**Abstract**

The project involved performing unsupervised machine learning (clustering) on a data set provided by KYND company. The data set contained information regarding cyber threats faced by the customers of KYND. The process began with literature review to study about various existing tools and algorithms to perform data clustering on a data set containing mixed types of variables and various efficiency evaluation methods that can be used. Hierarchical clustering method, with ward’s linkage produced the most desirable outputs. The final result partitioned the data into 4 clusters. Several variables were then plotted to see if they show different values based on clusters. Finally, the location for organization countries were plotted against the frequency of IPs generated.

**Introduction**

The aim of the project is to use unsupervised learning algorithms, to create clusters of companies based on their risk severity against various cyber-security risks. The project and the relevant resources (containing past 6-month data) is provided by the company KYND. The project involves tailoring the data to get relevant information, applying appropriate clustering algorithm to cluster the data, and plotting the location for the IPs of the organizations involved in the analysis.

**Data Description**

The data contained information generated by the client organizations of KYND company, it included information about domains, versions of those domains, update dates and times, authentication type, operating system, product categories, product classifications etc.

The risks the companies are exposed to are divided into following four categories.

* Domain
* Email
* Certificates
* Services

We also have IP data that contains location information of organizations in our data set.

Relevant Terms

**Domain Name**

Simply speaking, a domain name [36] is a unique identifier for a website and helps in accessing the website. The part including .com, .net, .co etc is called the suffix and is present in all domain names. Domains are alphanumeric and provide an easy to remember method to access websites, instead of remembering the IP addresses.

For example, Google’s domain is ‘google.com’ while 192.168.1.1 is the IP address.

Upon accessing a website, the domain name is converted to it IP address and then read.

**Sub-Domain**

Added to the beginning of a website’s domain [37], it allows additional information while also organizing the information.

**Registrar**

A domain’s registrar [38] manages the right and reservation of the domain names and the associated IP addresses to those domains.

**False Positive**

Whether a false alarm is raised by the system for an error, when in fact there was no error.

Severity:

Threat level of the corresponding risk. Three levels i.e. green, amber, and red with increasing severity from left to right.

**Record Type**

There are two authentication protocols [39], i.e. SPF (Sender Policy Framework) and DKIM (not used in our case). On top of both these protocols, an extra layer of protection, called DMARC (DOMAIN-BASED MESSAGE AUTHENTICATION, REPORTING & CONFORMANCE), which can be used to instruct the servers how to handle unauthenticated emails.

1. **SPF**

It lists the IPs that are authorized to send emails for a domain, if the receiving system has a valid SPF record the system passes the check, otherwise it fails, and the mail can be rejected or put in spam.

1. **DMARC**

DMARC provides policy to treat unauthenticated messages. The policy can be set to one of three options: NONE, QUARANTINE, and REJECT.

Policy = (p=none): no action and message delivered as normal

Policy = (p=quarantine): places the message to spam/junk/quarantine folder

Policy = (p=reject): the message rejected/bounced

The R in DMARC is for the Reporting component of the protocol.

**IP**

IP [40] or internet protocol are rules for sending data over the internet. IP addresses are essentially identifiers as they contain location information of devices.

**Certificate**

A certificate [41] is an encrypted signed document issued by a certifying authority. It contains the public key of the organization and is used to reliably identify that organization. It offers authentication and thus can be used to check if the website being accessed is legitimate.

**Literature Review**

**Unsupervised Machine Learning**

Unsupervised machine learning or clustering, in its simplicity, is the grouping of objects based on similarity of characteristics and the process is called classification. Clustering helps in find out hidden links between our data. From sorting products by their size, category, expiry date, etc in a grocery store, to grouping people showing common symptoms and thereby segregating those with disease from the healthy ones; clustering can be used in various fields of work such as science, engineering, medical, finance etc.

Clustering is done on data where there is no given information about different classes or categories within data and the algorithm finds the similarities on its own. It is different from supervised learning, in a sense that in supervised learning we already know the different categories and then use existing data as a training resource to classify new unlabelled data.

Rokach [1] explains that clustering breaks down the data into subsets by considering the similar patterns and ensuring that each subset has data entries with the highest similarity between them. We can think of clustering structure as a set A, and the operation of clustering gives us subsets A1, A2, A3, … Am, ensuring that the condition:

(1)

This consequently implies that data can be put in only one of the subsets and a data point in one subset cannot be in any other subset.

The absence of labels makes the process of assigning groups to data rather difficult, thus unsupervised learning is more difficult than its supervised counterpart. Coupling that with a data set of high dimensions i.e. large number of rows (entries) and columns (features), makes the process not only time and resource consuming, but also affects the efficiency with which the clustering is being done.

Looking back at Rokach’s [1] explanation, we can see that the whole concept of clustering hinges upon finding the similarity (or rather dissimilarity) between observations. For numerical data, the mostly applied method of finding similarity is to calculate the Euclidian distances between the data, lower distance indicates higher similarity.

**Clustering Methods**

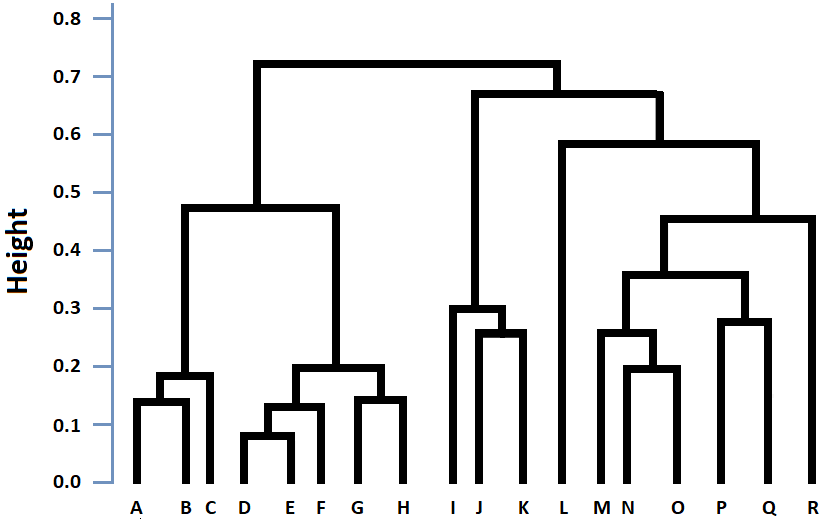
Various clustering techniques, along with their effectiveness, suitability, pros, and cons, will be discussed here. Fra- ley and Raftery [2] divided the clustering approaches into two groups: hierarchical and partitioning.

**Hierarchical Clustering**

Hierarchical clustering can further be done by top-down or bottom-up approach. In the top-down approach, the initial cluster is the set of all data, the process goes on by breaking this set into smaller and smaller clusters, until each observation is its own cluster, this approach is also called divisive clustering. The bottom-up approach is called agglomerative clustering and considers each observation as a separate atomic cluster at start, these cluster are then merged iteratively until there is a single cluster left. In both cases, we can select the desired number of clusters by setting some threshold according to the need of the application.

Hierarchical methods can further be divided depending upon the type of measures they use to find similarities (linkage).

Figure : Dendrogram for hierarchical clustering.



**Agglomerative**

**Divisive**

**Single linkage clustering**

In this type of clustering, the distance between any two clusters is the distance between the closest members of those clusters. Mathematically, the criterion can be stated as [3]:

**Figure 2:** Single linkage clustering.

**Cluster A Cluster B**

(2)

**Complete linkage clustering**

In complete link clustering, the inter-cluster distance for any two clusters is the maximum distance between any two members of those clusters. Mathematically, the criterion is [3]:

(3)

**Figure 3:** Complete linkage clustering.

**Cluster A Cluster B**

**Average linkage clustering**

As the name suggests, average linkage clustering uses the average distance between the members of clusters as the measure of distance between the clusters. Mathematically, the criteria can be stated as [3]:

(4)

**Centroid linkage clustering**

**Figure 4:** Average linkage clustering.

**Cluster A Cluster B**

Centroid clustering [21], as the name suggests, uses the distance between the centroids of two clusters as the inter-cluster distance and merges those clusters which have minimum distance between their centroids. The centroid of k points in n-dimensional space is calculated by as follows [21]:

(5)

where Vi is a vector with n entries for all dimensions of ith ­­data point.

**Figure 5:** Centroid linkage clustering.

**Cluster A Cluster B**

**Ward’s method of clustering**

In this method [20], clusters are merged such that there is a minimum increase in total intra-cluster variance. The algorithm first merges the two clusters in consideration and then find the centroid of the merged cluster, it then finds the squared distances between each point and the centroid and thus calculates the total Sum of Squared Errors (SSE) . The cluster pair that produces the smallest deviation is merged. R provides two options, namely “ward.D” and “ward.D2”. In both methods, clusters with minimum inter-cluster distances are merged. The only difference between the two is that the latter squares the distance to make the differences easily identifiable. Mathematically, the SSE and the cluster merging criterion, respectively, can be stated as [22]:

**Cluster A**

**Cluster B Cluster C**

**Cluster AB Cluster C**

(6) (7)

**Figure 6:** Cluster merging by Ward's method.

**Disadvantages of Hierarchical Clustering**

Hierarchical clustering is often criticized for its sensitivity to outliers and noise. Also an object, after once getting assigned to a cluster, is not used again by the algorithm, and can’t be moved to other clusters. Thus, it’s not possible for the algorithm to correct mistakes made in classification.

