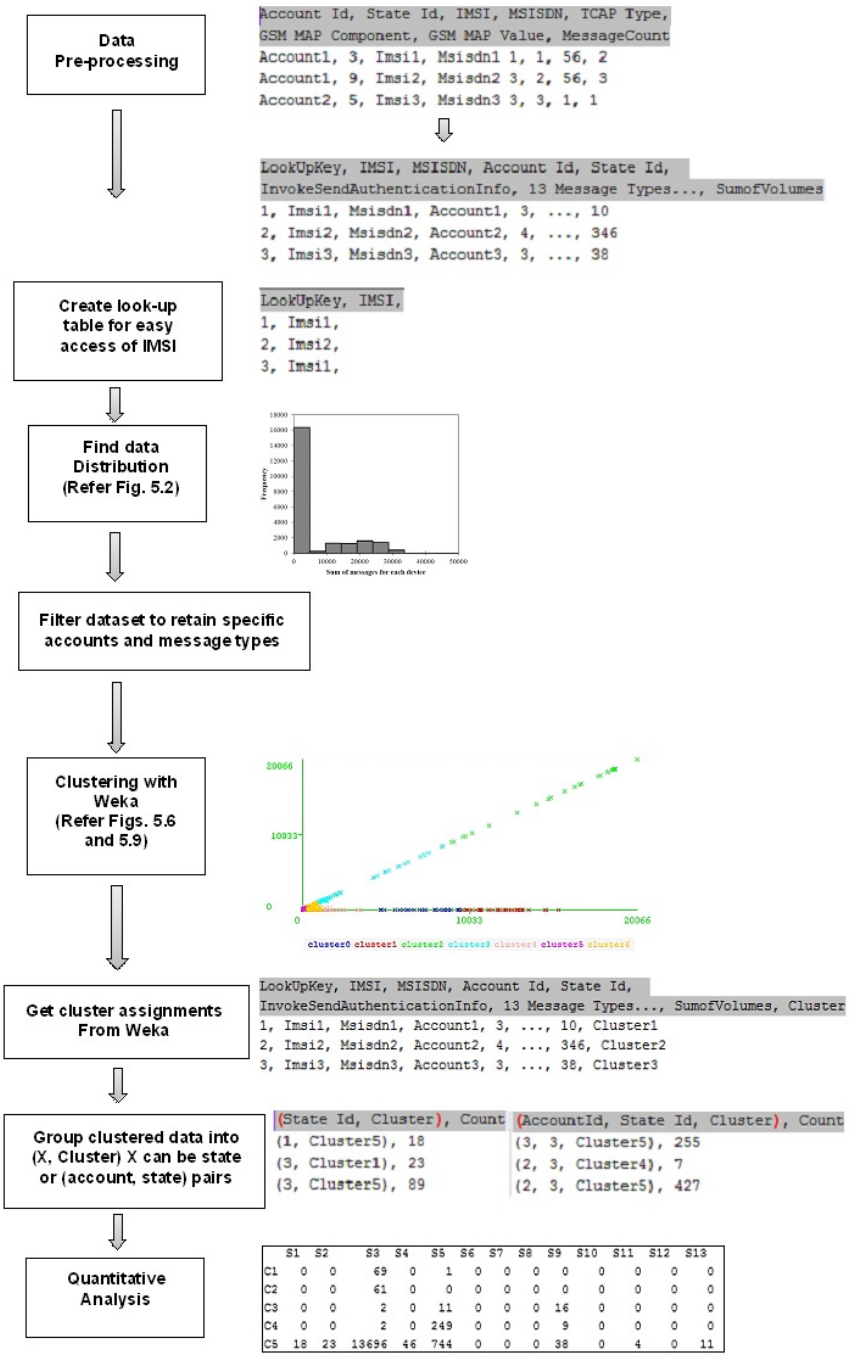


Figure 5.3: Local implementation of POC Workflow

The workflow starts with data preprocessing. The original SS7 message volumes consist of one message type and the respective message volume. This raw representation is illustrated in Table 4.1. The raw dataset is preprocessed to obtain the representation illustrated in Table 4.3. The preprocessing is achieved using Hadoop streaming with mapper and reducer scripts [17] implemented in Python [18]. Python version 2.7 is used. As detailed in chapter 4, each device in the dataset belongs to one among 13 different states. The complete preprocessed dataset for week 43 in the year 2013 consists of 1.5M rows, representing the communication by each IMSI in one row. Initially, a random sample of 15k rows were considered as one single dataset. After some experiments, the complete preprocessed dataset was analyzed. After preprocessing, the data distribution is analyzed and is found to be skewed as detailed in section 5.2. The preprocessed dataset is then filtered to retain only the 14 message types listed in 4.2, specific accounts, and states. At this stage, the focus is laid on states like 'good standing' as one of the goals is to find devices which are supposed to behave well but do not, i.e. misbehave / faulty. Devices with status like deactivated are filtered. Once the dataset is ready, unsupervised learning (k-means clustering) is applied.

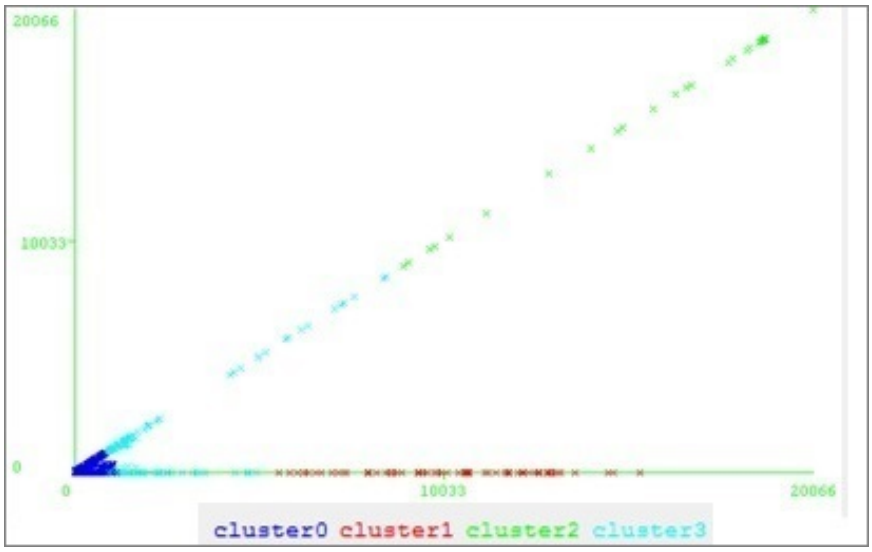
Basic input parameters for the algorithm include seed value and 'k' which represents the number of clusters. k-means is an iterative algorithm where the initial centers are chosen using the seed value and each observation in the dataset is assigned to the closest cluster. The centers are then recalculated as the average of all observations in a cluster. The assignment step is carried out iteratively until there are no more changes or maximum number of iterations is reached. The algorithm partitions the dataset containing n observations into 'k' clusters such that observations with similarities of some kind belong to the same cluster. Each observation belongs to the cluster with the nearest mean. Euclidean distance is the distance function used. In

our case, observations mean devices. Weka, an open source machine learning tool [19] was used to apply k-means clustering on the preprocessed dataset. The 'k' value for the data matrix is determined by manual inspection. The experiments for deciding the optimal 'k' are detailed in section 5.4.

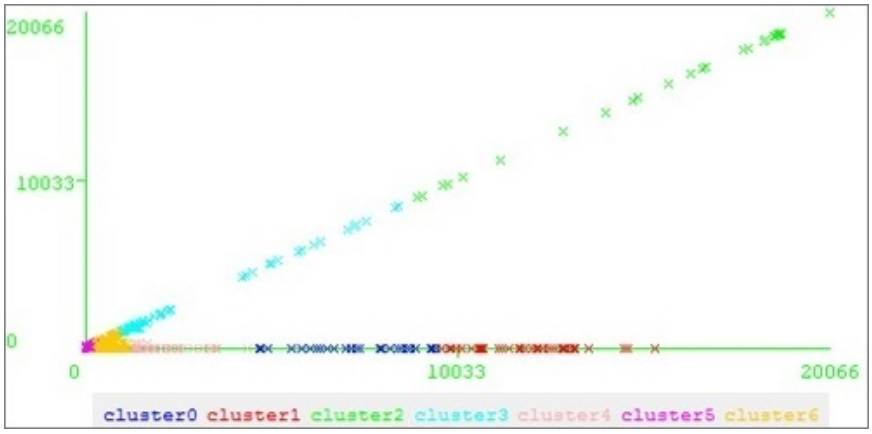
Once the value for 'k' is decided and k-means clustering is applied, cluster assignments are obtained. Each device / SIM is represented as a row along with the id of the cluster it is assigned to. Fig. 5.3 includes a sample screenshot of cluster assignments. This resultant dataset is then grouped by either (account, state) or (state) paired with the cluster id column. The grouping is done to further quantify the distribution of state and account information in the dataset that is being analyzed. This step is accomplished using Apache Pig, a high level language platform for analyzing datasets as MapReduce programs [20]. With the results of grouping, a matrix is created with cluster numbers represented by rows and 13 different states represented as columns. Fig. 5.3 demonstrates the matrix as the last step in the workflow. Fig. 5.6 displays the matrix. This matrix gives an understanding of the clustering of devices and the distribution of the device states among the clusters. The inferences derived are detailed in section 5.5