They also have the tendency to form spherical shapes and thus misclassify points and thus distort the hierarchical structure [4].

Furthermore, most of the hierarchical algorithms have a complexity of the order O(N2) and due to this limitation, they are not preferred for large scale data sets. However, recent modifications to classical methods can account for this limitation.

**Enhancements to Hierarchical Clustering**

Below are some algorithms that offer enhancements to some of the drawbacks of classical hierarchical algorithms:

1. **BIRCH – Balanced Iterative Reducing and Clustering Using Hierarchies**

BIRCH [5] uses cluster features, which is a triple of number of intra-cluster data objects (n), the linear sum (LS), and the sum of squares (SS) of the attribute values of those objects. As a result, instead of storing all the information in main memory, only the information of features is stored [6]. This makes BIRCH achieve a complexity of O(N) and thus allows it deal with larger data sizes. It is also robust towards outliers.

1. **CURE – Clustering Using Representatives**

CURE [7] also offers robustness to outliers and ability to work with large data. It can also allow clusters of different shapes and sizes to be formed. It is particularly good at handling 2D data. The computation complexity of CURE is O(N2logN).

1. **ROCK – Robust Clustering using links**

ROCK [8] is an agglomerative algorithm that can be used for categorical data. It is mainly developed to use for clustering binary data [19]. It checks the number of links between two records as measure of fit, where links capture similarity between variables. ROCK [3] does not use any distance function and uses random samples to deal with large data.

1. **CHAMELEON**

A hierarchical clustering algorithm [9] that merges clusters only if inter-cluster connectivity and proximity is relatively higher than the intra-cluster connectivity and proximity. It is tested on low dimensional spaces only.

**Partitional Clustering**

In partitional clustering, the data is divided into k-clusters by optimizing a predefined criterion function [10]. Euclidean distance is the most common of such function and assigns a point to a cluster which has the minimum distance from that point.

1. **K-means**

It is one of the most common algorithms used for clustering [10]. In k-means clustering, the number of clusters ‘k’ and their k centroids are first defined. The algorithm then assigns points to a cluster (say cluster f) if they have the minimum distance from the centroid of that cluster. The centroids are then recalculated, and the process repeated until the centroids stop moving. This can be stated mathematically as [3]:

(8)

With k-means there is no defined method for setting initial number of clusters and its accuracy also suffers from noisy data. It also sometimes distorts cluster shapes just to accommodate all points [11].

1. **K-modes**

K-modes [12] is a modification to k-means algorithm that allows the algorithm to cluster categorical data. The main differences between the two algorithms are that the k-modes [13]:

1. Uses a dissimilarity measure that finds the overall dissimilarity between two points by checking the total mismatches of all the attributes of those points.
2. Finds modes of clusters instead of means, which ensures the results are always discrete and represents the most common attributes within the cluster
3. Calculating cluster modes using a frequency-based method (i.e. setting the most common attributes within the cluster as the attributes for the centroid of the cluster)
4. **K-prototype**

K-prototype [33], combines both the k-modes and k-means algorithms to enable clustering of mixed (categorical and numerical) type of data. It is achieved by using a dissimilarity matrix that incorporates the influence of both the numerical and categorical variables present in the data. For instance, let Sn ­be the Euclidean distance matrix for the numerical variables. Similarly for the categorical features, the dissimilarity matrix, checking the frequency of mismatch among categories, be defined as Sc. The dissimilarity matrix for k-prototype algorithm will then be:

where, γ is the weight parameter to balance the effects of the two types of attributes.

According to Huang [34], the k-prototype algorithm processes the data in a similar manner to k-means (except for categorical data, where it uses k-modes approach). Due to the same method of processing, its efficiency is also comparable to k-means.

1. **FCM - Fuzzy c-means**

Unlike the methods discussed above, FCM [11] allows a point to be associated with more than one cluster. The basic idea is that clusters’ boundaries can be overlapped. The degree of overlap can be pre-defined. It is also sensitive to noise and outliers.

**Disadvantages of Partition Based Clustering**

Partitional clustering struggles with data that is not well-separated. It is also user dependent for clusters specification and can often get stuck in local optima which means that the whole algorithm has to run be run with different clusters’ initialization. Specifying clusters manually also make this technique more sensitive to noise and outliers present in the data.

**Similarity Measures**

The whole notion of clustering objects relies upon finding characteristic similarities among objects and then group those with the highest similarity together. However, data sets can vary significantly and often unique with mixed type of variables (numerical, categorical, and ordinal). Consequently, there cannot be a single method to find similarity between the data objects. The extent of similarity between two objects is to find how close those objects are, smaller the distance, higher the similarity. However, because of varying nature of data, we must use different suitable similarity measures. Some similarity measures are discussed below.

***Note:*** *In all distance measures explained below, xi and xj refer to data objects i and j. The same concept can be extended to all data points within the data set.*

**Minkowski Distance**

Minkowski [14] distance includes a group of different measures which calculates distance for numerical data as follows [3]:

(9)

where xik represents value of entity I for the variable k and similarly for xjk (variable k, entity j). For r = 1 [15], the distance becomes Manhattan distance, which calculates the sum of absolute distances between the attributes of points. We get Euclidean distance for r = 2, which calculates the root of sum of square of difference between the attributes of points. numerically. As mentioned before, Euclidean distance does not work well when there is noise in data. Similarly, features with high values can skew the results in their favour [11].

**Cosine Similarity**

Cosine similarity [3] has applications in information retrieval and text mining. It is calculated as the ratio of inner product of two object to the product of magnitude of those of objects. Mathematically,

(10)

Cosine similarity is invariant to rotation but not to linear transformations [11].

**Dice Coefficient**

The measure was given independently by Dice [16] and Sørensen [17]. It is a ratio of the number of common elements of both sets, taken twice, to the sum of their individual number of elements. In this way, it checks for common attributes between two sets to calculate similarity and thus can be used for qualitative data. For vectors, it is defined as [3]:

(11)

**Gower’s Distance**

Gower’s distance [18] caters to the data sets with mixed types of variables. It does so by calculating partial similarities for individual objects and then averaging over them. The gower distance is between 0 and 1, with 0 indicating that the two objects are identical and 1 showing maximum dissimilarity. Gower’s distance uses different metrics depending upon data.

* For numerical data, normalized Manhattan distance is used.
* Same distance metric is used for ordinal data; however, the variable is ranked beforehand. Further rules are included to handle ties.
* In case of nominal data, the number of k categories of the variable are split into k binary columns and then it uses the Dice coefficient to find the distance.

Mathematically, it can be represented as :

(12)

Where sj(x1,x2) is the partial similarity calculated between each attribute of object i and j.

**Evaluation Criteria**

In real world applications, in addition to creating clusters, it is also important to test whether the clusters formed are efficient and thus how many clusters would give the best results. Generally, good clustering makes sure that the clusters formed have maximum intra-cluster similarity and minimum inter-cluster similarity. In other words, clusters formed should have only those objects within them that are very similar to each other and there should be significant difference from objects of other clusters. The evaluation criteria can be divided into internal and external validation, respectively.

**Internal Validation Measures**

Internal measures rely on the information already present in the data and evaluates goodness of clusters formed by checking intra-cluster similarity and inter-cluster dissimilarity or a combination of both. It relies on three [23] basic principles of compactness, connectedness, and separation.

**Compactness**

Compactness [23] measures the closeness of objects within a cluster. A cluster is compact if it has lower intra-cluster variation and it is, therefore, a good cluster.

**Separation**

Separation [23] evaluates how distinct the boundaries between clusters are. In other words, it checks for how distant clusters are from each other.

**Connectivity**

Connectivity [23] checks for how many nearest neighbours from the data set are put together in the same cluster. Its value starts from 0 and goes to infinity. Lower values yield effective clustering results.

The commonly used internal validation indices are some weighted combination of the separation and compactness measures. A general formula with α and β as weights can be:

(13)

Discussed below are some common internal validation indices:

**Silhouette Width**

It [23] is used to measure the efficiency of formed clusters. The silhouette width, for every observation, is calculated by the following steps:

* + 1. For an observation i, the algorithm first calculates the dissimilarities between i and all other observations within the cluster X (cluster containing observation i). The average of those dissimilarities ai is then calculated.
    2. The algorithm then finds the nearest neighbour cluster to cluster X. It does so by finding average dissimilarity d(i,C) of i from all other clusters *C* in the same manner as in the previous step. The cluster with the smallest of average dissimilarity bi=mincd(i,C) is deemed as the nearest neighbour cluster to cluster X, and bi is the dissimilarity of i from its nearest neighbouring cluster.
    3. In the last step, the silhouette width si is calculated for the ith observation as:

Silhouette width is a ratio and ranges between -1 to 1. Larger values indicate well-formed clusters. Observations that lie between multiple clusters give a value of Si closer to 0, while negative values indicate that the observations are put in a different cluster than what they should be in.

**Dunn Index**

Dunn index [23] is another common internal validation criteria and is calculated as:

* + 1. For a cluster, the algorithm calculates the distance between each data point within that cluster to the points in other clusters. The process is repeated for all clusters.
    2. The smallest of these distances is considered as the minimum separation between the chosen cluster and the remaining clusters.
    3. The distance for each pair within the cluster is then computed. This is done for each cluster present in the data.
    4. The largest value among these distances is taken as the compactness (maximum diameter) of objects within the cluster.
    5. Dunn Index (D) is computed by the following formula:

Well-formed clusters are supposed to have objects that are close to each other and thus have small maximum diameter. Similarly, objects in good clusters are well separated from objects in other clusters, resulting in high minimum separation. As a result, Dunn index should be maximized to achieve efficient clustering.

Some other internal validation metrics include:

**Sum of Squared Errors**

One of the commonly used criteria [21] for internal validation is the sum of squared errors (SSE). It is the sum of squared distances between the centroid of each cluster and all the objects within the cluster, summed over all clusters. Mathematically [3],

(13)

where *k* refers to the number of clusters, *nk* is the number of objects within cluster *k* and *uk* is the centroid/mean of cluster *k*.

**Scatter Criteria**

The scatter criteria [1] are the scatter matrices that are used to evaluate the intra-cluster scatter, the inter-cluster scatter, and the total scatter (sum of the first two). For a cluster k (Ck­), scatter matrix is given as [1]:

The sum of Sk over all clusters is the intra-cluster scatter matrix:

and the inter-cluster scatter matrix is given as:

Similar to Sk, the total scatter matrix S­t ­will be:

where µ is the average of all k clusters’ means µk.

**External Validation Measures**

External measures [1] use predefined labels of the data to evaluate how accurate results the clustering algorithm produced by matching predicted labels with the true ones. When true labels are available, external validation provides an insight towards the efficiency of clustering algorithms used.

Before we discuss the metrics, we must understand the following terms:

**True positives (TP)**

True positives [3] indicate the total number of positive data values which are also predicted accurately as positives by the clustering algorithm.

**True negatives (TN)**

True negatives [3] indicate the total number of negative data values which are also predicted accurately as negatives by the clustering algorithm.

**False positives (FP)**

False positives [3] indicate the total number of negative data values which are predicted incorrectly as positives by the clustering algorithm.

**False negatives (FN)**

False negatives [3] indicate the total number of positive data values which are predicted incorrectly as negatives by the clustering algorithm.

**Precision (P)**

Precision (P) [24] is the ratio of accurately predicted positive values to the total predicted positive values. It is an indicator of how well the algorithm is identifying true positives. Mathematically [24],

The metrics for external validation are as follows:

**Recall (R)**

Recall (R) [25] is the ratio of accurately predicted positive values to the total positive values present in the data. It shows the proportion of positive values the algorithm was able to identify. It is given by [25],

**Rand Index**

Rand index [26] compares the clusters (or a cluster with the other clusters combined) formed with the true classifications to measure the similarity between predicted and true clusters. Rand index ranges from 0 to 1, with 1 indicating a perfect partition between two clusters (or a cluster compared to other clusters combined). Mathematically [3],

**F-measure**

Rand index [27] may produce undesirable results for some clustering problems, as false negatives and false positives are treated equally. F-measure overcomes this drawback by adding weight n (eta) to the precision and recall rates. The value ranges between 0 and 1, with 1 showing perfect clustering. The equation for F-measure is [3],

**Jaccard index**

It [28] measures the similarity between two data sets and does so by dividing the unique elements common to both sets to the sum of unique elements present in those sets. The value of Jaccard index ranges from 0 to 1, with higher values indicating greater similarity between sets A and B. It is given by the following formula [3]:

**Fowlkes-Mallows index**

The Fowlkes-Mallows index [29] compares the similarity among different clusters formed by a clustering algorithm. Similar to Jaccard index, Folkes-Mallows index outputs higher values when there is higher similarity among compared clusters. Mathematically, it is given by [3]:

**Confusion matrix**

A confusion matrix [30], also known as a contingency table, is a tool that helps in understanding the efficiency of clustering algorithm. It does so by checking the labels of the clusters formed and comparing it with true labels. The results are then visualized to show the count and proportion of the correct and incorrect outputs generated.

Saxena [3] explains that if a clustering algorithm is trained to classify shapes as either circle, square, or triangle, plotting the confusion matrix will provide a visual summary of the algorithms result. A confusion matrix for a sample of (say) 40 shapes, with 18 circles, 15 squares, and 7 triangles will be similar to the table below. The output of the confusion matrix can be used for testing the algorithm’s performance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Predicted Group**  **True Group** | **Circle** | **Square** | **Triangle** |
| **Circle** | 10 | 6 | 2 |
| **Square** | 3 | 11 | 1 |
| **Triangle** | 0 | 2 | 5 |

Both internal and external validation metrics provide insight to the outcomes of the clustering process. Internal validation helps in finding the parameters that produce the optimum outcomes for the problem at hand without knowing the true information about the data. The importance of using external validation tools lies in their ability to test different clustering algorithms to find the algorithms that produce the best results for a specific data set. Further details about evaluation method can be found in [31,32]

**Pre-Processing**

The project had two main objectives:

One was to perform unsupervised clustering on the KYND company’s cyber profile dataset to find hidden patterns within the data that links organizations together based on the types of risks.

***Note:*** *Starting from this point, the term data / data set / original data set all refers to the data set provided by KYND company.*

The second goal was to conduct a visual analysis of the locations of KYND company’s clients, obtained through the organizations’ IP addresses.

For objective 1 of the project, the first step involved cleaning the data set and transforming it into a form that can be read and processed by machine learning algorithms. To achieve the above task, all the information about different types of risks (Domain, Emails, Certificates, and Services) were merged into a single long form sheet.

The next step was variable selection. The variables that appeared more relevant towards explaining the patterns were selected, and the variables with either too many categories or irrelevant or duplicate information were ignored.

Afterwards, the variable “domain\_name”, containing information about the different websites used by each organization, was replaced with variable “domain” that contained just the top-level domain part (i.e .com or .gov.uk etc).

The variable “updated\_at”, showing the date of the update, was transformed into variable “update\_date” which showed the quadrant of the update and had six factor levels (Q1, Q2, Q3, Q4, Q5 and Q6) for each quadrant starting from January 2021 till June 2022. The “time” variable was also transformed into a factor variable “update\_time” with three levels dividing the day into three 12 hours period. Level “Morning” was from 4:00 to 11:59, “Evening” from 12:00 to 19:59, and “Night” from 20:00 to 3:59, respectively. This merged from contained 14 relevant variables and 594388 rows (i.e. 8321432 cells), with the variable “org\_id” as the organization identifier. All the above steps weredone onMs Excel.

The rest of the process was carried out on R-Studio. First installing packages and loading libraries required to perform and analyse clustering.

1. install.packages("tidyverse")
2. install.packages("cba")
3. install.packages("leaflet")
4. install.packages("data.table")
5. install.packages("clustMixType")
6. install.packages("ClustTools")
7. install.packages("NbClust")
8. install.packages("writexl")
9. install.packages("factoextra")
10. install.packages("devtools")
11. install.packages("caret")
12. install.packages("ClusterR")
13. install.packages("magittr")
14. install.packages("dandextend")
16. load\_packages <- function() {
17. #if(!require(devtools))
18. #devtools::install\_github("kassambara/factoextra")
19. #remotes::install\_github("jhmadsen/ClustTools")
20. library(remotes)
21. library(data.table)
22. library(leaflet)
23. library(tidyverse)
24. library(readxl)
25. library(cluster)
26. library(cba)
27. library(clValid)
28. library(clustMixType)
29. library(ClustTools)
30. library(writexl)
31. library(factoextra)
32. library(maps)
33. library(gridExtra)
34. library(caret)
35. library(ClusterR)
36. library(magrittr)
37. library(dendextend)
38. }

After this, the data set was loaded into R-Studio.

1. data1 <- read\_excel("KYND\_data\_history.xlsx", sheet = 1, col\_types=c('text', 'text', 'text', 'numeric', 'text', 'text', 'text', 'text', 'logical', 'text', 'text', 'text', 'text', 'text'))
2. head(data1)

It had the following form.