5.4 Local experiments and results

In this section, the experiments conducted as part of the POC are described. Experiments to find the optimal 'k' for k-means algorithm were performed. Experiments to understand the influence of the seed value on k-means clustering were carried out to reach a conclusion that the seed value did not change the results of clustering significantly. Experiments to know the most influencing message types were conducted by ignoring a set and considering another set of message types in the clustering step of the POC. Also, experiments on the machine learning tool to be used



a: k-means with k=4

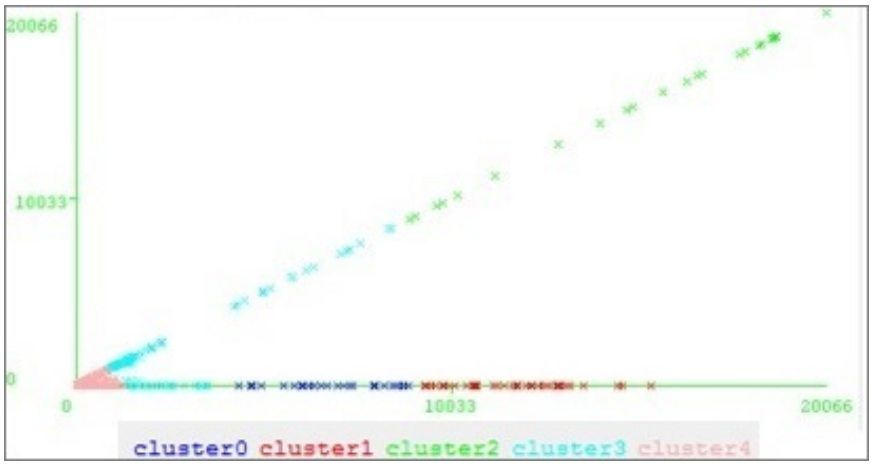


b: k-means with k=7

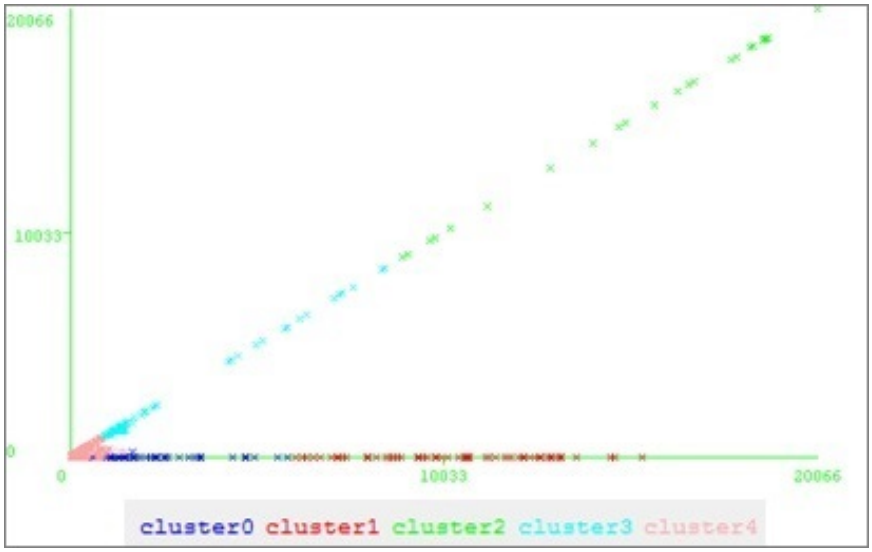
Figure 5.4: Local experiments for k value (1)

were carried out. After experiencing tools like R, KST and Weka, the POC was carried out in Weka attributing to its ease of use, intuitive user interface, visualizations and community support.

Finding the optimal 'k' was one of the challenges faced when applying k-means clustering. Manual inspection was used to understand the influence of 'k' on the datasets. A range of values for 'k' was applied with k-means algorithm using the machine learning tool Weka. Starting with a value as small as 2, increasing values for 'k'



a: k-means with k=5



b: k-means with k=5 and GPRS Location message types

Figure 5.5: Local experiments for k value (2)

were experimented. Fig. 5.4 and Fig. 5.5 demonstrate the visualizations of clustered devices / SIMs resulting from the application of k-means algorithm with different 'k' values. All the figures are scatter plots representing 'InvokeSendAuthentication-Info' message type on the x-axis and 'ReturnErrorUnknownSubscriber' on the y-axis. These message types are listed in the Table 4.2. Each point denoted by an 'x' mark in the scatter plot's XY space represents a single device / SIM in the dataset and the

color represents the cluster that the SIM belongs to. A good clustering would be one that has minimum overlapping of clusters with each cluster representing a distinct set of SIMs.

Fig. 5.4a demonstrates the clustering with 'k' equal to 4. In this figure, it can be observed that the SIMs colored in cyan which belong to cluster 3 are spread along the x-axis and also in between x and y axes. By a manual inspection of these points, it was identified that the SIMs along the x-axis mostly belong to state 3 which means good standing and the latter mostly belonged to state 5 which means deactivated. This means that the clusterer can do a better job in separating the different characteristics in the dataset with a better input value for 'k'. This is an indication that a bigger value for 'k' might identify distinct characteristics of the SIMs and group them into distinct clusters.

Fig. 5.5a demonstrates the clustering with 'k' equal to 5. In this figure, we can see that the SIMs along the x-axis that were previously colored in cyan in Fig. 5.4a are now colored in two different colors - cyan and blue. This is indicative of the different characteristics of those SIMs. In fact, manual inspection revealed that most of the SIMs colored in cyan belonged to state 5 (deactivated) and the ones colored in blue mostly belonged to state 3 (good standing). And the ones colored in pink had a mix of states 3 and 5 representing the SIMs that were similar in characteristics in terms of their communication volumes and patterns.

Fig. 5.5b demonstrates the influence of the different message types on the clustering. In Fig. 5.5a the following message types were considered: 'InvokeSendAuthenticationInfo', 'ReturnErrorUnknownSubscriber' and 'ReturnSendAuthenticationInfo'. Along with these, in Fig. 5.5b, 'InvokeUpdateGprsLocation' and 'ReturnUpdateGprsLocation' message types were also considered. This experiment shows how the clustering changed with the additional data. The SIMs along the x-axis that

were spread out into two different clusters colored in blue and cyan respectively, in Fig. 5.5a are now grouped into a single cluster colored in blue. This represents the similarity introduced by the new message types. The SIMs colored in pink still share similar characteristics in terms of their communication patterns and thus are grouped into the same cluster.

Fig.5.4b demonstrates the clustering with 'k' equal to 7. Here, the SIMs that belonged to a single cluster in the previous experiment, are spread across 2-3 clusters (clusters 4 to 6).This might be an indication of over fitting which means that the clusterer fits too closely to a limited set of data points. This may lead to errors rather than an optimal number of clusters.

For the purpose of POC, 5 was considered as the optimal value for k attributing to the minimum overlapping of clusters and a manual inspection of the message volumes.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
C1	0	0	69	0	1	0	0	0	0	0	0	0	0
C2	0	0	61	0	0	0	0	0	0	0	0	0	0
C3	0	0	2	0	11	0	0	0	16	0	0	0	0
C4	0	0	2	0	249	0	0	0	9	0	0	0	0
C5	18	23	13696	46	744	0	0	0	38	0	4	0	11

Figure 5.6: Matrix to illustrate the distribution of SIM states among the clusters

5.5 Observations

This section details the observations and inferences from the results of POC. k-means clustering was applied to the dataset consisting of SIMs belonging to various accounts and states.

Fig. 5.6 is a matrix that illustrates the the distribution of SIM states among different clusters. The matrix is a result of applying k-means clustering with a 'k' value of 5. Similar to the observations detailed in section 5.4, in the matrix, cluster 1 mostly contains the SIMS belonging to state 3. Cluster 2 consists of SIMS belonging to state 3 only. Cluster 2 is an example of a clear separation of the SIMS with distinct characteristics. Cluster 3 has a mix of SIMs belonging to states 3, 5 and 9. State 9 represents suspended status. Cluster 4 groups SIMs mostly belonging to state 5. Lastly, cluster 5 has a mixture of many states dominated by state 3. This represents the overlapping of state information of devices among the clusters. It is important to note that these SIMs belong to different accounts.

By examining the results, it was realized that the SIMs in different accounts do behave differently and exhibit distinct characteristics. So, by analyzing the SIMs partitioned by their account and also by their state would be a better path towards SIM characterization. Ignoring the account and state information would mean that all the SIMs are analyzed together. The partitioning also gives a fundamental understanding of the types of SIMs / devices being analyzed.

CHAPTER 6

FRAMEWORK DESCRIPTION

In this section, the various steps of the framework are detailed. The framework has several steps, starting with data preprocessing, cluster analysis, followed by cluster validation and labelling after which the labels are evaluated by applying k-NN binary classification. Finally, we extract actionable information from the results of previous steps with the support of domain expertise and other relevant information like past events. Fig. 6.1 illustrates the framework.

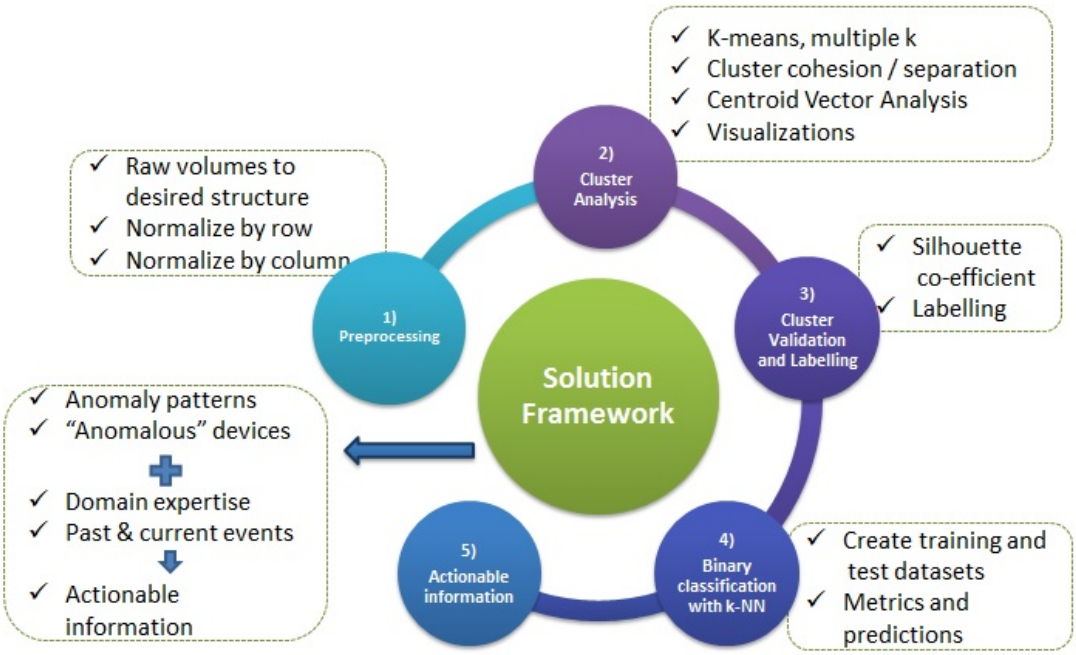


Figure 6.1: Framework Steps

6.1 Data preprocessing

Data preprocessing is an important starting step in the framework. This step transforms the raw SS7 transactions to a dataset that can collectively represent the SS7 message volumes by device. First, it is necessary to make sure that the dataset is free from missing or unknown values. Then the preprocessed data is analyzed by partitioning into different accounts and states which provides a fundamental understanding of the functionality of devices in each partition. The reasons for this are justified in the inferences derived from local experiments in chapter 5. As explained in section 4, analysis is focused on 14 important SS7 message types. In this step, the sum of message types that do not belong to the list of 14 types is computed and a new column called 'SumOfOtherMessageTypes' is created to represent the sum.

The raw data example in Table 4.1 is transformed to the data represented in Table 4.3. The mapping between SS7 message components in Table 4.1 and columns representing the names of message types in Table 4.3 can be found in Table 4.2. The new preprocessed dataset contains one row for each IMSI. The rows represent traffic volumes of each device over a period of one week.

6.2 Cluster analysis

The datasets are unlabeled and inherently irregular as a result of representing irregular events occurring in the network. Unsupervised learning is an effective tool for our case. Unsupervised learning techniques transform raw data into interesting structures and patterns which are hidden in the data. Cluster analysis is performed to find these hidden structures.

Cluster analysis groups devices in data such that the behaviors of devices in the same cluster are more similar to each other than those of devices in other clusters.

This is a popular technique used for unsupervised anomaly detection. In our context, an anomaly is a device which behaves differently and is not consistent with the behavior of other devices in the dataset. By definition of cluster analysis, such devices representing anomalies must cluster away from the ones with normal behavior.

Cluster analysis is performed by applying the simple and popular Lloyd's k-means algorithm in R, an open source tool for statistical computing [21]. The algorithm is explained in section 5.3 of chapter 5. The tools R [22] and, Weka [19] are used for cluster analysis and visualizations.

Determining the optimal value for 'k' is a popular problem in data analysis and several methods exist for making this decision. The inherent heterogeneity and skewed distribution of the traffic volumes make the decision making process more challenging.

A collaborative and exploratory approach is followed to choose an optimal k as detailed below. Fig. 6.2 demonstrates the approach and each step is described below. The figures representing WCSS / BCSS and optimal 'k' are expanded in Fig. 6.3.

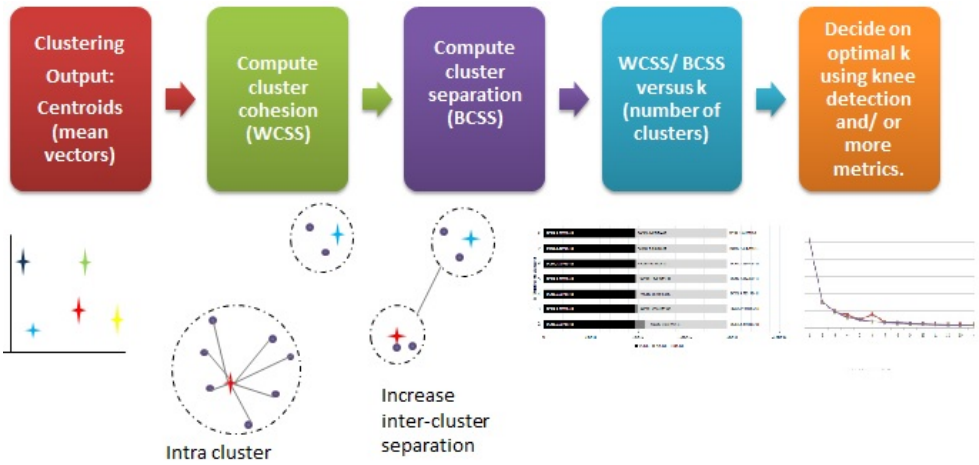
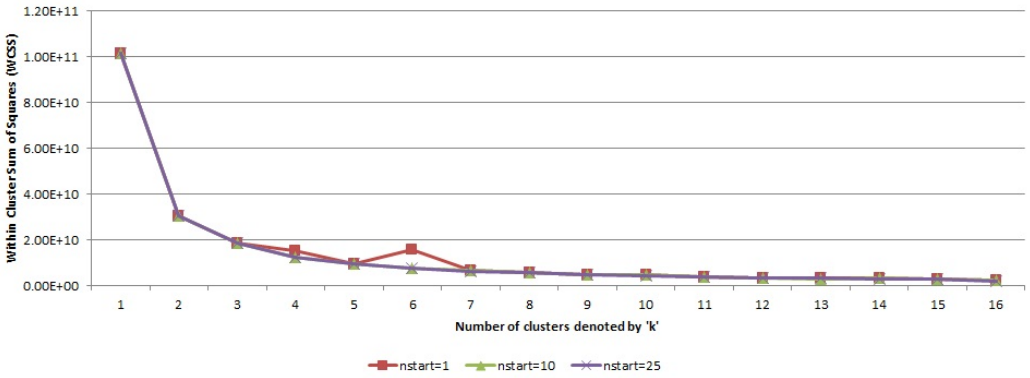
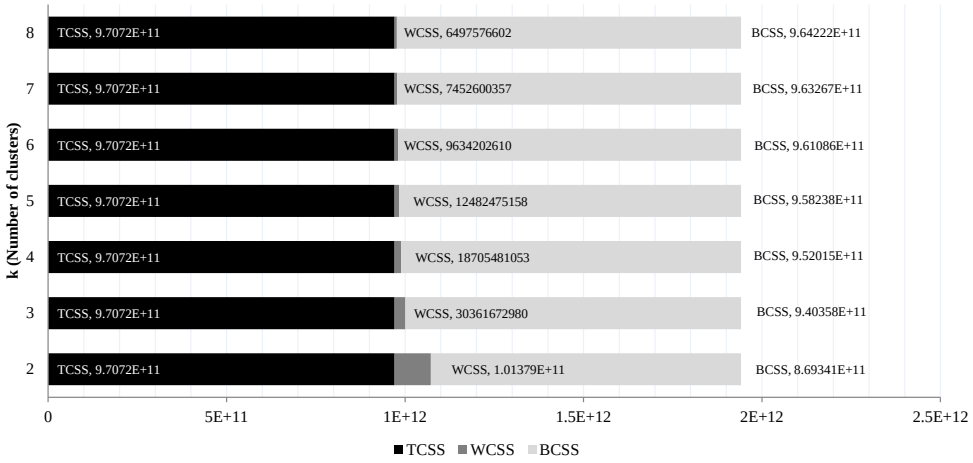


Figure 6.2: Cluster analysis



a: k vs WCSS with nstart trends



b: Relationship of WCSS and BCSS

Figure 6.3: Optimal k analysis

6.2.1 Measures from cluster analysis

Within Cluster Sum of Squares (WCSS) is a measure of cluster cohesion and Between Cluster Sum of Squares (BCSS) is a measure of cluster separation. These two measures are considered to determine the optimal k. The goal is to find a 'k' which minimizes WCSS and maximizes BCSS so that the clusters are clearly separated indicating clear distinction of characteristics in the data. However, this is a rare occasion. The datasets are inherently irregular and heterogeneous.

Knee detection method is applied where the dataset is clustered for several values of 'k' producing clusters and WCSS value. These values are then visualized

with 'k' on the x-axis and WCSS on the y-axis. WCSS decreases with increasing value of k. A knee can be defined as a point where the decrease in WCSS slows down. Optimal 'k' is the point at which a knee is formed. Fig. 6.3a depicts the knee plot for vehicle tracking devices. Although a clear knee is not evident, multiple knee points can be seen at 'k' values 4 and 5. Fig. 6.3b demonstrates the relation of BCSS and WCSS. BCSS increases with decreasing WCSS.

6.2.2 Centroid vector analysis

Centroids produced by k-means algorithms are the means of the dataset. These centroids and cluster sizes are analyzed for unusual numbers. Domain expertise is an invaluable tool here. The identification and interpretation of patterns are detailed in case studies in chapter 8.2.

6.2.3 Visualizations

Data visualizations are indispensable in the decision making process. Centroid vectors that represent the means of big data set are visualized with an intent to spot unusual patterns or anomalies. Our case studies in chapter 8.2 illustrate these visualizations. The experiments detailed in chapter 8 describe the investigations conducted in the process of determining optimal k.

6.3 Cluster validation and labelling

Silhouette coefficient is another popular method to validate clusters of data based on Euclidean distance. Silhouette is a measure of how closely an observation is matched to other observations inside its cluster and how loosely it matches those in other clusters. A value close to 1 implies that the observation fits well in its cluster. A value close to -1 implies that the observation is a misfit, i.e. the observation is

in a wrong cluster. An average silhouette over all the observations in the dataset is a measure of how appropriately the dataset is clustered. This method is used to validate our choice of k. A value of close to 1 indicates good quality clusters. The expression below demonstrates the silhouette coefficient computation.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i),\ b(i)\}}$$

where a(i) is the average dissimilarity of point i to a cluster c (i belongs to c),
b(i) is the lowest average dissimilarity of point i to any cluster other than c,
and -1 $\leq s(i) \leq$ 1.

The goal of this research is to identify anomalous devices in the dataset. We step closer to the goal by labelling the validated clusters under the assumptions explained below:

- By the definition of cluster analysis, similarly behaving devices are grouped together in the same cluster and devices with dissimilar behaviors belong to different clusters. [23]
- A fundamental assumption of unsupervised anomaly detection is that a majority of devices behave normally and only a small subset of the devices behave anomalously [24]. These anomalous devices do not fit the behavior of most of the devices. This means that this small number of anomalous devices must be in a separate cluster provided the clusters are well separated and are of good quality.

These assumptions are illustrated in our case studies in chapter 8.2. A single small cluster can be observed. These small number of devices are clustered away from the rest of the clusters. A very large cluster can also be observed. This probably indicates "normally" behaving devices.

In view of the assumptions and knowledge acquired from experiments and case studies, the validated clusters are labelled as either “anomalous” or “normal”. The smallest cluster is labeled “anomalous” and the biggest cluster is labeled “normal”. Labeled devices are considered as a training dataset and the remaining unlabeled devices are considered as a test dataset. Next, a k-NN classifier is trained with the training set and the test set is classified into the two categories.

6.4 Binary classification with k-NN

k-NN classification is a non-parametric technique used for anomaly detection. It is important to note that ’k’ in the context of k-NN means number of neighbors whereas ’k’ in the context of cluster analysis means number of clusters. k-NN classifies an observation by a majority vote of its ’k’ neighbors, with the observation being assigned to the most common class / label. A weighted Euclidean distance 1/d where ’d’ is the Euclidean distance of the neighbor is used, in order to let nearest neighbors contribute more to the average than the distant ones. This makes sense as similarly behaving devices are closer to each other than devices with dissimilar behaviors. A technique called n-fold cross-validation is used to choose the number of neighbors where the algorithm chooses the appropriate value for ’k’. In n-fold cross-validation, the original dataset is randomly partitioned into n equal sized subsets. Of the n subsets, a single subset is retained as the validation data for testing the model, and the remaining (n - 1) subsets are used as training data. The cross-validation process is then repeated n times (the folds), with each of the n subsets used exactly once as the validation data. The n results from the folds can then be averaged (or otherwise combined) to produce a single result [25].

The labeled devices are classified as either “anomalous” or “normal” based on the weighted Euclidean distance function and majority voting. The k-NN classifier

outputs evaluation metrics such as accuracy, root mean square error, true positive rate, false positive rate and confusion matrix of cluster labels and predicted labels. Accuracy is defined as follows:

$$Accuracy = \frac{No.\ of\ correctly\ classified\ devices}{No.\ of\ devices} \times 100$$

These metrics and results evaluate our labeling.