# Console Output:

A tibble: 6 x 15

org\_id domain recor~1 version updat~2 updat~3 time updat~4 false~5 issuer

<fct> <fct> <fct> <fct> <chr> <fct> <chr> <fct> <dbl> <fct>

1 01d90a3~ ca NA 1 44218 Q1 0.64~ Evening 0 NA

2 01d90a3~ ca NA 2 44492 Q4 0.68~ Evening 0 NA

3 01d90a3~ ca NA 1 44218 Q1 0.72~ Evening 0 NA

4 01d90a3~ ca NA 2 44492 Q4 0.76~ Evening 0 NA

5 01d90a3~ ca NA 1 44369 Q2 0.80~ Evening 0 NA

6 01d90a3~ ca NA 2 44492 Q4 0.85~ Night 0 NA

# ... with 3 more variables: risk\_type <fct>, severity <fct>,

# severity\_num <int>, and abbreviated variable names 1: record\_type,

# 2: updated\_at, 3: update\_date, # 4: update\_time, 5: false\_positive,...

The data type of the categorical variables was changed to factors. The variable “false\_positive” was converted into a numerical form with “1” when “false\_positive” equalled TRUE and “0" for FALSE. The variable “severity” was transformed into “severity\_num” which converted severity levels from “green”, “amber”, and “red” to numerical factors of “1”, “2”, and “3”, respectively. Finally, the old forms of the transformed variables were removed. The code to perform above steps is given below:

1. data1 <- read\_excel("KYND\_data\_history.xlsx", sheet = 1, col\_types=c('text', 'text', 'text', 'numeric', 'text', 'text', 'text', 'text', 'logical', 'text', 'text', 'text', 'text', 'text'))
2. data1$org\_id <- as.factor(data1$org\_id)
3. data1$domain <- as.factor(data1$domain)
4. data1$record\_type <- as.factor(data1$record\_type)
5. data1$false\_positive <- as.factor(data1$false\_positive)
6. data1$issuer <- as.factor(data1$issuer)
7. data1$version <- as.factor(data1$version)
8. data1$product\_category <- as.factor(data1$product\_category)
9. data1$product\_classification <- as.factor(data1$product\_classification)
10. data1$risk\_type <- as.factor(data1$risk\_type)
11. data1$severity <- factor(data1$severity, levels = c('green', 'amber', 'red'))
12. data1$update\_date <- as.factor(data1$update\_date)
13. data1$update\_time <- as.factor(data1$update\_time)
14. data1$false\_positive <- as.numeric(data1$false\_positive == TRUE)
15. data1$severity\_num <- as.integer(data1$severity)
16. dm <- data1[,c(-5,-7,-14)]
17. dm <- as.data.table(dm)

The huge size of data set meant that the distance matrix that needs to be created to perform cluster would have a dimension of 594388\*594388. This large data set meant that any operation performed would be very resource consuming. Thus, additional transformation of data was required. The goal now was to consolidate the data in a form where each organization only had a single row entry and all relevant information about that organization would be encompassed in that single row.

The data was first grouped by the organizations while converting the information about each risk’s minimum severity, maximum severity, and count to separate variables. This wide form transformation of the data reduced it to a format with 100 row and 154 columns (variables). As a result the distance matrix now created would be a 100\*100 matrix, which reduced the computation time and made distance calculation possible without increasing computational resources. The data type of the newly created categorical variables was also changed to factors. The processed data was stored in “final\_data”. The code is as follows:

1. dm1 <- dm %>%
2. group\_by(org\_id) %>%
3. summarise(domains = sum(!is.na(unique(domain))),
4. record\_types = sum(!is.na(unique(record\_type))),
5. versions = sum(!is.na(unique(version))),
6. update\_dates = sum(!is.na(unique(update\_date))),
7. update\_times = sum(!is.na(unique(update\_time))),
8. false\_positive\_ratio = sum(false\_positive, na.rm = TRUE)/sum(false\_positive == 0, na.rm = TRUE),
9. issuers = sum(!is.na(unique(issuer))),
10. product\_categories = sum(!is.na(unique(product\_category))),
11. product\_classifications = sum(!is.na(unique(product\_classification)))
12. )
14. d2 <- dm[,c(1,11,12)]
16. d3 <- d2 %>%
17. group\_by(org\_id, risk\_type) %>%
18. summarise(count = n(),
19. min\_sev = min(severity\_num),
20. max\_sev = max(severity\_num)) %>%
21. group\_by(org\_id, risk\_type) %>%
22. nest() %>%
23. pivot\_wider(names\_from = risk\_type, values\_from = data) %>%
24. unnest(cols = everything(), names\_sep = "\_")

27. final\_data <- left\_join(dm1, d3, by = "org\_id")
28. names <- c(5,12,13,15,16,18,19,21,22,24,25,27,28,30,31,33,34,36,37,39,40,42,43,45,46,48,49,51,52,54,55,57,58,60,61,63,64,66,67,69,70,72,73,75,76,78,79,81,82,84,85,87,88,90,91,93,94,96,97,99,100,102,103,105,106,108,109,111,112,114,115,117,118,120,121,123,124,126,127,129,130,132,133,135,136,138,139,141,142,144,145,147,148,150,151,153,154)
29. final\_data[,names] <- lapply(final\_data[,names] , factor)

The new structure of data can be seen in figure 0342.

1. head(final\_data)

# Console Output:

# A tibble: 6 x 154

org\_id domains recor~1 versi~2 updat~3 updat~4 false~5 issuers produ~6 produ~7

<fct> <int> <int> <int> <int> <int> <dbl> <int> <int> <int>

1 01d90~ 2 2 10 5 3 0 0 5 9

2 03023~ 6 2 13 6 3 20.6 4 4 10

3 03564~ 1 2 20 6 3 0 3 7 13

4 0ed32~ 1 2 16 5 3 0 5 4 16

5 0f35f~ 1 2 16 4 3 0 6 5 15

6 106ff~ 10 2 15 5 3 0 29 12 39

# ... with 142 more variables: rag\_email\_dmarc\_not\_present\_max\_sev <int>,

# rag\_email\_spf\_not\_present\_count <int>,

# rag\_email\_spf\_not\_present\_min\_sev <int>,

# rag\_email\_spf\_not\_present\_max\_sev <int>, ...

**Clustering**

With the data transformed to a readable format, it was now time to apply different clustering algorithms. As the data set contained both numerical and categorical variables, k-prototype, ROCK and hierarchical (with Gower’s distance for mixed data) clustering algorithms were shortlisted.

**K-prototype**

The problem faced with using k-prototype algorithm is that it cannot work with missing values in the data and removes entire rows with any missing value. As the data set contained missing values in all rows, the algorithm failed to perform the clustering operation. The code and output below show this.

1. set.seed(007)
2. pclust <- kproto(final\_data [-1], k=3, nstart = 10) # Cant' work with NAs

# Console Output:

100 observation(s) with NAs.

Observations with NAs are removed.

Error in kproto.default(final\_data[-1], k = 3, nstart = 10) :

Data frame has less complete observations than clusters!

**Simulation**

Before using the remaining algorithms, it was important to check which of those two were producing better outputs for the data set. To check this, a simulated data, similar to the original data set, was generated. It had six variables, two being numerical and three as categorical. The “var2” and “var5” were added as categorical and numerical noise, respectively, and did not affect clustering. The sixth variable contained true cluster labels for the simulated data. The condition for creating the simulated data and its clusters can be seen by the following code.

1. set.seed(123)
2. size <- 100
4. var1 <- c(sample(c("1", "2", "3"), size = size, replace = TRUE, prob = c(1, 1, 1)),
5. sample(c("1", "2", "3"), size = size, replace = TRUE, prob = c(1, 2, 3)),
6. sample(c("1", "2"), size = size, replace = TRUE, prob = c(1, 1)),
7. sample(c("1", "2"), size = size, replace = TRUE, prob = c(3, 1)),
8. rep("3", cluster\_size),
9. sample(c("2", "3"), size = size, replace = TRUE, prob = c(2, 1))
10. )
11. var2 <- sample(c("1", "2", "3"), size = 6 \* size, replace = TRUE) # Factorial noise
12. var3 <- c(rnorm(size, 3.5, 2), rnorm(size, 7), rnorm(size, 10, 1.2),
13. rnorm(size, 15, 2), rnorm(size, 20,2.5), rnorm(size, 28, 3))
14. var4 <- c(rep("2", size),
15. sample(c("1", "2", "3"), size = size, replace = TRUE, prob = c(1, 1, 2)),
16. sample(c("1", "2"), size = size, replace = TRUE, prob = c(2, 1)),
17. rep("1", size),
18. sample(c("1", "3"), size = size, replace = TRUE, prob = c(1, 3)),
19. sample(c("1", "3"), size = size, replace = TRUE, prob = c(1, 1))
20. )
21. var5 <- c(rexp(6\*size)) # Continuous noise
23. clusters <- rep(1:6, each = size)
24. sim\_data <- data.frame(var1, var2, var3, var4, var5, clusters)
25. head(sim\_data)

# Console Output:

var1 var2 var3 var4 var5 clusters

1 2 2 5.038105 2 0.5543285 1

2 1 2 3.741439 2 0.4342878 1

3 3 3 5.227297 2 0.3263524 1

4 1 3 6.261029 2 0.4061840 1

5 1 3 7.432496 2 1.9273811 1

6 2 3 3.443210 2 0.1961564 1

Following this, bar plots for categorical variables and density plots for continuous variables were generated for the simulated data to check if the clusters had the assigned differences.

1. # Bar plots for categorical variables
3. ggplot(sim\_data) +
4. theme\_classic() +
5. geom\_bar(aes(fill = var1, x = clusters))
7. ggplot(sim\_data) +
8. theme\_classic() +
9. geom\_bar(aes(fill = var2, x = clusters))
11. ggplot(sim\_data) +
12. theme\_classic() +
13. geom\_bar(aes(fill = var4, x = clusters))
14. # Density plots for continuous variables
16. ggplot(sim\_data) +
17. theme\_classic() +
18. geom\_density(aes(color = factor(clusters), x = var3), lwd = 0.8))
20. ggplot(sim\_data) +
21. theme\_classic() +
22. geom\_density(aes(color = factor(clusters), x = var5), lwd = 0.8))

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Chart, line chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

**ROCK algorithm**

With the simulated data created and authenticated, the remaining clustering algorithms were compared. First, the distance matrix was created, as the simulated data set contained mixed variables, Gower’s distance matrix was used. After that ROCK algorithm was applied to the simulated data. The ROCK algorithm [35] has two coupled parameters, i.e. the neighbourhood parameter “theta” and the distance threshold “beta” (1 – theta). Consequently, high values of theta (ranging from 0 to 1) more common neighbours are required for every data pair. Also, only the points with distance less than or equals to beta, are considered as neighbours. Hence, multiple values of theta were tried as follows.