6.5 Actionable information

The visualizations in cluster analysis step disclose interesting patterns and structures in datasets. Analysis of these patterns leads to insightful knowledge accumulation about the datasets both in general and in anomaly detection context. The case studies detailed in section 8.2 demonstrate the anomaly patterns and structures. The anomaly patterns and labeled device information bring to light insightful actionable information when combined with relevant domain expertise, prior knowledge about past events and other facts like the account and state devices belong to. This actionable information is valuable and boosts further analysis by directing the focus on a small subset of devices, eventually leading to detection of faulty or truly anomalous devices.

CHAPTER 7

CLOUD IMPLEMENTATION

In this section, cloud implementation of the framework which is described in chapter 6 is outlined. The framework is made scalable, cost-effective and efficient by leveraging cloud computing resources, and distributed storage and computing offered by the Hadoop ecosystem. The cloud solution framework is illustrated in Fig. 7.1. Raw data extracted from the M2M network is stored using Amazon Simple Storage Service (S3) [11].

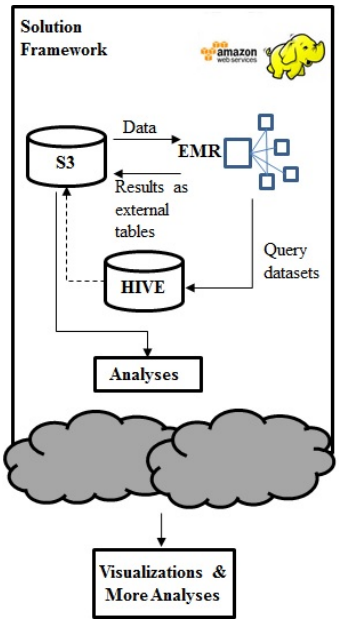


Figure 7.1: Cloud framework architecture

The framework is implemented using Amazon Web Services (AWS). With AWS, vast amounts of data can be analyzed by distributing the computational work across

a cluster of virtual servers running on the Amazon cloud. This cluster is managed by Apache Hadoop, an open-source framework. Data preprocessing, cluster analysis, silhouette computation and k-NN are all performed using Apache Hive queries that are executed on an Amazon Elastic Map Reduce (EMR) Hive cluster. The computations are implemented as custom reduce scripts using bash, R, and RWeka [26] scripts. These scripts are run in the reduce step of Map and Reduce phases of Hive jobs. Data such as raw datasets, preprocessed datasets, cluster centroid vectors, clustered datasets, silhouette coefficient and k-NN result metrics are all stored as Hive External tables on S3.

Here is a sample hive query which is used to cluster the message volumes of devices stored in a table called 'Preprocessed_Data'. The data this table holds can be seen in Table 4.3. This data is input into a custom R script named 'Cluster_Sims.R' which is responsible for performing cluster analysis and outputting the cluster assignments for each device. The 'k' value is passed as an argument to this script. In this example a value of 5 is used. This Hive query will generate a MapReduce job. The R script is executed in the reduce phase of the job. The results of this query which are cluster assignments are inserted into the external table named 'Clustered_Devices' along with the input. The destination table is partitioned by many columns along with the cluster id assigned. Amazon S3 stores partitions in the form of directories.

FROM (FROM (

SELECT * FROM **Preprocessed_Data**) v

MAP v.Imsi, v.Msisdn,

v.InvokeSendAuthenticationInfo,

v.ReturnSendAuthenticationInfo,

. . . remaining message types . . .

v.SumOfOtherMessageTypes,

v.year, v.week, v.AccountId, v.StateId

USING '/bin/cat'

AS Imsi, Msisdn,

InvokeSendAuthenticationInfo,

ReturnSendAuthenticationInfo,

. . . remaining message types . . .

SumOfOtherMessageTypes,

year, week, AccountId, StateId

CLUSTER BY Year, Week, AccountId, StateId) d

INSERT INTO TABLE **Clustered_Devices** PARTITION (year, week, AccountId, StateId, k, clusterId)

REDUCE d.Imsi, d.Msisdn,

d.InvokeSendAuthenticationInfo,

d.ReturnSendAuthenticationInfo,

. . . remaining message types . . .

d.SumOfOtherMessageTypes,

d.year, d.week, d.AccountId, d.StateId

USING **'Cluster_Sims.r 5'**

AS Imsi, Msisdn,

InvokeSendAuthenticationInfo,

ReturnSendAuthenticationInfo,

. . . remaining message types . . .

SumOfOtherMessageTypes,

year, week, AccountId, StateId, k, **clusterId**;

The cloud implementation allows us to take advantage of the benefits of distributed computing on machines with high computational power that form the EMR cluster. The cluster configuration is very flexible and can be tuned to elastic business demands. Important advantages from the cloud are elasticity, scalability and fault tolerance offered by Hadoop.

CHAPTER 8

EXPERIMENTS AND RESULTS

In this chapter the experiments that aided in structuring the framework are described. Preliminary experiments are detailed in section 5.4. Areas covered include cluster analysis, anomaly patterns found in raw and normalized datasets, and k-NN classification results.

8.1 Experiments

This section details the experiments conducted by varying k-means algorithmic parameters for cluster analysis. The significance of the parameters is also explained. Experiments were performed to learn how well a few cluster analysis algorithms other than k-means work with our dataset. The clustered dataset was visualized by reducing the 15 dimensions to 2 or 3 using PCA. Results from these experiments are detailed below.

8.1.1 Cluster Analysis

As explained in section 6.2, Lloyd's k-means algorithm implemented in R is used for cluster analysis. The algorithm is sensitive to the seed value. With an attempt to reduce the sensitivity, a parameter named 'nstart' which can be seen in the implementation in R is used. The parameter 'nstart' is an integer value with a default value of 1. The algorithm computes 'nstart' number of random initial configurations for cluster centers and aims to partition 'n' observations into 'k' clusters such that WCSS is minimized. The goal is to find the best initial selection of centroids. This

approach is often recommended [22]. The seed value is kept constant so that the results are reproducible. Fig. 6.3a demonstrates relationship of WCSS and 'k' along with 'nstart' trends. The line representing 'nstart' value of 10 is smoother than the one with default value of 1. This means that the variation in WCSS decreases when better initial centers are selected. With real datasets, finding a clear knee is rarely possible. Therefore, the centroid vectors and visualizations are also analyzed to decide on an optimal 'k' value. Centroid vectors are the centers of the clusters in 15-dimensional space where the dimensions represent 14 message types and the sum of other message types. We examine these aiming to identify distinct centroid co-ordinates. An example of a distinct co-ordinate is highlighted in Table 8.1 which displays the centroid vectors for a security account dataset. The centroid co-ordinate for 'InvokePurgeMs' message type contains a distinct co-ordinate value of 41.1162 for cluster 2. By examining the centroid vector for cluster 2 we can infer that cluster 2 is located far away from other clusters in the Euclidean space.

8.1.2 Other clustering algorithms

Experiments were conducted by performing cluster analysis with algorithms other than k-means. Hierarchical clustering [27], Cobweb [28], and k-means++ [29] clustering were performed. Hierarchical clustering and Cobweb are compute and time-intensive algorithms. The results did not seem to match our needs at that point of time. However, more experiments are needed to draw conclusions about these algorithms. At the time of conducting experiments, k-means suited our data and goals the best. So, efforts on analysis with k-means and k-means++ were taken forward.

k-means++ is an improvised version of k-means and aims to reduce the sensitivity of the algorithm to initial seed value by specifying a procedure to initialize

Table 8.1: Description of centroids for security dataset

Message Type	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
InvokeSendAuthenticationInfo	187.0366	3582.5851	776.7758	205.4188	24.471
ReturnSendAuthenticationInfo	180.5045	3581.1286	764.3554	201.2777	15.5561
ReturnErrorUnknownSubscriber	6.5757	1.473	12.4311	4.153	8.9154
InvokeUpdateLocation	48.6868	4076.7469	750.6562	214.2052	5.5581
ReturnUpdateLocation	48.3041	4076.3237	749.1495	213.9152	5.5514
InvokeCancelLocation	14.1095	2473.9627	551.8125	87.9201	1.6415
ReturnCancelLocation	14.1091	2473.9585	551.8107	87.9116	1.6413
InvokeInsertSubscriberData	28.599	4269.3237	720.1139	187.1654	4.1065
ReturnInsertSubscriberData	28.5628	4265.0622	717.9727	187.0217	4.1036
InvokeUpdateGprsLocation	2.8701	398.971	172.4958	14.0734	0.2132
ReturnUpdateGprsLocation	2.827	398.9668	172.476	14.06	0.2127
InvokePurgeMs	0.6167	41.1162	2.9882	1.0192	0.0382
ReturnPurgeMs	0.6166	41.1162	2.9882	1.0192	0.0382
TcapAbort	0.002	0.3278	0.0346	0.0144	0.0004
SumOfOtherMessageTypes	6.5643	21.4772	7.9601	6.2955	5.6035
Cluster Size	50k	250	4.5k	28.5k	270k

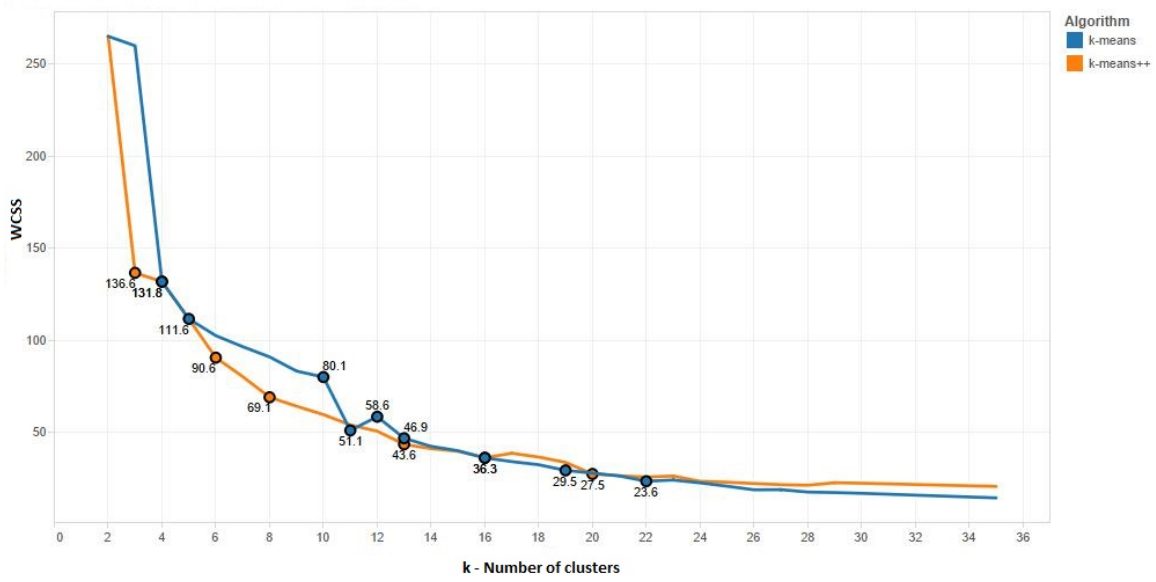


Figure 8.1: k vs WCSS - Comparison of k-means and k-means++

cluster centers. This procedure is applied before k-means is performed. Results from experiments with k-means++ on Weka are described below. The visualizations in Fig. 8.1 and Fig. 8.2 were created using Tableau [30].

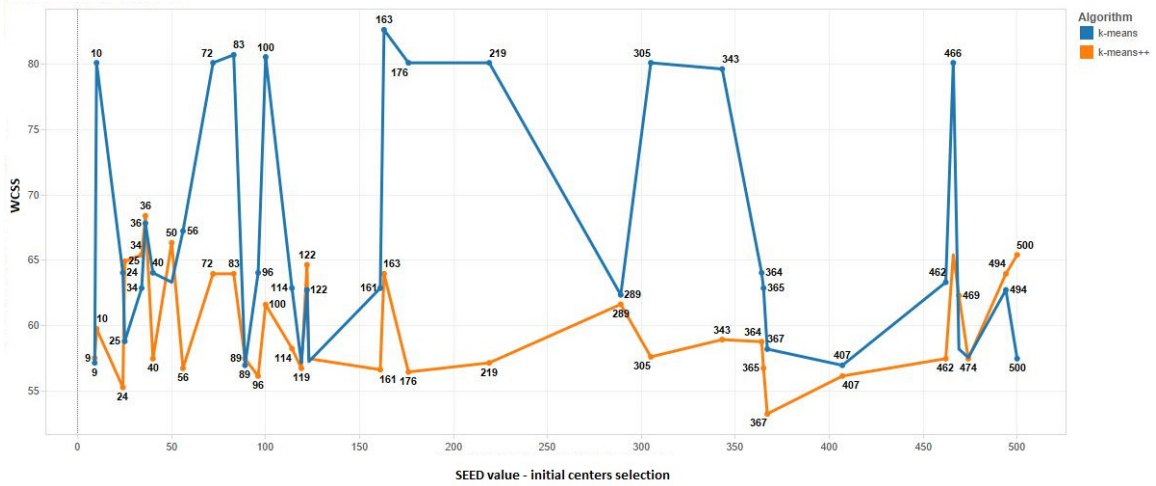


Figure 8.2: Seed sensitivity- Comparison of k-means and k-means++

Fig. 8.1 displays a comparison of k-means and k-means++ algorithms with respect to WCSS which is a measure of cluster cohesion. The highlighted points in the figure represent possible knee points. The concept of knee-detection which is used to decide optimal 'k' is explained in section 6.2.1. It can be observed that there is little difference in the charts for k-means and k-means++.

Fig. 8.2 demonstrates a comparison of k-means and k-means++ algorithms with respect to sensitivity to seed values. It can be observed the variation of WCSS with changing seed values is lower with k-means++. However, the difference in results from k-means and k-means++ were not substantial. As explained in the previous section, 'nstart' is a parameter which can be used to select good initial centers. The research was carried further with the k-means implementation in R so that the 'nstart' parameter could be leveraged. The seed value was fixed to a certain value for the purpose of the experiments in order to ensure a repeatable process.

The version of k-means (Lloyd's algorithm) used in our framework is widely used and finds reasonable solutions most times. So, the cluster analysis efforts were focused on k-means for the current version of the framework. This was done to focus

on the bigger picture of anomaly detection instead of drilling down to well-known problems specific to k-means.

8.1.3 Dimensionality Reduction using PCA

In this step, the data resulting from cluster analysis is visualized in reduced dimensions. It aids in identifying relationships among message types in the data. 15 dimensions make it impractical to be able to make any meaningful sense out of the data. So, the 15 message types are reduced to fewer dimensions - 2 or 3 artificial attributes that represent the same data. These artificial attributes are called principal components. Principal Component Analysis (PCA) is a statistical procedure that uses orthogonal transformation for dimensionality reduction. The new dimensions are arranged in a decreasing order of the variance in the data they account for, i.e. first principal component accounts for the largest possible variance in the dataset, and so on. PCA was performed using RapidMiner [31]. The parallel co-ordinate plot visualized using RapidMiner in Fig. 8.3 illustrates the variances that the 15 principal components account for. The principal components are represented on the x-axis with variance indicated on the y-axis.

We visualize the reduced dimensions on first 2 or 3 principal components. This visualization is also a demonstration of the quality of clustering. Non-overlapping clusters (colors) indicate tight clustering and support our optimal k conclusion. Completely separated clusters rarely occur with real datasets. Fig. 8.4 demonstrates clusters next to each other with little overlapping. This visualization was created using R. Fig. 8.5 represents the same information in 2D. This visualization was created using Tableau.

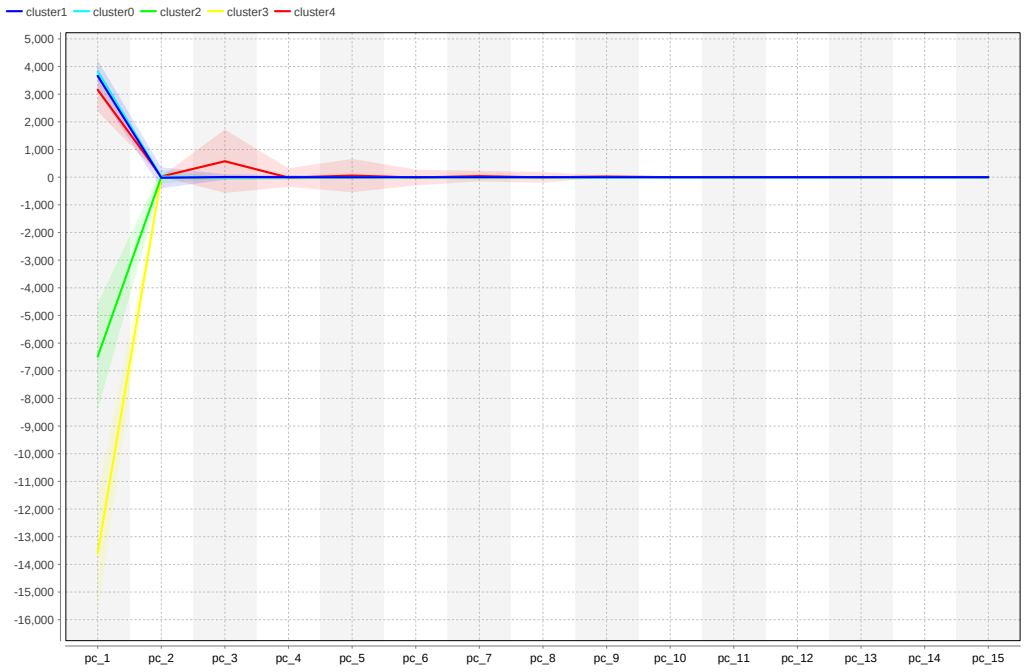


Figure 8.3: Parallel co-ordinate plot after applying PCA on clustered vehicle tracking dataset

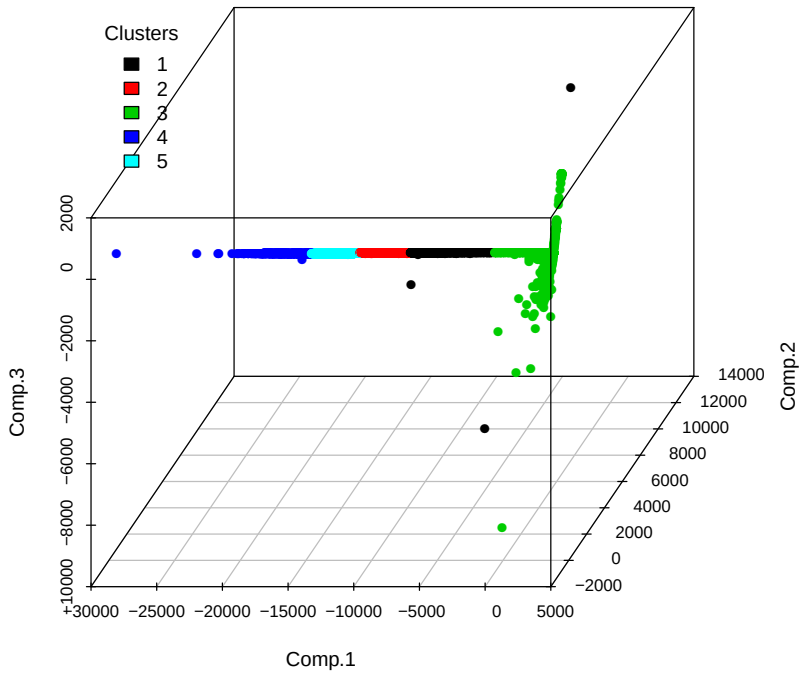


Figure 8.4: Vehicle tracking devices on first 3 principal components - 3D

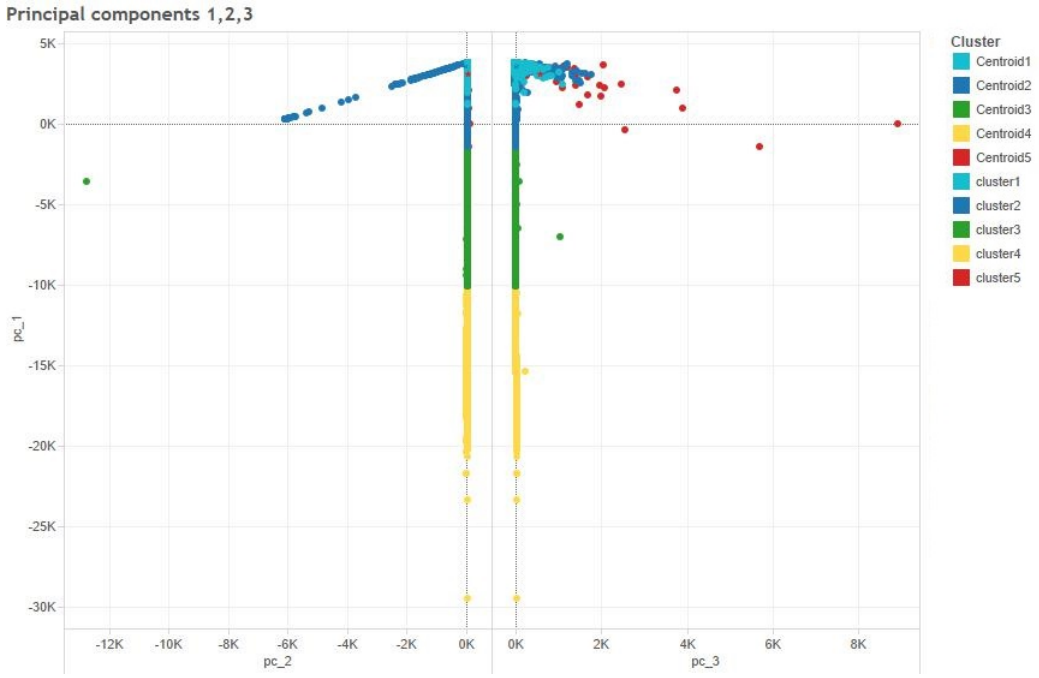


Figure 8.5: Vehicle tracking devices on first 3 principal components - 2D

8.2 Case Studies

This section details the case studies demonstrating the anomaly patterns and anomalous devices discovered by applying the framework to the following two accounts: vehicle tracking, and security account.