1. rownames(sim\_data) <- c(1:600)
2. sim\_data$var1 <- as.factor(sim\_data$var1)
3. sim\_data$var2 <- as.factor(sim\_data$var2)
4. sim\_data$var4 <- as.factor(sim\_data$var4)
6. sim\_dist <- dist(sim\_data[-6], method = "gower")
8. sim\_rc <- rockCluster(as.matrix(sim\_dist), n=6, theta = 0.33, debug=TRUE)
9. # Same result for theta ≤ 0.33
10. print(sim\_rc)
11. rf <- fitted(sim\_rc)
12. table(sim\_data$clusters, rf$cl)
13. sim\_rc2 <- rockCluster(as.matrix(sim\_dist), n=6, theta = 0.34, debug=TRUE)
14. # Same result for theta ≥ 0.34
15. print(sim\_rc2)
16. rf2 <- fitted(sim\_rc2)
17. table(sim\_data$clusters, rf2$cl)

For theta ≤ 0.33, the algorithm was only assigning values to cluster 5 and 6, which was not the intended result.

# Console Ouptut:

1 2 3 4 5 6

1 0 0 0 0 45 55

2 0 0 0 0 50 50

3 0 0 0 0 52 48

4 0 0 0 0 54 46

5 0 0 0 0 48 52

6 0 0 0 0 47 53

For theta ≥ 0.33, the ROCK algorithm performed even worst and assigned all the values to cluster 1. It was evident from the results that ROCK clustering algorithm was not suitable for a data set with mixed variable.

# Console Output

1

1 100

2 100

3 100

4 100

5 100

6 100

Furthermore, the lack of testing methods available for the output of ROCK algorithm adds to the limitations it has.

**Hierarchical Algorithms**

Finally, different hierarchical clustering algorithms were applied to check they efficiency towards the simulated data (and thus the original data). Different linkage methods i.e. “ward”, “complete”, “single”, “average”, and “centroid” were used. To compare performance of these algorithms, functions were created to plot dendrogram, apply various internal and external validation methods, such as Silhouette width, Rand index, confusion matrix etc. The code below creates the functions.

1. internal\_validation <- function(my\_cluster, k, my\_distance){
3. dend <- my\_cluster %>% as.dendrogram %>% ### USE THIS FOR DENDROGRAMS
4. set("branches\_k\_color", k = k) %>% set("branches\_lwd", 0.7) %>%
5. set("labels\_cex", 0.6) %>% set("labels\_colors", k = k) %>%
6. set("leaves\_pch", 19) %>% set("leaves\_cex", 0.5) %>% plot(horiz = FALSE)
7. plot(silhouette(cutree(my\_cluster, k), my\_distance), col=1:k, border=NA)
8. DI <- dunn(distance = my\_distance, cutree(my\_cluster, k))
9. print(paste(“Dunn index is “, round(DI, 5)))
10. }
12. confusion\_matrix <- function(my\_data, my\_cluster, k){
13. truth <- as.factor(my\_data$clusters)
14. pred <- as.factor(cutree(my\_cluster, k))
15. confusionMatrix(pred, truth)
16. table <- data.frame(confusionMatrix(pred, truth)$table)
18. plotTable <- table %>%
19. mutate(goodbad = ifelse(table$Prediction == table$Reference, "correct", "incorrect")) %>%
20. group\_by(Reference) %>%
21. mutate(prop = Freq/sum(Freq))
23. ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad, alpha = prop)) +
24. geom\_tile() +
25. geom\_text(aes(label = Freq), vjust = .5, fontface = "bold", alpha = 1) +
26. scale\_fill\_manual(values = c(correct = "green", incorrect = "red")) +
27. theme\_bw() +
28. xlim(rev(levels(table$Reference)))
29. }

**Ward linkage**

Clustering using ward linkage and its performance testing is done by code below.

1. sim\_clust <- hclust(sim\_dist, method = "ward.D2")
2. internal\_validation(sim\_clust, 6, sim\_dist)
3. confusion\_matrix(sim\_data, sim\_clust)
4. Chart

   Description automatically generatedexternal\_validation(sim\_data$clusters, cutree(sim\_clust,6), summary\_stats = TRUE)
5. Calendar

   Description automatically generated

From the output, it can be seen that the silhouette width is 0.37, the accuracy from confusion matrix is 42.50% (255/600), the remaining results are present below.

# Console Output:

----------------------------------------

“Dunn index is 0.33344”

----------------------------------------

# External validation metrics

----------------------------------------

purity : 0.5467

entropy : 0.506

normalized mutual information : 0.4554

variation of information : 2.7451

normalized var. of information : 0.7052

----------------------------------------

specificity : 0.8626

sensitivity : 0.4782

precision : 0.4081

recall : 0.4782

F-measure : 0.4404

----------------------------------------

accuracy OR rand-index : 0.7991

adjusted-rand-index : 0.3189

jaccard-index : 0.2824

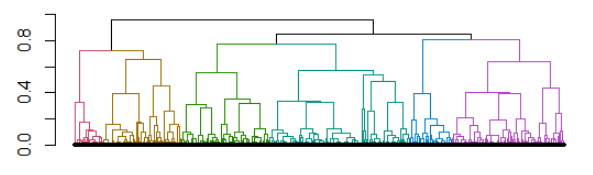
fowlkes-mallows-index : 0.4418

mirkin-metric : 72202

----------------------------------------

**Complete linkage**

The code for clustering using complete linkage and its performance testing is as follows.

1. sim\_clust2 <- hclust(sim\_dist, method = "complete")
2. internal\_validation(sim\_clust2, 6, sim\_dist)
3. confusion\_matrix(sim\_data, sim\_clust2, 6)
4. external\_validation(sim\_data$clusters, cutree(sim\_clust2,6), summary\_stats = TRUE)

Chart

Description automatically generated

The results are as follows, the silhouette width is 0.31, the accuracy from confusion matrix is 25.50% (153/600), and the remaining results are given below.

Calendar

Description automatically generated

# Console Output:

----------------------------------------

“Dunn index is 0.3041”

----------------------------------------

# External validation metrics

----------------------------------------

purity : 0.4983

entropy : 0.5656

normalized mutual information : 0.3797

variation of information : 3.0987

normalized var. of information : 0.7657

----------------------------------------

specificity : 0.8391

sensitivity : 0.4166

precision : 0.3389

recall : 0.4166

F-measure : 0.3737

----------------------------------------

accuracy OR rand-index : 0.7693

adjusted-rand-index : 0.2342

jaccard-index : 0.2298

fowlkes-mallows-index : 0.3757

mirkin-metric : 82922

----------------------------------------

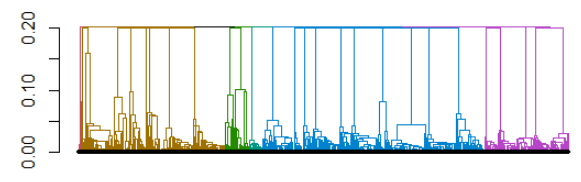
**Single linkage**

Similarly, for single linkage.

1. sim\_clust3 <- hclust(sim\_dist, method = "single")
2. internal\_validation(sim\_clust3, 6, sim\_dist)
3. confusion\_matrix(sim\_data, sim\_clust3, 6)
4. external\_validation(sim\_data$clusters, cutree(sim\_clust3,6), summary\_stats = TRUE)

Calendar

Description automatically generatedChart

Description automatically generated

The results for single linkage clustering gives a silhouette width of 0.14, the accuracy from confusion matrix is 23.50% (141/600), which is very low. The remaining results are as follows.

# Console Output:

----------------------------------------

"Dunn index is 0.23486"

----------------------------------------

# External validation metrics

----------------------------------------

purity : 0.4967

entropy : 0.3674

normalized mutual information : 0.4072

variation of information : 2.6309

normalized var. of information : 0.7443

----------------------------------------

specificity : 0.7348

sensitivity : 0.624

precision : 0.3178

recall : 0.624

F-measure : 0.4211

----------------------------------------

accuracy OR rand-index : 0.7165

adjusted-rand-index : 0.2588

jaccard-index : 0.2667

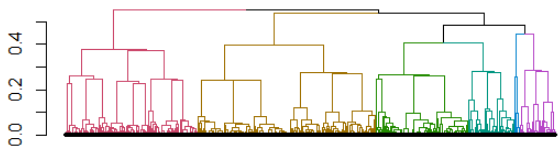
fowlkes-mallows-index : 0.4453

mirkin-metric : 101902

----------------------------------------

**Average linkage**

The clustering and validation for average linkage is performed by following code.



For average linkage clustering, the silhouette width is 0.32 and from confusion matrix, the accuracy comes out to be 22.33% (134/600). Other results can be seen below.