8.2.1 Anomaly Patterns

In this section, the "anomaly" patterns discovered in both unnormalized and normalized datasets are presented.

8.2.1.1 Vehicle Tracking Account

Vehicle tracking devices are responsible for transmitting tracking information of the vehicles on which they are installed. The total number of devices in the dataset analyzed is close to 23k.

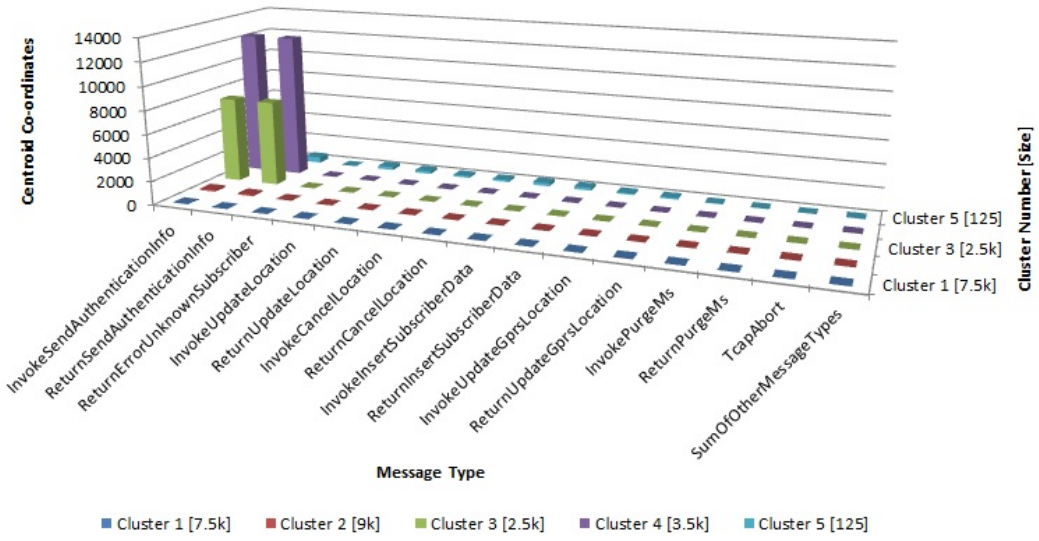


Figure 8.6: Patterns discovered in unnormalized vehicle tracking dataset

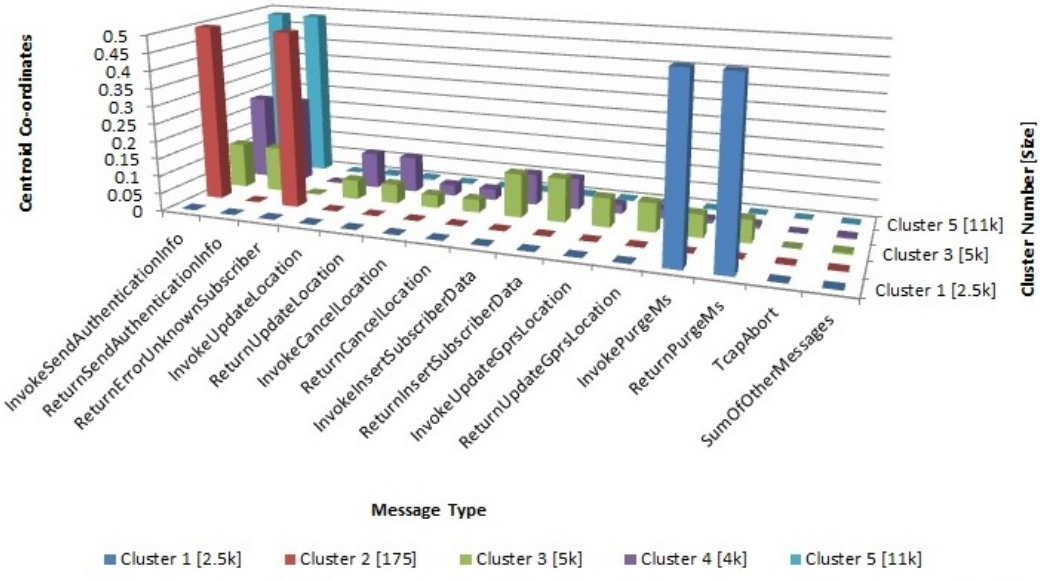


Figure 8.7: Patterns in vehicle tracking dataset normalized by row

Fig. 8.6 visualizes the cluster vectors of unnormalized dataset. The clusters 3 and 4 highlight devices that behave differently than the devices in other clusters. This difference is with respect to 'InvokeSendAuthenticationInfo', and 'ReturnSendAuthenticationInfo' messages relayed by the devices. The messages indicate that these

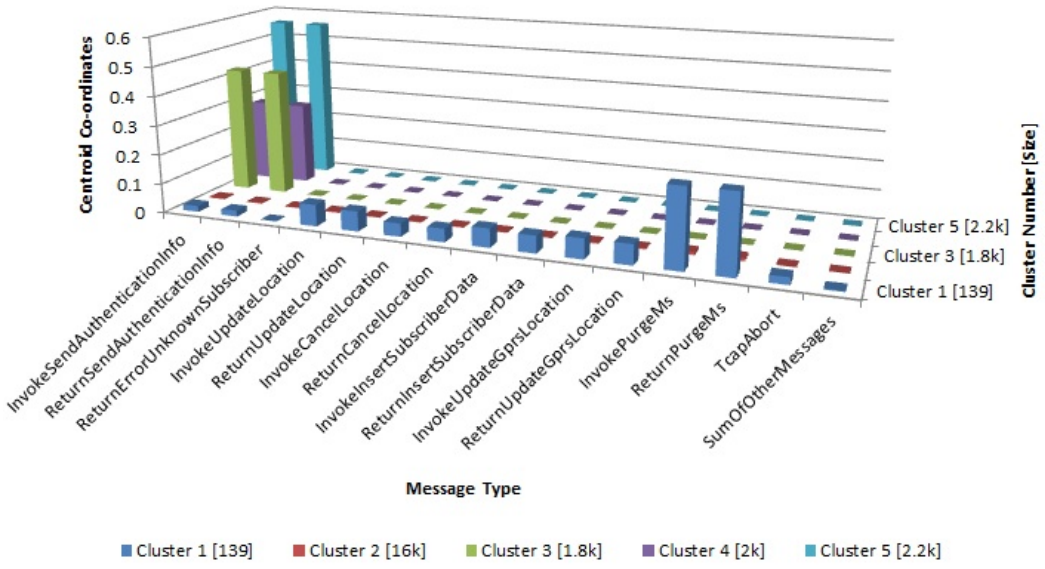


Figure 8.8: Patterns in vehicle tracking dataset normalized by column

Table 8.2: Description of centroids for vehicle tracking dataset

Message Type	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
InvokeSendAuthenticationInfo	24.4408	152.3611	7314.6047	12338.8311	491.877
ReturnSendAuthenticationInfo	24.4407	119.2664	7310.6036	12335.5264	491.877
ReturnErrorUnknownSubscriber	0.0003	33.0951	4.0019	3.3013	0
InvokeUpdateLocation	6.2862	10.2136	0.265	0.0495	269.8033
ReturnUpdateLocation	6.2859	10.2121	0.2672	0.056	245.877
InvokeCancelLocation	3.7972	12.1702	0.0211	0.1367	183.2377
ReturnCancelLocation	3.7968	12.1699	0.0211	0.1361	183.2377
InvokeInsertSubscriberData	5.3638	13.2403	0.2709	0.0498	267.4344
ReturnInsertSubscriberData	5.3588	13.2205	0.2705	0.0498	243.3115
InvokeUpdateGprsLocation	3.7001	8.9523	0.0056	0	132.1557
ReturnUpdateGprsLocation	3.6971	8.9501	0.0056	0	131.9836
InvokePurgeMs	1.3948	0.0241	0.0148	0.0059	20.2541
ReturnPurgeMs	1.3948	0.0238	0.0118	0.0028	20.2541
TcapAbort	0.0018	0.0009	0	0	0.0738
SumOfOtherMessageTypes	0.8471	1.9165	0.3879	0.0823	24.5
Cluster Size	7.5k	9k	3k	3k	150

devices attempted to authenticate themselves with the network more frequently than other devices. There can be many reasons behind the transmission of this message type such as the device might be roaming, the device might have lost connection, the

device might be just maintaining its active functional state or the device might have entered an error state and therefore, is trying to get back to its functional state by attempting authentication more times than usual. However, being in an error state, the device's attempts to authenticate turn out to be futile and will eventually end up in a persistent chain of authentication attempts and responses. The last row in Table 8.2 lists the sizes of the five clusters. The clusters 3 and 4 together account to around 27% of the devices in the dataset.

As mentioned in section 4, normalization helps in eliminating the bias introduced by message types with high message counts and in turn brings many other significant messages types to focus. Fig. 8.7 depicts the patterns discovered in dataset normalized by row. It can be observed that cluster 1 contains devices with high count of 'PurgeMS' messages, cluster 2 identifies around 175 devices which receive an error message called 'ReturnErrorUnknownSubscriber' as a result of their requests for authentication. This error message means that these devices were not identified at the HLR and suggests further investigation of the network elements. Clusters 3 and 4 contain devices with a mix of different message types and 0 'ReturnErrorUnknownSubscriber'. Cluster 5 contains around 11k devices with high 'InvokeSendAuthenticationInfo', and 'ReturnSendAuthenticationInfo' and 0 'ReturnErrorUnknownSubscriber'. This cluster probably contains "normal" devices.