Calendar

Description automatically generatedChart

Description automatically generated

# Console Output:

----------------------------------------

"Dunn index is 0.28018"

----------------------------------------

# External validation metrics

----------------------------------------

purity : 0.5583

entropy : 0.3967

normalized mutual information : 0.4801

variation of information : 2.4702

normalized var. of information : 0.6842

----------------------------------------

specificity : 0.8156

sensitivity : 0.6

precision : 0.3918

recall : 0.6

F-measure : 0.474

----------------------------------------

accuracy OR rand-index : 0.7799

adjusted-rand-index : 0.3426

jaccard-index : 0.3106

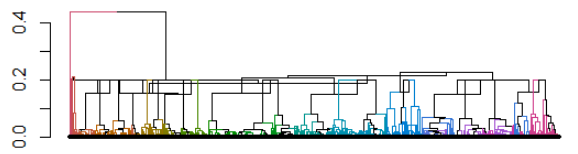
fowlkes-mallows-index : 0.4848

mirkin-metric : 79086

----------------------------------------

**Centroid linkage**

Lastly, for centroid linkage, the code below clusters the data and tests for efficiency.

1. sim\_clust5 <- hclust(sim\_dist, method = "centroid")
2. internal\_validation(sim\_clust5, 6, sim\_dist)
3. confusion\_matrix(sim\_data, sim\_clust5, 6)
4. external\_validation(sim\_data$clusters, cutree(sim\_clust5,6), summary\_stats = TRUE)

Chart

Description automatically generated

Calendar

Description automatically generated

The output of centroid linkage gives a silhouette width of 0.27 and an accuracy of 19.50% (117/600), calculated from confusion matrix. Remaining metrics are given below.

# Console Output:

----------------------------------------

"Dunn index is 0.24228"

----------------------------------------

# External validation metrics

----------------------------------------

purity : 0.5633

entropy : 0.3048

normalized mutual information : 0.5079

variation of information : 2.2247

normalized var. of information : 0.6596

----------------------------------------

specificity : 0.793

sensitivity : 0.6849

precision : 0.3958

recall : 0.6849

F-measure : 0.5017

----------------------------------------

accuracy OR rand-index : 0.7751

adjusted-rand-index : 0.3697

jaccard-index : 0.3348

fowlkes-mallows-index : 0.5207

mirkin-metric : 80816

----------------------------------------

From the results, we can see that, compared to other methods, Ward method produces relatively better results in terms of internal validation, with highest average silhouette width and Dunn Index. For external measures, the confusion matrix gives the highest accuracy for ward’s method. Centroid linkage gives the highest values for F-measure and adjusted Rand index; however, it also has the highest Fowlkes-mallows index, indicating the clusters formed have higher similarity with each other. These results can be seen from tables above. Both ward’s method and centroid linkage are selected to be applied to the original data set.

**Ward’s method**

The distance matrix using Gower’s method was first created. It was followed by creating clusters using the ward’s linkage method. A height to cluster plot was then created to check the optimal number of clusters to keep in the data and the result indicated to keep either 4 or 5 clusters. Next step was to apply internal validation metrics to see whether 4 or 5 clusters produce better results. All these steps are performed as follows.

1. distances <- dist(final\_data[-1], method = "gower")
2. clust\_ward <- hclust(distances, method = "ward.D2")
4. ggplot(clust\_ward$height %>%
5. as\_tibble() %>%
6. add\_column(groups = length(clust\_ward$height):1) %>%
7. rename(height=value),
8. aes(x=groups, y=height)) +
9. geom\_point() +
10. geom\_line()
12. internal\_validation(clust\_ward, 4, distances)
13. Chart, line chart

    Description automatically generatedinternal\_validation(clust\_ward, 5, distances)

Chart

Description automatically generatedChart, histogram

Description automatically generated

# Console Output:

For 4 clusters: "Dunn index is 0.10813"

For 5 clusters: "Dunn index is 0.12951"

The silhouette index gives better results for 4 clusters while the Dunn index is better for 5 clusters. However, upon testing, it was found that the results of Dunn index were biased towards increasing number of clusters, with highest value of 1.239 for 98 clusters. Hence, silhouette width was given preference and 4 clusters were selected for ward’s method.

**Centroid method**

Similar to ward’s method, the clustering algorithm with centroid linkage was applied to the data set and cluster to height plot was generated. The graph showed significant height drop at 2 and 7 clusters. Internal validation was used to compare the output of 2 clusters with that of 7 clusters. Here is the relevant code.

1. clust\_centroid <- hclust(distances, method = "centroid")
3. ggplot(clust\_centroid$height %>%
4. as\_tibble() %>%
5. add\_column(groups = length(clust\_centroid$height):1) %>%
6. rename(height=value),
7. aes(x=groups, y=height)) +
8. geom\_point() +
9. geom\_line()
11. internal\_validation(clust\_centroid, 2, distances)
12. internal\_validation(clust\_centroid, 7, distances)
13. Chart, histogram

    Description automatically generatedChart

    Description automatically generated

Both the silhouette width and Dunn index for 2 clusters case are better than that of 7 clusters scenario. Hence, for centroid linkage method, the output with two clusters was given preference.

# Console Output:

For 2 clusters: "Dunn index is 0.9245"

For 7 clusters: "Dunn index is 0.2167"

The final step was to compare the two short-listed outputs from ward’s and centroid linkage.

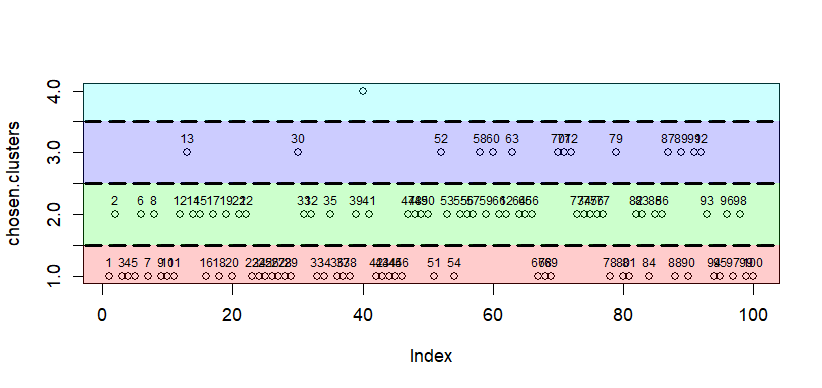
While the 2 clusters partition of the data, formed by centroid linkage, gives better results for both silhouette width and Dunn index, the clusters formed are not very informative. The centroid linkage groups all the observations except organization number 40 (org\_id = 630eee90-67a1-11eb-9d98-6ba16061e7ed) into one cluster, and thus this clustering outcome does not give much insight about the hidden patterns within data. It can be seen below.

1. chosen.clusters <- cutree(clust\_centroid, 2)
2. plot(chosen.clusters)
3. rect(-10,0,110,1.5, col= rgb(1,0,0, alpha = 0.2), border = "black", lty= 2, lwd= 3)
4. rect(-10,1.5,110,3, col= rgb(0,1,0, alpha = 0.2), border = "black", lty= 2, lwd= 3)
5. Chart

   Description automatically generatedtext(chosen.clusters, labels = names(chosen.clusters), cex = 0.7, pos = 3)

The output of ward’s method makes more sense. It groups data into 4 clusters. It is important to note that this method also put the organization 40 (org\_id = 630eee90-67a1-11eb-9d98-6ba16061e7ed) into a separate cluster, indicating that the observation is significantly different from the rest of the data set. The code is as follows.

1. chosen.clusters <- cutree(clust\_ward, 4)
2. plot(chosen.clusters)
3. rect(-10,0,110,1.5, col= rgb(1,0,0, alpha=0.2), border="black", lty= 2, lwd= 3)
4. rect(-10,1.5,110,2.5, col= rgb(0,1,0, alpha=0.2), border="black", lty= 2, lwd= 3)
5. rect(-10,2.5,110,3.5, col= rgb(0,0,1, alpha=0.2), border="black", lty= 2, lwd= 3)
6. rect(-10,3.5,110,4.5, col= rgb(0,1,1, alpha=0.2), border="black", lty= 2, lwd= 3)
7. text(chosen.clusters, labels = names(chosen.clusters), cex = 0.7, pos = 3)

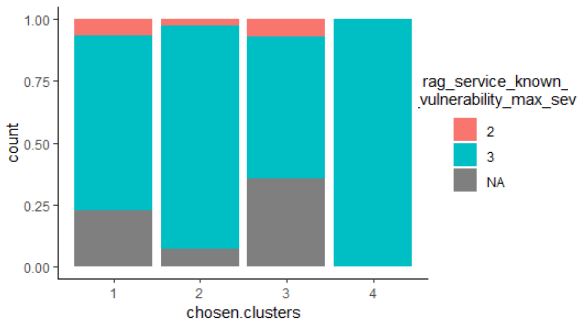
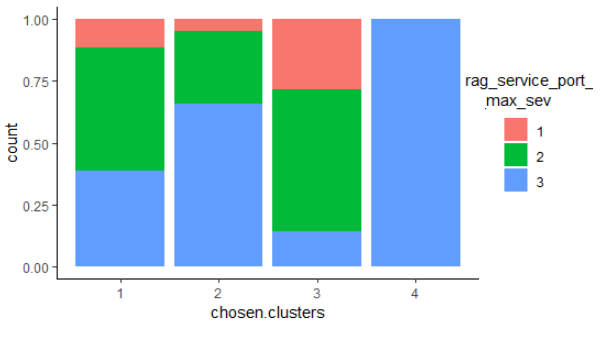


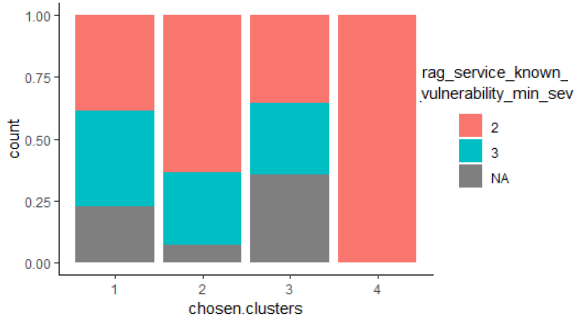
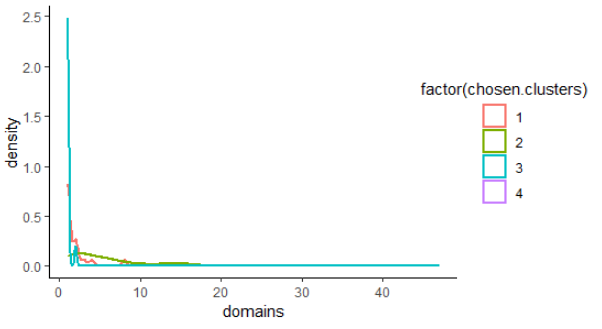
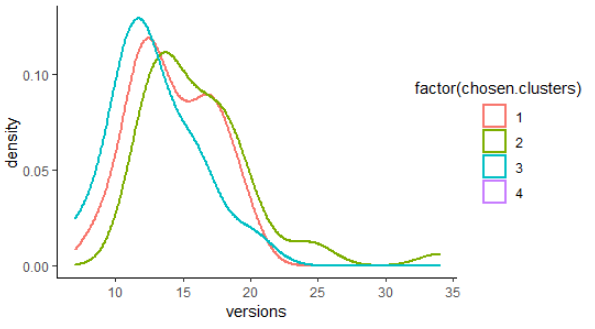
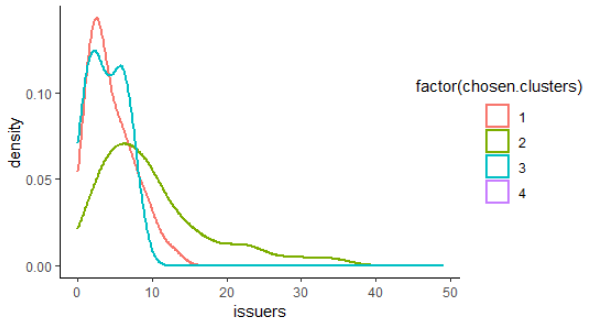
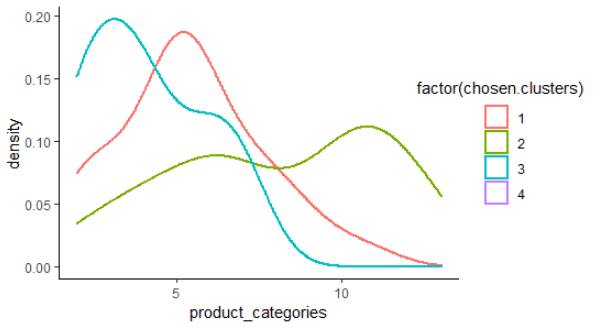
Chart

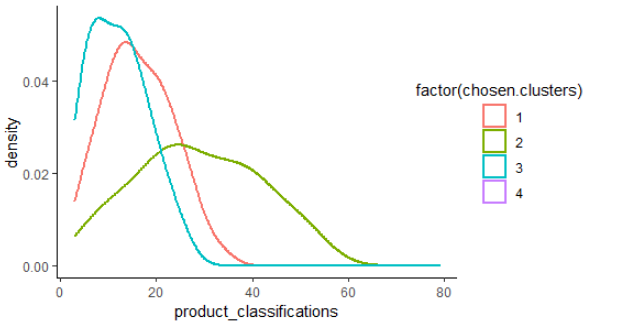
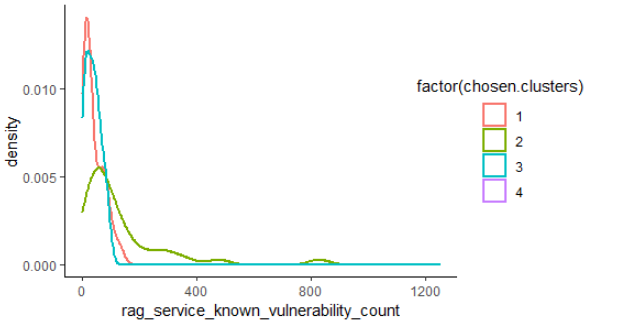
Description automatically generated with medium confidence

The final step regarding data clustering was to interpret the results to see why certain organizations were grouped together and try to understand the hidden patterns behind those groupings. For this purpose, graphs of several variables were plotted to see whether the variables differ significantly for different cluster. The code and plots are given below.

1. test1 <- cbind(chosen.clusters,final\_data)
3. # Bar plots for categorical variables
5. ggplot(test1) + theme\_classic() +
6. geom\_bar(aes(fill = update\_dates, x = chosen.clusters))
7. ggplot(test1) + theme\_classic() +
8. geom\_bar(aes(fill = rag\_service\_known\_vulnerability\_min\_sev,
9. x = chosen.clusters))
10. ggplot(test1) + theme\_classic() +
11. geom\_bar(aes(fill = rag\_service\_known\_vulnerability\_max\_sev,
12. x = chosen.clusters))
13. ggplot(test1) + theme\_classic() +
14. geom\_bar(aes(fill = rag\_service\_port\_max\_sev, x = chosen.clusters))
16. # Density plots for continuous variables
18. ggplot(test1) + theme\_classic() +
19. geom\_density(aes(color = factor(chosen.clusters), x = domains), lwd = 0.8)
20. ggplot(test1) + theme\_classic() +
21. geom\_density(aes(color = factor(chosen.clusters), x = versions), lwd = 0.8)
22. ggplot(test1) + theme\_classic() +
23. geom\_density(aes(color = factor(chosen.clusters), x = issuers), lwd = 0.8)
24. ggplot(test1) + theme\_classic() +
25. geom\_density(aes(color = factor(chosen.clusters), x = product\_categories),
26. lwd = 0.8)
27. ggplot(test1) + theme\_classic() +
28. geom\_density(aes(color = factor(chosen.clusters),
29. x = product\_classifications), lwd = 0.8)
30. ggplot(test1) + theme\_classic() +
31. geom\_density(aes(color = factor(chosen.clusters),
32. x = rag\_service\_known\_vulnerability\_count), lwd = 0.8)

Chart, bar chart

Description automatically generated 

For the plot of categorical variables, “rag\_service\_known\_vulnerability\_min\_sev”, and “rag\_service\_port\_max\_sev” showed difference in the proportion of categories with different clusters. For the continuous variables, the distributions “issuers”, “product\_categories”, and “product\_classifications” were somewhat different for different clusters. Furthermore, the organization 40 (org\_id = 630eee90-67a1-11eb-9d98-6ba16061e7ed) is considered a separate cluster. It can be due to the fact that it has the highest number of domains (47), which is much higher than any other organization.

**Location Visualisation**

The second goal of the project was to plot the locations of the organizations so as to visualize where the majority of KYND’s clients are from. For this purpose, relevant information such as organization id, country, continent, longitude, and latitude were extracted from the provided data. This information was transformed to see how many organizations originated from the same location. An interactive plot of this location data was created and is presented below.

1. data2 <- read\_excel("KYND\_data\_history.xlsx", sheet = 5)
2. icon\_salesdata <- makeAwesomeIcon(text = data2$`count of org\_id`)
3. df <- data.frame(longitude = data2$longitude, latitude = data2$latitude)
4. leaflet(df) %>%
5. addTiles() %>%
6. addProviderTiles(providers$CartoDB.Positron) %>%
7. addAwesomeMarkers(lng = data2$longitude, lat = data2$latitude,
8.  icon = icon\_salesdata)

The link for the interactive map is: [http://rpubs.com/khurram95/934772]

Furthermore, a heat map for the location data was also generated. As the count scale differ very significantly i.e. from 1 to 38676 and it meant the map won’t be able to capture the whole range effectively. Hence, the counts were converted to a log scale before plotting, which reduced the scale range. The code and plot are given below.