Fig. 8.8 illustrates the patterns discovered in dataset normalized by column. The messages 'InvokePurgeMs' and 'ReturnPurgeMs' are brought to attention in cluster 1. Purge MS is a communication message sent by the VLR to HLR indicating that the subscriber's (device) data was removed because the device had been inactive for an extended period of time or is being forced off the network. Another observation is that 'InvokeSendAuthenticationInfo', and 'ReturnSendAuthenticationInfo' messages

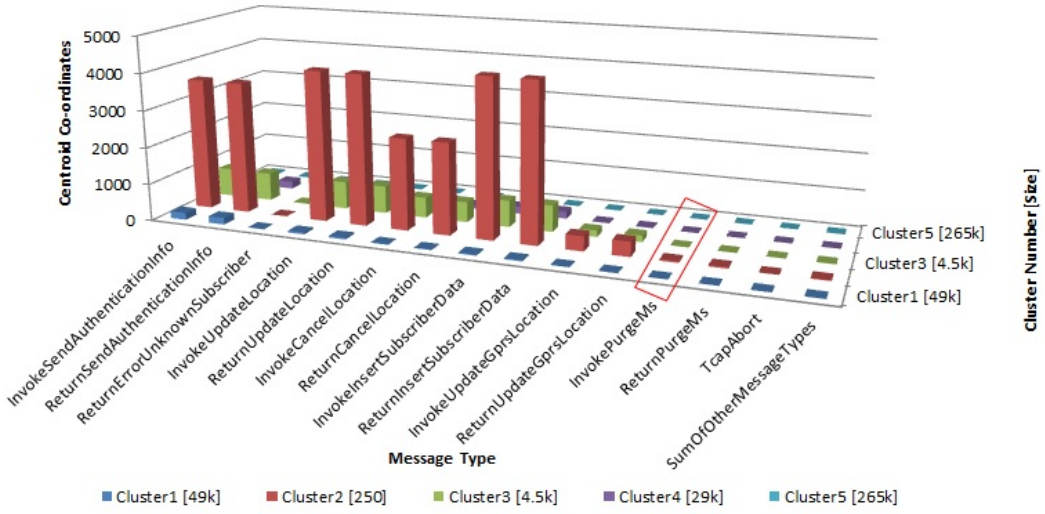


Figure 8.9: Patterns discovered in unnormalized security account dataset

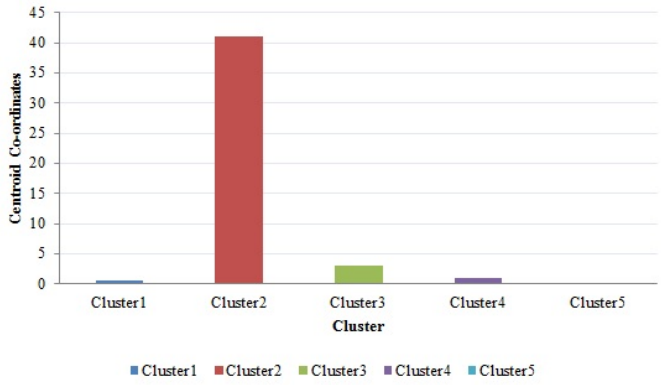


Figure 8.10: 'InvokePurgeMS' pattern in unnormalized security account dataset

are significant in clusters 3, 4 and 5 than other clusters. Cluster 2 consists of devices which talk less compared to other devices in the dataset.

8.2.1.2 Security Account

Security account contains devices embedded in stationary systems like home security systems, and commercial security systems. The total number of devices in the dataset analyzed is close to 350k.

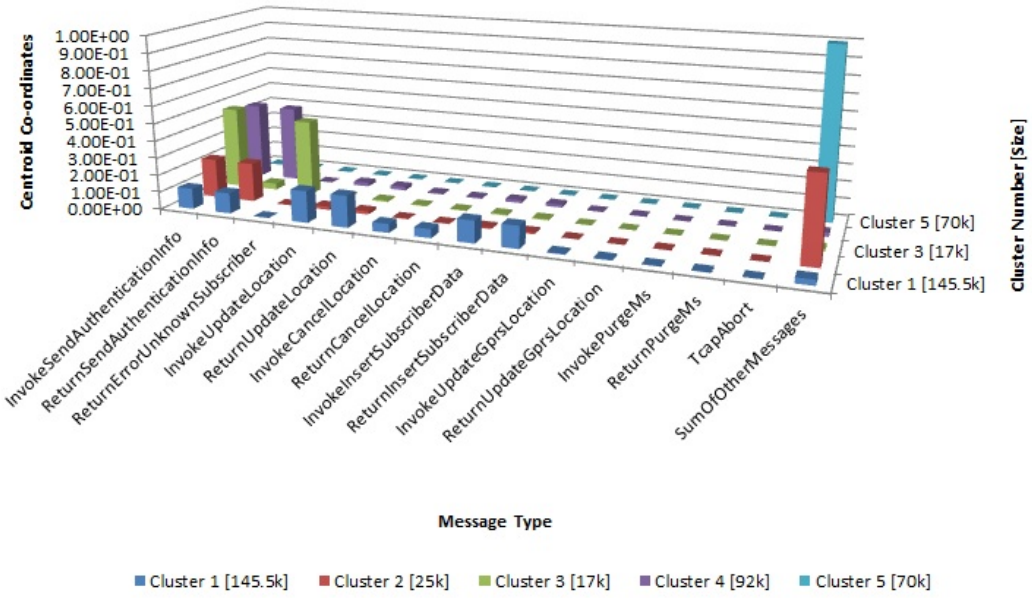


Figure 8.11: Patterns in security dataset normalized by row

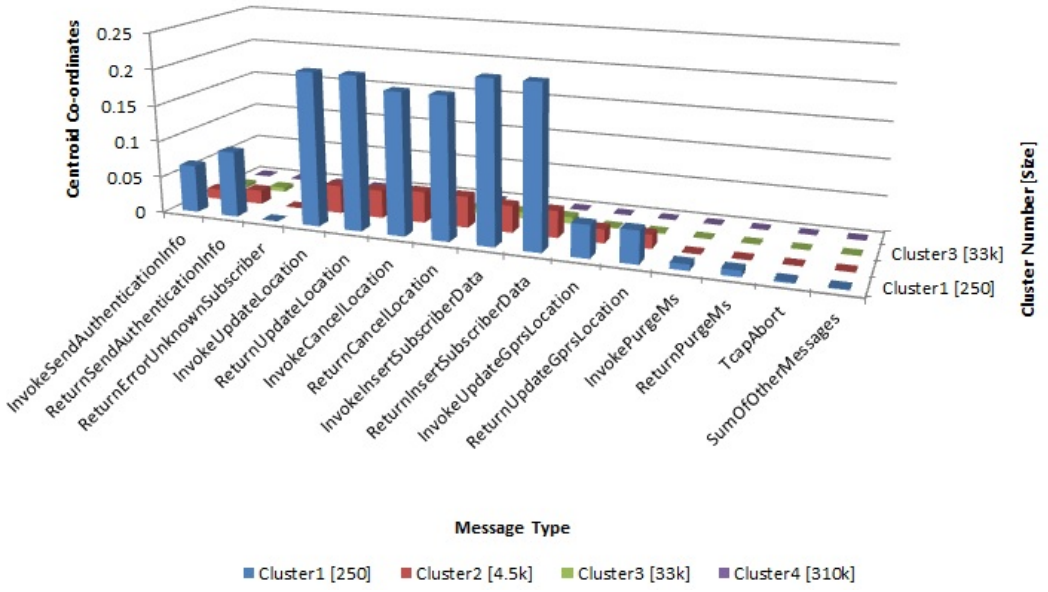


Figure 8.12: Patterns in security dataset normalized by column

Fig. 8.9 demonstrates the centroid vectors of clusters resulting from cluster analysis of unnormalized data. The information concealed in it is hardly interpretable unless visualized in different ways. Fig. 8.10 is a view of Fig. 8.9 along the z-axis. It

represents a pattern of the message type named 'InvokePurgeMs'. This figure reveals an important pattern where cluster 2 alone contains all the devices with a high count of 'InvokePurgeMs'. Around 250 devices are discovered out of the total 350k devices in the dataset which amounts to 0.07% of devices. Given the fact that these devices are grouped in the same cluster and that cluster is the smallest cluster, we can infer that this might be a group of misbehaving devices. With the discovery of this pattern, our focus has now been reduced to just 250 devices out of the total 350k devices. With the support of domain expertise and knowledge about past events, it was speculated that a probable cause for this pattern was the maintenance shutdown initiated by a network partner. This event could have caused all the devices being serviced by that specific network partner to be purged. This case expresses the influence of the external network partner on our devices and communication patterns. Each ("anomaly") pattern can be studied to make different inferences about the devices, their behaviors and influencing factors.

Fig. 8.11 demonstrates the communication patterns found in security dataset normalized by row. It can be observed that clusters 1, 2, and 4 contain devices with authentication request and response messages types. The clustering technique has managed to separate out 17k devices out of total 350k with 'ReturnErrorUnknownSubscriber' error message into cluster 3. Another interesting observation is that cluster 5 contains devices which communicate messages of type other than the 14 types that we have considered and listed in Table 4.2.

Fig. 8.12 demonstrates the patterns discovered in dataset normalized by column. Here we consider a 'k' value of 4 to input the number of clusters for cluster analysis. It can be observed that cluster 1 highlights 250 devices out of total 350k devices. These devices show a mix of message types and almost zero 'ReturnErrorUnknownSubscriber'. Cluster 2 represents devices with a similar behavior but less talkative

ones. Cluster 4 is the biggest cluster with devices with low communication. An interesting observation found here was that the 250 devices contained in cluster 1 are the same 250 devices that were highlighted in cluster 2 of unnormalized dataset in Fig. 8.9. This finding emphasizes that these devices are strongly similar to each other and are strongly different from other devices as they were separated into a different cluster in both normalized and unnormalized contexts.

8.2.2 Labelling and classification

After normalization, data is scaled to fit in a specific range. In our case, the range is [0,1]. Normalization techniques ensure that all message types contribute to the distance computation. The normalized clusters are labelled under the assumptions listed in section 6.3. As a result of labelling, our dataset is partitioned into training and test sets. Further, k-NN binary classification is applied to classify the unlabeled test set.

Fig. 8.13 shows the average silhouette. The average silhouette for the vehicle tracking dataset is 0.92 in this case. The silhouette of each cluster is also close to 1. This is an indication of good quality clustering.