1. mydata <- as.tibble(data2)
3. mydata <- mydata %>%
4. dplyr::rename(
5. count\_id = "count of org\_id"
6. )
8. mydata$region[mydata$region == "United States"] <- "USA"
10. world\_map <- map\_data("world")
11. world\_map <- subset(world\_map, region != "Antarctica")
13. b <- c(0, 5, 10, 15)
15. ggplot(mydata) +
16. geom\_map(dat = world\_map, map = world\_map, aes(map\_id = region),
17. fill = "lightgray") +
18. geom\_map(map = world\_map, aes(map\_id = region, fill = log(count\_id)),
19. size = 0.5) +
20. geom\_map(dat = world\_map, map = world\_map, aes(map\_id = region), fill = NA,
21. color = "#2b2d2f", size = 0.5) +
22. scale\_fill\_gradientn(limits = c(1, 15), colours=c("yellow","orange", "red"),
23. breaks=b, labels=format(b)) +
24. expand\_limits(x = world\_map$long, y = world\_map$lat)

Map

Description automatically generated

From the plots, we can see that most of the IPs are generated from North America and Europe. The lowest were generated from Africa.

**Limitations**

First issue faced was that the data set was very large and had to be transformed, because the original form could not be processes with the available computer resources. In terms of clustering algorithms, it was observed that there was a scarcity of algorithms available to be used for data sets containing both numerical and categorical variables. It is a concerning issue as real-world data is usually contains mixed type of data. The available algorithms also had limitations of their own. For instance, K-proto could not handle missing values and thus could not be used. Furthermore, the output of ROCK algorithm is of a different format than most other clustering algorithms output. As a result, it could not be put and used in available cluster evaluation tools and there were no evaluation tools provided in the package. This meant that there was no method to test the efficiency of clusters created.

**Conclusion**

The data set was first cleaned and transformed in a format that can be read by the clustering algorithms. K-proto, ROCK , and hierarchical clustering algorithms were selected to be applied because of their ability to handle mixed data. First, a simulated data set was created to test which of the mentioned algorithms was performing better. Various internal and external validation metrics were used to test the performance. K-proto couldn’t handle NA values. ROCK algorithm is tailored towards binary data and was producing clusters with no values. Hierarchical clustering was then applied with five different linkage methods. Ward’s method and centroid linkage produced gave better outcomes for validation metrics and hence, were selected to be able to the original data set. Centroid linkage output was not desirable, as it was suggesting 2 clusters, with 1 cluster only containing a single observation. Ward’s method produced better results and suggested 4 clusters as the optimum partition. The output was then interpreted by visualising several variables within the data to see if their proportion or distribution changes with clusters and the result had considerable differences. Thus it was concluded that the algorithm was able to find hidden patterns within the data and the clusters were not entirely meaningless. However, the evaluation metrics only showed low efficiency of results, indicating room for improvement in algorithm used. Furthermore, the organisations locations were plotted to see the frequency of organizations IP addresses present in each country.

**Acknowledgements**

I would like to thank Dr Bruno Santos for supervising this project and helping me throughout the project. I am also thankful to KYND company and their representatives Will Hesselmann, Paulo Ferreira, and Andrew Thomas for providing the data set to work on.

**References**

1. L. Rokach , Clustering methods, Data Mining and Knowledge Discovery Hand- book, Springer, 2005, pages 331–352.
2. C. Fraley , A.E. Raftery , How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis, Department of Statistics University of Washington, 1998 Technical Report No. 329 .
3. A. Saxena et al, A review of clustering techniques and developments, Neurocomputing, volume 267, 2017, pages 664-681,
4. R. Xu , D. Wunsch , Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678 .
5. T. Zhang , R. Ramakrishnan , M. Linvy , BIRCH: an efficient method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 1996.
6. A. Periklis , Data Clustering Techniques, University of Toronto, 2002 .
7. S. Guha , R. Rastogi , S. Kyuseok , CURE: An Efficient Clustering Algorithm For Large Databases, ACM, 1998 .
8. S. Guha , R. Rastogi , K. Shim , Rock: a robust clustering algorithm for categorical attributes, in: Proceedings of the IEEE Conference on Data Engineering, 1999 .
9. Karypis, George & Han, Eui-Hong & Kumar, Vipin. (1999). CHAMELEON A hierarchical clustering algorithm using dynamic modeling. Computer. 32. 68 - 75. 10.1109/2.781637.
10. D. Lam , D.C. Wunsch , Clustering, academic press library in signal processing, Signal Process. Theory Mach. Learn. 1 (2014) 1115–1149 .
11. R. Xu , D. Wunsch , Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678 . **duplicate**
12. Lakshmi, K., Visalakshi, N.K., Shanthi, S. and Parvathavarthini, S., 2017. CLUSTERING CATEGORICAL DATA USING k-MODES BASED ON CUCKOO SEARCH OPTIMIZATION ALGORITHM. Ictact journal on Soft Computing, 8(1).
13. Huang, Z., 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data mining and knowledge discovery, 2(3), pp.283-304.
14. P. Sneath , R. Sokal , Numerical Taxonomy, W.H. Freeman Co, San Francisco, CA, 1973.
15. Ranjitkar, Hari Sagar and Sudip Karki. “Comparison of A\*, Euclidean and Manhattan distance using Influence map in MS. Pac-Man.” (2016).
16. T. Sørensen , A method of establishing groups of equal amplitude in plant so- ciology based on similarity of species and its application to analyses of the vegetation on Danish commons, K. Dan. Vidensk. Selsk. 5 (4) (1948) 1–34.
17. L.R. Dice , Measures of the amount of ecologic association between species, Ecology 26 (3) (1945) 297–302.
18. Anand, D., 2022. Gower’s Distance. [online] Medium. Available at: <https://medium.com/analytics-vidhya/gowers-distance-899f9c4bd553> [Accessed 1 August 2022].
19. S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Science, Vol. 25, No. 5, 2000.
20. Murphy, P., 2022. Clustering Data in R. [online] Rstudio-pubs-static.s3.amazonaws.com. Available at: <https://rstudio-pubs-static.s3.amazonaws.com/599072\_93cf94954aa64fc7a4b99ca524e5371c.html> [Accessed 9 August 2022].
21. Jarman, A.M., 2020. Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method. Georgia Southern University.
22. Vijaya, S. Sharma and N. Batra, "Comparative Study of Single Linkage, Complete Linkage, and Ward Method of Agglomerative Clustering," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019, pp. 568-573, doi: 10.1109/COMITCon.2019.8862232.
23. Theodoridis S, Koutroubas K (2008). Pattern Recognition. 4th edition. Academic Press.
24. D.L. Olson , D. Delen , Advanced Data Mining Techniques, first ed., Springer, 2008 .
25. D.M.W. Powers , Evaluation: from precision, recall and F-factor to ROC, in- formedness, markedness & correlation, J. Mach. Learn. Technol. 2 (1) (2007) 37–63 .
26. W.M. Rand , Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850 .
27. V. Rijsbergen , Information Retrieval, Butterworths, London, 1979 .
28. P. Jaccard , Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines, Bull. Soc. Vaud. Sci. Nat. 37 (1901) 241–272 .
29. E.B. Fowlkes , C.L. Mallows , A method for comparing two hierarchical cluster- ings, J. Am. Stat. Assoc. 78 (383) (1983) 553–569 2010 .
30. S.V. Stehman , Selecting and interpreting measures of thematic classification accuracy, Remote Sens. Environ. 62 (1) (1997) 77–89 .
31. C. Chen, L. Pau, and P. Wang, Cluster analysis and related issue, R. Dubes Eds. Handbook of Pattern Recognition and Computer Vision, World Scientific, Sin- gapore, pp. 3–32.
32. A. Jain , R. Dubes , Algorithms for Clustering Data, Prentice-Hall, Englewood, Cliffs, NJ, 1988 .
33. Huang, Z. 1997. Clustering large data sets with mixed numeric and categorical values. Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore: World Scientific, pp. 21–34
34. Huang, Z., 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data mining and knowledge discovery, 2(3), pp.283-304.
35. S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Science, Vol. 25, No. 5, 2000. **duplicate**
36. WhatIs.com. 2022. What is a domain in computing? [online] Available at: <https://www.techtarget.com/whatis/definition/domain#:~:text=1)%20In%20computing%20and%20telecommunication,a%20set%20of%20network%20addresses> [Accessed 24 August 2022].
37. Blog.hubspot.com. 2022. What's a Subdomain & How Is It Used? [online] Available at: <https://blog.hubspot.com/website/what-is-a-subdomain#:~:text=What%20is%20a%20subdomain%20name,the%20rest%20of%20your%20website.> [Accessed 24 August 2022].
38. Cloudfare.com. 2022. What is a domain name registrar? [online] Available at: <https://www.cloudflare.com/en-gb/learning/dns/glossary/what-is-a-domain-name-registrar/> [Accessed 24 August 2022].
39. Higher Logic. 2022. SPF, DKIM, DMARC: The 3 Pillars of Email Authentication - Higher Logic. [online] Available at: <https://www.higherlogic.com/blog/spf-dkim-dmarc-email-authentication/#:~:text=The%20two%20primary%20authentication%20protocols,if%20they%20receive%20unauthenticated%20mail> [Accessed 24 August 2022].
40. www.kaspersky.com. 2022. What is an IP Address – Definition and Explanation. [online] Available at: <https://www.kaspersky.com/resource-center/definitions/what-is-an-ip-address> [Accessed 24 August 2022].
41. Definitions, C. and Hope, C., 2022. What is a Certificate?. [online] Computerhope.com. Available at: <https://www.computerhope.com/jargon/c/certific.htm> [Accessed 24 August 2022].