After the clusters are validated we labeled the clusters. The smallest cluster in the dataset normalized by row is cluster 2. It contains 175 devices out of the total 23k. This cluster was labeled "anomalous". The biggest cluster, cluster 5 consisting around 11k devices was labeled "normal". These two clusters form the training set. The remaining clusters 1, 3, and 4 form the test set. A k-NN classifier was modeled with the training data. The accuracy of the model was 99.2%. Then the test data was classified using the k-NN model generated. 0.04% of the predicted labels for clusters 1, 3, and 4 turned out to be "anomalous" and the rest were labeled "normal". Summing up all the numbers around 0.7% of devices were labeled "anomalous" in the dataset.

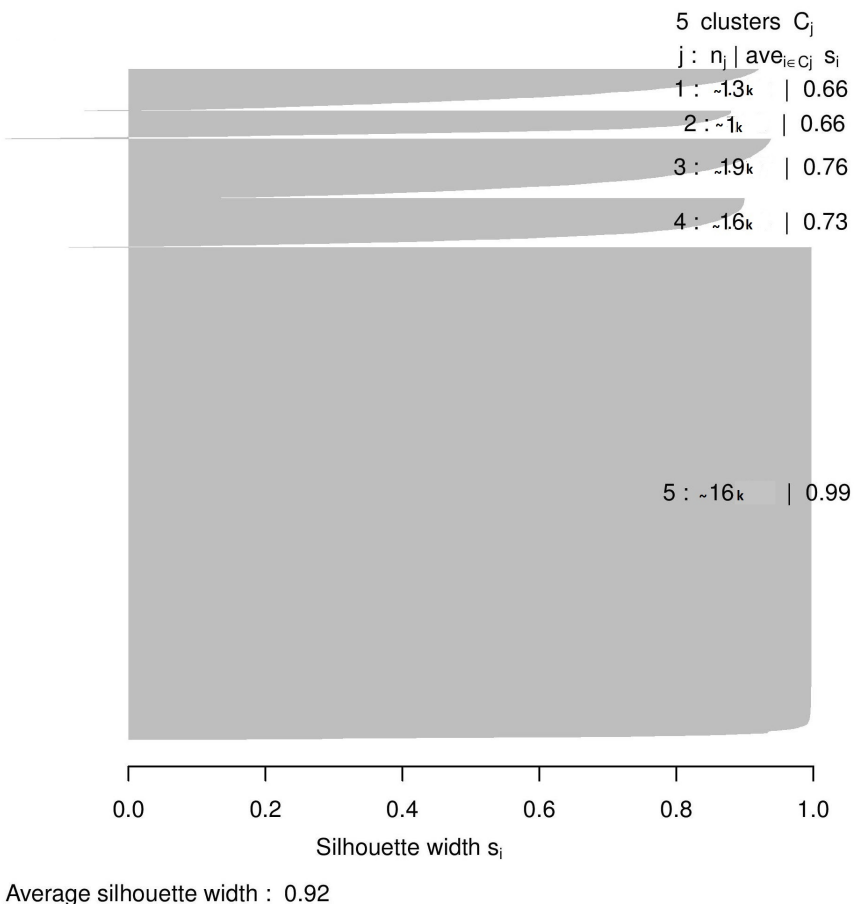


Figure 8.13: Silhouette coefficient for vehicle tracking dataset

The results were different when the same process was carried out for dataset normalized by column. The smallest cluster is cluster 1 with 139 devices and the biggest is cluster 2 with around 16k devices. The test set consisted of devices from clusters 3, 4, and 5. The accuracy of the k-NN model was 99.7%. All the devices in the test set were labeled "normal". Summing up all the numbers around 0.6% of devices were labeled "anomalous".

CHAPTER 9

SUMMARY AND CONCLUSION

This research started with a goal to classify device behaviors and identify anomalous devices on the M2M network of a leading M2M service provider. Identifying anomalous devices is analogous to finding a needle in a haystack. Challenges involved in anomaly detection in a real heterogeneous M2M network were detailed in early chapters. A proof of concept (POC) which was the first step towards the framework was described. The burgeoning future of M2M strengthens the need for an automated, proactive and scalable framework. One such framework which is based on data mining was presented in this thesis. The Hadoop based framework is scalable and fault tolerant. This framework is also cost-effective and elastic to the needs of analysis. All these are achieved by leveraging the benefits of cloud computing and Hadoop framework.

Experiments and analyses were conducted on real datasets which represent the inherent irregular nature of the M2M network. The framework was applied to 2 accounts: vehicle tracking, and security. Various anomaly patterns were discovered and described in the case studies. These patterns aided in understanding the devices by using just their communication volumes. The case studies demonstrated the identification of small groups of "anomalous" devices out of hundreds of thousands of devices in the network. This reduced the size of our "haystack" significantly; From hundreds of thousands to a few lower hundreds. This, coupled with domain expertise and information about past events make it easier to identify faulty devices. The knowledge and

learning gained from the framework can help in intelligent decision-making process, fault diagnosis and device monitoring.

Anomaly detection in M2M will continue to be an important area of research with the growing M2M / IoT market. This thesis presented anomaly detection solution in the form of a cloud based, scalable, and cost-effective data mining framework.

CHAPTER 10

FUTURE WORK

In the future, experiments with unsupervised learning algorithms apart from k-means will be a good space to explore. With the framework in place, we can now run many compute-intensive machine learning algorithms such as hierarchical clustering on the cloud. By combining the results of current framework with the analyses by applying different algorithms that work well with our datasets will bring to light compelling results. These results might lead us to new dimensions or ways to solve anomaly detection.

This thesis has enhanced the understanding of the traffic volumes of the devices and their account-wise communication patterns. With this information and by conducting more experiments, relationships among the 14 important message types can be explored. For example, authentication request, response, and error message types share a relationship. After such relationships are verified, they can be put together as a new dataset. For example, 3 columns for authentication request, response, and error can be replaced with 2 columns: one column for ratio of authentication response and request indicating the success rate and another column for the ratio of authentication error and request indicating the error rate. Analysis of such verified relationships will lead to insightful revelations about important message types, their relationships and their role in identifying anomalous behaviors.

A further investigation and monitoring of how the "anomalous" devices identified by the current framework behaved in the past and behave at the present time will

provide supporting evidences and will strengthen our understanding of the network and devices.

Exploration of different ways of cluster validation, labelling the clustered datasets and normalizing and / or standardizing data at different phases will yield interesting ideas and results. This thesis applies k-NN classification to classify devices as either "anomalous" or "normal". Exploration of more classification algorithms that can be applied to our datasets will be a fascinating area for future experiments and analyses.

Another area to look into is the influence of the events initiated by cellular network partners and network elements like VLR, and HLR on the SS7 traffic generated by devices. This thesis detected anomalies by analyzing only the communication volumes and external information like past network events was brought in at a later stage. An analysis of the communication volumes of devices along with data representing network elements will yield connected results. A collaborative analysis can shed light on various anomalies concerning the network elements, devices and their combined behaviors. This will help achieve an accurate root cause analysis.

REFERENCES

[1] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything," *CISCO white paper*, pp. 1–11, 2011. [Online]. Available: http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:The+Internet+of+Things+-+How+the+Next+Evolution+of+the+Internet+is+Changing+Everything\#0

[2] Wikipedia, "Machine to machine — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Machine_to_machine&oldid=622707855}

[3] *Signalling System No. 7 (SS7) signalling transport in core network; Stage 3*, 3GPP Std., Rev. 11.0.0, 2012.

[4] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 34, pp. 1302–1325, 2011.

[5] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Distributed Anomaly Detection in Wireless Sensor Networks," in *2006 10th IEEE Singapore International Conference on Communication Systems*, 2006, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4085803

[6] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, 2010.

[7] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta, "Clustering distributed data streams in peer-to-peer environments," *Information Sciences*, vol. 176, pp. 1952–1985, 2006.

[8] H.-b. Wang, Z. Yuan, and C.-d. Wang, "Intrusion Detection for Wireless Sensor Networks Based on Multi-agent and Refined Clustering," in *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, vol. 3, 2009, pp. 450–454.

[9] Z. Yu and J. J. P. Tsai, "A Framework of Machine Learning Based Intrusion Detection for Wireless Sensor Networks," *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pp. 272–279, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4545769

[10] D. H. Garrette, "Unsupervised Learning to Improve Anomaly Detection," 2006. [Online]. Available: http://digitalcommons.iwu.edu/cs_honproj/3

[11] G. Ghidini, S. P. Emmons, F. A. Kamangar, and J. O. Smith, "Advancing M2M communications management: A cloud-based system for cellular traffic analysis," in *Proc. of the 15th IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM)*, 2014.

[12] Wikipedia, "Network switching subsystem — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Network_switching_subsystem&oldid=628516447}

[13] *3rd Generation Partnership Project; Technical Specification Group Core Network; Numbering, addressing and identification*, 3GPP Std., Rev. 3.14.0, 1999.

[14] *Specifications of Signalling System No. 7 Transaction capabilities application part*, 3GPP Std., Rev. Q.773, 1998.

[15] *Mobile Application Part (MAP) specification (3GPP TS 29.002)*, 3GPP Std., Rev. 11.9.0, 2014.

[16] Wikipedia, "Principal component analysis — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=628241372}

[17] "Apache hadoop." [Online]. Available: \url{http://hadoop.apache.org/}

[18] G. Rossum, "Python reference manual," Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 1995.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[20] Wikipedia, "Pig (programming tool) — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Pig_(programming_tool)&oldid=604947835}

[21] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1056489

[22] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org

[23] Wikipedia, "Cluster analysis — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=626905998}

[24] ——, "Anomaly detection — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Anomaly_detection&oldid=614238880}

[25] ——, "Cross-validation (statistics) — wikipedia, the free encyclopedia," 2014. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=617649520

[26] K. Hornik, C. Buchta, and A. Zeileis, "Open-source machine learning: R meets Weka," *Computational Statistics*, vol. 24, no. 2, pp. 225–232, 2009.

[27] Wikipedia, "Hierarchical clustering — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=620515032}

[28] ——, "Cobweb (clustering) — wikipedia, the free encyclopedia," 2014. [Online]. Available: \url{http://en.wikipedia.org/w/index.php?title=Cobweb_(clustering)&oldid=606908855}

[29] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of carefull seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[30] "Tableau." [Online]. Available: \url{http://www.tableausoftware.com/}

[31] "Rapidminer." [Online]. Available: \url{http://rapidminer.com/}

BIOGRAPHICAL STATEMENT

Prathibha Datta Kumar joined the University of Texas at Arlington in spring 2013. She received her B.Tech in Computer Science from Amrita Vishwa Vidyapeetham, Bangalore in 2009.

She worked as a software engineer for 3.5 years in Thomson Reuters and Infosys Technologies Ltd. in India. In the US, she worked as a graduate research assistant at UTA and interned at Intel Corporation.

Her current research interests are in the areas of data analysis / mining, big data technologies, internet of things, graph analytics, distributed computing and software development.